INTERNATIONAL STANDARD

ISO/IEC 9075-11

Fourth edition
2016-12-15

Information technology — Database languages — SQL —

Part 11:
Information and definition schemas (SQL/Schemata)

Technologies de l'information — Langages de base de données — SQL —

Partie 11: Schémas des informations et des définitions (SQL/ Schemata)

Contents

Page

Tables

Table   **Page**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 32 *Data management and interchange*.

This fourth edition of ISO/IEC 9075-11 cancels and replaces the third edition (ISO/IEC 9075-11:2011), which has been technically revised.

A list of all parts in the ISO/IEC 9075 series, published under the general title *Information technology — Database languages — SQL*, can be found on the ISO website.

NOTE   The individual parts of multi-part standards are not necessarily published together. New editions of one or more parts can be published without publication of new editions of other parts.

Introduction

The organization of this part of ISO/IEC 9075 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 9075.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 9075.

4) Clause 4, "Concepts", presents concepts used in the definition of Persistent SQL modules.

5) Clause 5, "Information Schema", defines viewed tables that contain schema information.

6) Clause 6, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.

7) Clause 7, "Conformance", defines the criteria for conformance to this part of ISO/IEC 9075.

8) Annex A, "SQL Conformance Summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

9) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

10) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.

11) Annex D, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.

12) Annex E, "Incompatibilities with ISO/IEC 9075:2011", is an informative Annex. It lists incompatibilities with the previous version of this part of ISO/IEC 9075.

13) Annex F, "SQL feature taxonomy", is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by an identifier and a short descriptive name. This taxonomy is used to specify conformance.

14) Annex G, "Defect reports not addressed in this edition of this part of ISO/IEC 9075", is an informative Annex. It describes the Defect Reports that were known at the time of publication of this part of this International Standard. Each of these problems is a problem carried forward from the previous edition of ISO/IEC 9075. No new problems have been created in the drafting of this edition of this International Standard.

In the text of this part of ISO/IEC 9075, Clauses and Annexes begin new odd-numbered pages, and in Clause 5, "Information Schema", through Clause 7, "Conformance", Subclauses begin new pages. Any resulting blank space is not significant.

Information technology — Database languages — SQL —

Part 11:
Information and Definition Schemas (SQL/Schemata)

1 Scope

This part of ISO/IEC 9075 specifies an Information Schema and a Definition Schema that describes:

— The structure and integrity constraints of SQL-data.

— The security and authorization specifications relating to SQL-data.

— The features and subfeatures of ISO/IEC 9075, and the support that each of these has in an SQL-implementation.

— The SQL-implementation information and sizing items of ISO/IEC 9075 and the values supported by an SQL-implementation.

(Blank page)

2   Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1   ISO and IEC standards

[ISO9075-1] ISO/IEC 9075-1:2016, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[ISO9075-2] ISO/IEC 9075-2:2016, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

ISO/IEC 9075-11:2016(E)

(Blank page)

3 Definitions, notations, and conventions

This Clause modifies Clause 3, "Definitions, notations, and conventions", in ISO/IEC 9075-2.

3.1 Conventions

This Subclause modifies Subclause 3.3, "Conventions", in ISO/IEC 9075-2.

Insert this paragraph The Descriptions in Clause 6, "Definition Schema", sometimes specify values that are to appear in rows of base tables. When such a value is given as a sequence of capital letters enclosed in <double quote>s, it denotes the same value as would be denoted by the <character string literal> obtained by replacing the enclosing <double quote>s by <quote>s. The need for such notation arises when the column in question sometimes, in other rows, contains character strings denoting SQL expressions, possibly even <character string literal>s.

Definitions, notations, and conventions 5

(Blank page)

4   Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1   Introduction to the Definition Schema

The Definition Schema base tables are defined as being in a schema named DEFINITION_SCHEMA. The table definitions are as complete as the definitional power of SQL allows. The table definitions are supplemented with assertions where appropriate.

The only purpose of the Definition Schema is to provide a data model to support the Information Schema and to assist understanding. An SQL-implementation need do no more than simulate the existence of the Definition Schema, as viewed through the Information Schema views. The specification does not imply that an SQL-implementation shall provide the functionality in the manner described in the Definition Schema.

A Definition Schema *DS* completely describes all contents of every schema, excluding itself, but including the Information Schema, contained in the catalog *C* that contains *DS*. When some object, such as a constraint or a view, references an object contained in a schema contained in a catalog *OC*, *OC* ≠ *C*, the reference to that object cannot be confirmed, because the information about objects contained in *OC* is not necessarily available to *DS*. The constraints defined in *DS* can thus guarantee consistency only within *C*.

Because the DEFINITION_SCHEMA references objects that are only defined in the INFORMATION_SCHEMA, it would not normally be possible to create these two schemata using the CREATE SCHEMA statements as written. It is assumed that an INFORMATION_SCHEMA and its underlying DEFINITION_SCHEMA are created in some implementation-dependent way at the time that a catalog is created.

In addition to the descriptors created by the effective execution of the CREATE INFORMATION_SCHEMA statement, the DEFINITION_SCHEMA must also contain descriptions of other object defined by the standard, *e.g.*, the character repertoire SQL_IDENTIFER.

The way in which certain constraints are expressed caters for the possibility that an object is being referenced that exists in a catalog that is outside the purview of the Definition Schema containing the reference in question. For example, the definition of the VIEW_TABLE_USAGE base table in Subclause 6.70, "VIEW_TABLE_USAGE base table", includes the following constraint:

```
CONSTRAINT VIEW_TABLE_USAGE_CHECK_REFERENCES_TABLES
    CHECK ( TABLE_CATALOG NOT IN
            ( SELECT CATALOG_NAME
              FROM SCHEMATA )
        OR
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM TABLES ) )
```

Either the table being used by the view exists in a catalog within this Definition Schema's purview, in which case its existence is guaranteed, or it is assumed but not guaranteed to exist in some catalog that is outside this Definition Schema's purview.

Because <unqualified schema name>s are prohibited by SR 10) of Subclause 5.4, "Names and identifiers", in [ISO9075-2], from specifying DEFINITION_SCHEMA, the Definition Schema cannot normally be accessed in an SQL-statement. However, view definitions in the Information Schema assume the existence of the Definition Schema and reference base tables whose <schema name> is DEFINITION_SCHEMA. They use the Definition Schema to define the content of the Information Schema. Regardless of SR 14) of Subclause 5.4, "Names and identifiers", in [ISO9075-2], the <schema name> DEFINITION_SCHEMA is never qualified by a <catalog name>. It is implementation-defined whether the DEFINITION_SCHEMA referenced by an INFORMATION_SCHEMA describes schemas in catalogs other than the catalog in which the INFORMATION_SCHEMA is located.

4.2 Introduction to the Information Schema

The views of the Information Schema are viewed tables defined in terms of the base tables of the Definition Schema.

The Information Schema views are defined as being in a schema named INFORMATION_SCHEMA, enabling these views to be accessed in the same way as any other tables in any other schema. SELECT on most of these views is granted to PUBLIC WITH GRANT OPTION, so that they can be queried by any user and so that SELECT privilege can be further granted on views that reference these Information Schema views. No other privilege is granted on them, so they cannot be updated.

In order to provide access to the same information that is available via the INFORMATION_SCHEMA to an SQL-Agent in an SQL-environment where the SQL-implementation does not support Feature F391, "Long identifiers", alternative views are provided that use only short identifiers. The Information Schema also contains a small number of domains on which the columns of the Definition Schema are based. USAGE on all these domains is granted to PUBLIC WITH GRANT OPTION, so that they can be used by any user.

An SQL-implementation may define objects that are associated with INFORMATION_SCHEMA that are not defined in this Clause. An SQL-implementation or any future version of ISO/IEC 9075 may also add columns to tables that are defined in this Clause.

> NOTE 2 — The Information Schema tables may be supposed to be represented in the Definition Schema in the same way as any other tables, and are hence self-describing.

> NOTE 3 — The Information Schema is a definition of the SQL data model, specified as an SQL-schema, in terms of <SQL schema statement>s as defined in ISO/IEC 9075. Constraints defined in this Clause are not actual SQL constraints.

The representation of an <identifier> in the base tables and views of the Information Schema is by a character string corresponding to its <identifier body> (in the case of a <regular identifier>) or its <delimited identifier body> (in the case of a <delimited identifier>). Within this character string, any lower-case letter appearing in a <regular identifier> is replaced by the equivalent upper-case letter, and any <doublequote symbol> appearing in a <delimited identifier body> is replaced by a <double quote>. Where an <actual identifier> has multiple forms that are equal according to the rules of Subclause 8.2, "<comparison predicate>", in [ISO9075-2], the form stored is that encountered at definition time.

5   Information Schema

5.1   INFORMATION_SCHEMA Schema

Function

Identify the schema that is to contain the Information Schema tables.

Definition

```
CREATE SCHEMA INFORMATION_SCHEMA
    AUTHORIZATION INFORMATION_SCHEMA;
```

Conformance Rules

None.

5.2   INFORMATION_SCHEMA_CATALOG_NAME view

Function

Identify the catalog that contains the Information Schema.

Definition

```
CREATE VIEW INFORMATION_SCHEMA_CATALOG_NAME AS
    SELECT CATALOG_NAME
    FROM DEFINITION_SCHEMA.CATALOG_NAME
    WHERE DEFINITION_SCHEMA.CATALOG_NAME = 'CN';

GRANT SELECT ON TABLE INFORMATION_SCHEMA_CATALOG_NAME
    TO PUBLIC WITH GRANT OPTION;
```

Description

1)   *CN* is the name of the catalog in which this Information Schema resides.

Conformance Rules

1)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.

2)   Without Feature F651, "Catalog name qualifiers", conforming SQL language shall not reference INFOR-MATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.

5.3   CARDINAL_NUMBER domain

Function

Define a domain that contains a non-negative number.

Definition

```
CREATE DOMAIN CARDINAL_NUMBER AS INTEGER
    CONSTRAINT CARDINAL_NUMBER_DOMAIN_CHECK
      CHECK ( VALUE >= 0 );

GRANT USAGE ON DOMAIN CARDINAL_NUMBER
    TO PUBLIC WITH GRANT OPTION;
```

Description

1) The domain CARDINAL_NUMBER contains any non-negative number that is less than or equal to the implementation-defined maximum for INTEGER.

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CARDINAL_NUMBER.

5.4   CHARACTER_DATA domain

Function

Define a domain that contains any character data.

Definition

```
CREATE DOMAIN CHARACTER_DATA AS
    CHARACTER VARYING (ML)
    CHARACTER SET SQL_TEXT;

GRANT USAGE ON DOMAIN CHARACTER_DATA
    TO PUBLIC WITH GRANT OPTION;
```

Description

1)   This domain specifies any character data.

2)   *ML* is the implementation-defined maximum length of a variable-length character string.

Conformance Rules

1)   Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CHARACTER_DATA.

5.5 SQL_IDENTIFIER domain

Function

Define a domain that contains all valid <identifier body>s and <delimited identifier body>s.

Definition

```
CREATE DOMAIN SQL_IDENTIFIER AS
    CHARACTER VARYING (L)
    CHARACTER SET SQL_IDENTIFIER;

GRANT USAGE ON DOMAIN SQL_IDENTIFIER
    TO PUBLIC WITH GRANT OPTION;
```

Description

1) This domain specifies all variable-length character values that conform to the rules for formation and representation of an SQL <identifier body> or an SQL <delimited identifier body>.

> NOTE 4 — There is no way in SQL to specify a <domain constraint> that would be true for the body of any valid SQL <regular identifier> or <delimited identifier> and false for all other character string values.

2) *L* is the implementation-defined maximum length of <identifier body> and <delimited identifier body>.

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IDENTIFIER.

5.6   TIME_STAMP domain

Function

Define a domain that contains a timestamp.

Definition

```
CREATE DOMAIN TIME_STAMP AS TIMESTAMP(2) WITH TIME ZONE;

GRANT USAGE ON DOMAIN TIME_STAMP
    TO PUBLIC WITH GRANT OPTION;
```

Description

1)   The domain TIME_STAMP contains an SQL timestamp value.

Conformance Rules

1)   Without Feature F251, "Domain support", and Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.TIME_STAMP.

5.7   YES_OR_NO domain

Function

Define a domain that contains a character string value, but allows only two possible strings, YES or NO.

Definition

```
CREATE DOMAIN YES_OR_NO AS
    CHARACTER VARYING (3)
    CHARACTER SET SQL_IDENTIFIER
      CONSTRAINT YES_OR_NO_CHECK
        CHECK (VALUE IN ( 'YES', 'NO' ) );

GRANT USAGE ON DOMAIN YES_OR_NO
    TO PUBLIC WITH GRANT OPTION;
```

Description

1)   This Domain specifies all boolean values, which are needed in the definition schema, encoded in the two strings 'YES' and 'NO'.

Conformance Rules

1)   Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.YES_OR_NO.

5.8   ADMINISTRABLE_ROLE_AUTHORIZATIONS view

Function

Identify role authorizations for which the current user or role has WITH ADMIN OPTION.

Definition

```
CREATE VIEW ADMINISTRABLE_ROLE_AUTHORIZATIONS AS
    SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORS
    WHERE ROLE_NAME IN
        ( SELECT ROLE_NAME
          FROM INFORMATION_SCHEMA.APPLICABLE_ROLES
          WHERE IS_GRANTABLE = 'YES' );

GRANT SELECT ON TABLE ADMINISTRABLE_ROLE_AUTHORIZATIONS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.

5.9 APPLICABLE_ROLES view

Function

Identifies the applicable roles for the current user.

Definition

```
CREATE RECURSIVE VIEW APPLICABLE_ROLES ( GRANTEE, ROLE_NAME, IS_GRANTABLE ) AS
    ( ( SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
        FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORS
        WHERE ( GRANTEE IN
                ( CURRENT_USER, 'PUBLIC' )
            OR
                GRANTEE IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
      UNION
      ( SELECT RAD.GRANTEE, RAD.ROLE_NAME, RAD.IS_GRANTABLE
        FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORS RAD
          JOIN
            APPLICABLE_ROLES R
            ON
              RAD.GRANTEE = R.ROLE_NAME ) );

GRANT SELECT ON TABLE APPLICABLE_ROLES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.APPLICABLE_ROLES.

5.10 ASSERTIONS view

Function

Identify the assertions defined in this catalog that are owned by a given user or role.

Definition

```
CREATE VIEW ASSERTIONS AS
    SELECT A.CONSTRAINT_CATALOG, A.CONSTRAINT_SCHEMA, A.CONSTRAINT_NAME,
           A.IS_DEFERRABLE, A.INITIALLY_DEFERRED
    FROM DEFINITION_SCHEMA.ASSERTIONS AS A
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
      ON ( ( A.CONSTRAINT_CATALOG, A.CONSTRAINT_SCHEMA )
         = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
         OR
           S.SCHEMA_OWNER IN
           ( SELECT ER.ROLE_NAME
             FROM ENABLED_ROLES AS ER ) )
      AND
        A.CONSTRAINT_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ASSERTIONS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F521, "Assertions", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.ASSERTIONS.

5.11 ATTRIBUTES view

This Subclause is modified by Subclause 24.1, "ATTRIBUTES view", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 21.3, "ATTRIBUTES view", in ISO/IEC 9075-14.

Function

Identify the attributes of user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ATTRIBUTES AS
      SELECT DISTINCT
        UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
        A.ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
        DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
        D1.CHARACTER_SET_CATALOG, D1.CHARACTER_SET_SCHEMA, D1.CHARACTER_SET_NAME,
        D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
        NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
        D1.USER_DEFINED_TYPE_CATALOG AS ATTRIBUTE_UDT_CATALOG,
        D1.USER_DEFINED_TYPE_SCHEMA AS ATTRIBUTE_UDT_SCHEMA,
        D1.USER_DEFINED_TYPE_NAME AS ATTRIBUTE_UDT_NAME,
        D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
        MAXIMUM_CARDINALITY, A.DTD_IDENTIFIER, IS_DERIVED_REFERENCE_ATTRIBUTE,
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
        DECLARED_NUMERIC_SCALE
      FROM ( DEFINITION_SCHEMA.ATTRIBUTES AS A
          LEFT JOIN
            DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
            ON ( ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME,
                  'USER-DEFINED TYPE', A.DTD_IDENTIFIER )
              = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                  D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
      WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME ) IN
          ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG,
                  UDTP.USER_DEFINED_TYPE_SCHEMA,
                   UDTP.USER_DEFINED_TYPE_NAME
            FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
            WHERE ( UDTP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    UDTP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
        AND
          A.UDT_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ATTRIBUTES
```

```
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ATTRIBUTES.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ATTRIBUTES.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.12 CHARACTER_SETS view

Function

Identify the character sets defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW CHARACTER_SETS AS
    SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
            CHARACTER_REPERTOIRE, FORM_OF_USE, DEFAULT_COLLATE_CATALOG,
            DEFAULT_COLLATE_SCHEMA, DEFAULT_COLLATE_NAME
    FROM DEFINITION_SCHEMA.CHARACTER_SETS
    WHERE ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
            'CHARACTER SET') IN
            ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME,
                    UP.OBJECT_TYPE
              FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
              WHERE ( UP.GRANTEE IN
                        ( 'PUBLIC', CURRENT_USER )
                    OR
                        UP.GRANTEE IN
                        ( SELECT ROLE_NAME
                            FROM ENABLED_ROLES ) ) )
                AND
                    CHARACTER_SET_CATALOG
                = ( SELECT CATALOG_NAME
                    FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE CHARACTER_SETS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CHARACTER_SETS.

5.13 CHECK_CONSTRAINT_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which a domain constraint, table check constraint or assertion defined in this catalog is dependent.

Definition

```
CREATE VIEW CHECK_CONSTRAINT_ROUTINE_USAGE AS
    SELECT CCRU.CONSTRAINT_CATALOG, CCRU.CONSTRAINT_SCHEMA, CCRU.CONSTRAINT_NAME,
           CCRU.SPECIFIC_CATALOG, CCRU.SPECIFIC_SCHEMA, CCRU.SPECIFIC_NAME
    FROM DEFINITION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE AS CCRU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( CCRU.SPECIFIC_CATALOG, CCRU.SPECIFIC_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          CCRU.SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE CHECK_CONSTRAINT_ROUTINE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.

5.14 CHECK_CONSTRAINTS view

Function

Identify the check constraints defined in this catalog that are owned by a given user or role.

Definition

```
CREATE VIEW CHECK_CONSTRAINTS AS
    SELECT CC.CONSTRAINT_CATALOG, CC.CONSTRAINT_SCHEMA,
           CC.CONSTRAINT_NAME, CC.CHECK_CLAUSE
    FROM DEFINITION_SCHEMA.CHECK_CONSTRAINTS AS CC
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
      ON ( ( CC.CONSTRAINT_CATALOG, CC.CONSTRAINT_SCHEMA )
         = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
         OR
           S.SCHEMA_OWNER IN
           ( SELECT ER.ROLE_NAME
             FROM ENABLED_ROLES AS ER ) )
      AND CC.CONSTRAINT_CATALOG
       = ( SELECT ISCN.CATALOG_NAME
           FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE CHECK_CONSTRAINTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.15 COLLATIONS view

Function

Identify the character collations defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW COLLATIONS AS
    SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
           PAD_ATTRIBUTE
    FROM DEFINITION_SCHEMA.COLLATIONS
    WHERE ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
            'COLLATION' ) IN
          ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME,
                   UP.OBJECT_TYPE
            FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
            WHERE ( UP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    UP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
      AND COLLATION_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE COLLATIONS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F690, "Collation support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLLATIONS.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLLATIONS.

5.16 COLLATION_CHARACTER_SET_APPLICABILITY view

Function

Identify the character sets to which each collation is applicable.

Definition

```
CREATE VIEW COLLATION_CHARACTER_SET_APPLICABILITY AS
    SELECT CCSA.COLLATION_CATALOG, CCSA.COLLATION_SCHEMA, CCSA.COLLATION_NAME,
          CCSA.CHARACTER_SET_CATALOG, CCSA.CHARACTER_SET_SCHEMA, CCSA.CHARACTER_SET_NAME
    FROM DEFINITION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY AS CCSA
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( CCSA.COLLATION_CATALOG, CCSA.COLLATION_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND CCSA.COLLATION_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE COLLATION_CHARACTER_SET_APPLICABILITY
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.

2) Without Feature F690, "Collation support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.

5.17 COLUMN_COLUMN_USAGE view

Function

Identify each case where a generated column depends on a base column in a base table owned by a given user or role.

Definition

```
CREATE VIEW COLUMN_COLUMN_USAGE AS
    SELECT CCU.TABLE_CATALOG, CCU.TABLE_SCHEMA, CCU.TABLE_NAME,
           CCU.COLUMN_NAME, CCU.DEPENDENT_COLUMN
    FROM DEFINITION_SCHEMA.COLUMN_COLUMN_USAGE AS CCU
    JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
      ON ( CCU.TABLE_CATALOG, CCU.TABLE_SCHEMA )
       = ( S.CATALOG_NAME, S.SCHEMA_NAME )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
           ( SELECT ER.ROLE_NAME
             FROM ENABLED_ROLES AS ER ) )
      AND
          CCU.TABLE_CATALOG
       = ( SELECT ISCN.CATALOG_NAME
           FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE COLUMN_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_COLUMN_USAGE.

2) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_COLUMN_USAGE.

3) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_COLUMN_USAGE.

5.18   COLUMN_DOMAIN_USAGE view

Function

Identify the columns defined that are dependent on a domain defined in this catalog and owned by a user or role.

Definition

```
CREATE VIEW COLUMN_DOMAIN_USAGE AS
    SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
           C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME
    FROM ( DEFINITION_SCHEMA.COLUMNS AS C
         JOIN
           ( DEFINITION_SCHEMA.DOMAINS AS D
           JOIN
             DEFINITION_SCHEMA.SCHEMATA AS S
             ON ( ( D.DOMAIN_CATALOG, D.DOMAIN_SCHEMA )
                 = ( S.CATALOG_NAME, S.SCHEMA_NAME ) ) )
         USING ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          DOMAIN_NAME IS NOT NULL
      AND
          DOMAIN_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE COLUMN_DOMAIN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_DOMAIN_USAGE.

2) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_DOMAIN_USAGE.

3) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_DOMAIN_USAGE.

5.19 COLUMN_PRIVILEGES view

Function

Identify the privileges on columns of tables defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW COLUMN_PRIVILEGES AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
        PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
    WHERE ( GRANTEE IN
            ( 'PUBLIC', CURRENT_USER )
        OR
            GRANTEE IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES )
        OR
            GRANTOR
          = CURRENT_USER
        OR
            GRANTOR IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
      AND
        TABLE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE COLUMN_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_PRIVILEGES.

5.20   COLUMN_UDT_USAGE view

Function

Identify the columns defined that are dependent on a user-defined type defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW COLUMN_UDT_USAGE AS
    SELECT DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
           C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME
    FROM ( DEFINITION_SCHEMA.COLUMNS AS C
        JOIN
          ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
          JOIN
            DEFINITION_SCHEMA.SCHEMATA AS S
          ON ( DTD.USER_DEFINED_TYPE_CATALOG, DTD.USER_DEFINED_TYPE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
        ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
             'TABLE', C.DTD_IDENTIFIER )
           = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA, DTD.OBJECT_NAME,
             DTD.OBJECT_TYPE, DTD.DTD_IDENTIFIER ) ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          DTD.DATA_TYPE = 'USER-DEFINED' ;

GRANT SELECT ON TABLE COLUMN_UDT_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMN_UDT_USAGE.

5.21   COLUMNS view

This Subclause is modified by Subclause 24.3, *"COLUMNS view", in ISO/IEC 9075-9.*
This Subclause is modified by Subclause 21.4, *"COLUMNS view", in ISO/IEC 9075-14.*

Function

Identify the columns of tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW COLUMNS AS
    SELECT DISTINCT
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      C.COLUMN_NAME, ORDINAL_POSITION, COLUMN_DEFAULT, IS_NULLABLE,
      COALESCE (D1.DATA_TYPE, D2.DATA_TYPE) AS DATA_TYPE,
      COALESCE (D1.CHARACTER_MAXIMUM_LENGTH, D2.CHARACTER_MAXIMUM_LENGTH)
              AS CHARACTER_MAXIMUM_LENGTH,
      COALESCE (D1.CHARACTER_OCTET_LENGTH, D2.CHARACTER_OCTET_LENGTH)
              AS CHARACTER_OCTET_LENGTH,
      COALESCE (D1.NUMERIC_PRECISION, D2.NUMERIC_PRECISION)
              AS NUMERIC_PRECISION,
      COALESCE (D1.NUMERIC_PRECISION_RADIX, D2.NUMERIC_PRECISION_RADIX)
              AS NUMERIC_PRECISION_RADIX,
      COALESCE (D1.NUMERIC_SCALE, D2.NUMERIC_SCALE)
              AS NUMERIC_SCALE,
      COALESCE (D1.DATETIME_PRECISION, D2.DATETIME_PRECISION)
              AS DATETIME_PRECISION,
      COALESCE (D1.INTERVAL_TYPE, D2.INTERVAL_TYPE)
              AS INTERVAL_TYPE,
      COALESCE (D1.INTERVAL_PRECISION, D2.INTERVAL_PRECISION)
              AS INTERVAL_PRECISION,
      COALESCE (D1.CHARACTER_SET_CATALOG, D2.CHARACTER_SET_CATALOG)
              AS CHARACTER_SET_CATALOG,
      COALESCE (D1.CHARACTER_SET_SCHEMA, D2.CHARACTER_SET_SCHEMA)
              AS CHARACTER_SET_SCHEMA,
      COALESCE (D1.CHARACTER_SET_NAME, D2.CHARACTER_SET_NAME)
              AS CHARACTER_SET_NAME,
      COALESCE (D1.COLLATION_CATALOG, D2.COLLATION_CATALOG)
              AS COLLATION_CATALOG,
      COALESCE (D1.COLLATION_SCHEMA, D2.COLLATION_SCHEMA)
              AS COLLATION_SCHEMA,
      COALESCE (D1.COLLATION_NAME, D2.COLLATION_NAME)
              AS COLLATION_NAME,
      DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
      COALESCE (D1.USER_DEFINED_TYPE_CATALOG, D2.USER_DEFINED_TYPE_CATALOG)
              AS UDT_CATALOG,
      COALESCE (D1.USER_DEFINED_TYPE_SCHEMA, D2.USER_DEFINED_TYPE_SCHEMA)
              AS UDT_SCHEMA,
      COALESCE (D1.USER_DEFINED_TYPE_NAME, D2.USER_DEFINED_TYPE_NAME)
              AS UDT_NAME,
      COALESCE (D1.SCOPE_CATALOG, D2.SCOPE_CATALOG) AS SCOPE_CATALOG,
      COALESCE (D1.SCOPE_SCHEMA, D2.SCOPE_SCHEMA) AS SCOPE_SCHEMA,
```

```
        COALESCE (D1.SCOPE_NAME, D2.SCOPE_NAME) AS SCOPE_NAME,
        COALESCE (D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY)
                AS MAXIMUM_CARDINALITY,
        COALESCE (D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER)
                AS DTD_IDENTIFIER,
        IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
        IDENTITY_START, IDENTITY_INCREMENT,
        IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE,
        IS_GENERATED, GENERATION_EXPRESSION, IS_SYSTEM_TIME_PERIOD_START,
        IS_SYSTEM_TIME_PERIOD_END, SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION,
        IS_UPDATABLE,
        COALESCE (D1.DECLARED_DATA_TYPE, D2.DECLARED_DATA_TYPE)
                AS DECLARED_DATA_TYPE,
        COALESCE (D1.DECLARED_NUMERIC_PRECISION, D2.DECLARED_NUMERIC_PRECISION)
                AS DECLARED_NUMERIC_PRECISION,
        COALESCE (D1.DECLARED_NUMERIC_SCALE, D2.DECLARED_NUMERIC_SCALE)
                AS DECLARED_NUMERIC_SCALE
  FROM ( ( DEFINITION_SCHEMA.COLUMNS AS C
          LEFT JOIN
            DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
            ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
                  'TABLE', C.DTD_IDENTIFIER )
              = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                  D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) ) )
        LEFT JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
          ON ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
                'DOMAIN', C.DTD_IDENTIFIER )
            = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
                D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
  WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME ) IN
          ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME, CP.COLUMN_NAME
            FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
            WHERE ( CP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    CP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
      AND
          C.TABLE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE COLUMNS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMNS.

2) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMNS.IS_GENERATED.

3) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLUMNS.GENERATION_EXPRESSION.

4) Without Feature T111, "Updatable joins, unions, and columns", conforming SQL language shall not refer-ence INFORMATION_SCHEMA.COLUMNS.IS_UPDATABLE.

5) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

6) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference any of the columns IS_SYSTEM_TIME_PERIOD_START, IS_SYSTEM_TIME_PERIOD_END, and SYS-TEM_TIME_PERIOD_TIMESTAMP_GENERATION.

5.22   CONSTRAINT_COLUMN_USAGE view

Function

Identify the columns used by referential constraints, unique constraints, check constraints, and assertions defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW CONSTRAINT_COLUMN_USAGE AS
    SELECT AC.TABLE_CATALOG, AC.TABLE_SCHEMA, AC.TABLE_NAME, AC.COLUMN_NAME,
           AC.CONSTRAINT_CATALOG, AC.CONSTRAINT_SCHEMA, AC.CONSTRAINT_NAME
    FROM ( ( SELECT CCU.TABLE_CATALOG, CCU.TABLE_SCHEMA, CCU.TABLE_NAME, CCU.COLUMN_NAME,
                    CCU.CONSTRAINT_CATALOG, CCU.CONSTRAINT_SCHEMA, CCU.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.CHECK_COLUMN_USAGE AS CCU )
        UNION
        ( SELECT KCU.TABLE_CATALOG, KCU.TABLE_SCHEMA, KCU.TABLE_NAME, KCU.COLUMN_NAME,
                 RC.CONSTRAINT_CATALOG, RC.CONSTRAINT_SCHEMA, RC.CONSTRAINT_NAME
          FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
            JOIN
              DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU
              ON
                ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
                  RC.UNIQUE_CONSTRAINT_NAME )
              = ( KCU.CONSTRAINT_CATALOG, KCU.CONSTRAINT_SCHEMA,
                  KCU.CONSTRAINT_NAME ) )
        UNION
        ( SELECT KCU.TABLE_CATALOG, KCU.TABLE_SCHEMA, KCU.TABLE_NAME, KCU.COLUMN_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU
            JOIN
              DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
            USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
          WHERE TC.CONSTRAINT_TYPE IN
                ( 'UNIQUE', 'PRIMARY KEY' ) ) ) AS AC (TABLE_CATALOG, TABLE_SCHEMA,
                                                       TABLE_NAME, COLUMN_NAME,
                                                       CONSTRAINT_CATALOG,
                                                       CONSTRAINT_SCHEMA,
                                                       CONSTRAINT_NAME)
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON
          ( ( AC.TABLE_CATALOG, AC.TABLE_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND AC.CONSTRAINT_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );
```

```
GRANT SELECT ON TABLE CONSTRAINT_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CONSTRAINT_COLUMN_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CONSTRAINT_COLUMN_USAGE.

5.23   CONSTAINT_PERIOD_USAGE view

Function

Identify the periods used by referential constraints, unique constraints, check constraints, and assertions defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW CONSTRAINT_PERIOD_USAGE AS
    SELECT AC.TABLE_CATALOG, AC.TABLE_SCHEMA, AC.TABLE_NAME,
           AC.PERIOD_NAME, AC.CONSTRAINT_CATALOG, AC.CONSTRAINT_SCHEMA,
           AC.CONSTRAINT_NAME
    FROM ( ( SELECT CPU.TABLE_CATALOG, CPU.TABLE_SCHEMA, CPU.TABLE_NAME, CPU.PERIOD_NAME,
                    CPU.CONSTRAINT_CATALOG, CPU.CONSTRAINT_SCHEMA, CPU.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.CHECK_PERIOD_USAGE AS CPU )
         UNION
           ( SELECT KPU.TABLE_CATALOG, KPU.TABLE_SCHEMA, KPU.TABLE_NAME, KPU.PERIOD_NAME,
                    RC.CONSTRAINT_CATALOG, RC.CONSTRAINT_SCHEMA, RC.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
               JOIN
                 DEFINITION_SCHEMA.KEY_PERIOD_USAGE AS KPU
                 ON
                   ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
                     RC.UNIQUE_CONSTRAINT_NAME )
                 = ( KPU.CONSTRAINT_CATALOG, KPU.CONSTRAINT_SCHEMA,
                     KPU.CONSTRAINT_NAME ) )
         UNION
           ( SELECT KPU.TABLE_CATALOG, KPU.TABLE_SCHEMA, KPU.TABLE_NAME, KPU.PERIOD_NAME,
                    CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.KEY_PERIOD_USAGE AS KPU
               JOIN
                 DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
               USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
             WHERE TC.CONSTRAINT_TYPE IN
                   ( 'UNIQUE', 'PRIMARY KEY' ) ) ) AS AC (TABLE_CATALOG, TABLE_SCHEMA,
                                                          TABLE_NAME, PERIOD_NAME,
                                                          CONSTRAINT_CATALOG,
                                                          CONSTRAINT_SCHEMA,
                                                          CONSTRAINT_NAME)
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON
           ( ( AC.TABLE_CATALOG, AC.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND AC.CONSTRAINT_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );
```

```
GRANT SELECT ON TABLE CONSTRAINT_PERIOD_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CONSTRAINT_PERIOD_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.CONSTRAINT_PERIOD_USAGE.

3) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_PERIOD_USAGE.

5.24 CONSTRAINT_TABLE_USAGE view

Function

Identify the tables that are used by referential constraints, unique constraints, check constraints, and assertions defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW CONSTRAINT_TABLE_USAGE AS
    SELECT AC.TABLE_CATALOG, AC.TABLE_SCHEMA, AC.TABLE_NAME,
           AC.CONSTRAINT_CATALOG, AC.CONSTRAINT_SCHEMA, AC.CONSTRAINT_NAME
    FROM ( ( SELECT CTU.TABLE_CATALOG, CTU.TABLE_SCHEMA, CTU.TABLE_NAME,
                    CTU.CONSTRAINT_CATALOG, CTU.CONSTRAINT_SCHEMA, CTU.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.CHECK_TABLE_USAGE AS CTU )
           UNION
           ( SELECT TC.TABLE_CATALOG, TC.TABLE_SCHEMA, TC.TABLE_NAME,
                    RC.CONSTRAINT_CATALOG, RC.CONSTRAINT_SCHEMA, RC.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
               JOIN
                 DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
               ON ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
                    RC.UNIQUE_CONSTRAINT_NAME )
                = ( TC.CONSTRAINT_CATALOG, TC.CONSTRAINT_SCHEMA,
                    TC.CONSTRAINT_NAME ) )
           UNION
           ( SELECT TC.TABLE_CATALOG, TC.TABLE_SCHEMA, TC.TABLE_NAME,
                    TC.CONSTRAINT_CATALOG, TC.CONSTRAINT_SCHEMA, TC.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
             WHERE TC.CONSTRAINT_TYPE IN
                 ( 'UNIQUE', 'PRIMARY KEY' ) ) ) AS AC (TABLE_CATALOG, TABLE_SCHEMA,
                                                        TABLE_NAME, CONSTRAINT_CATALOG,
                                                        CONSTRAINT_SCHEMA,
                                                        CONSTRAINT_NAME)
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( AC.TABLE_CATALOG, AC.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
            OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND CONSTRAINT_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE CONSTRAINT_TABLE_USAGE
```

```
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1)   Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-
     TION_SCHEMA.CONSTRAINT_TABLE_USAGE.

2)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
     TION_SCHEMA.CONSTRAINT_TABLE_USAGE.

5.25 DATA_TYPE_PRIVILEGES view

Function

Identify those schema objects whose included data type descriptors are accessible to a given user or role.

Definition

```
CREATE VIEW DATA_TYPE_PRIVILEGES
    ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
      OBJECT_TYPE, DTD_IDENTIFIER ) AS
      SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
             'USER-DEFINED TYPE', DTD_IDENTIFIER
      FROM ATTRIBUTES
    UNION
      SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
             'TABLE', DTD_IDENTIFIER
        FROM COLUMNS
    UNION
      SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
             'DOMAIN', DTD_IDENTIFIER
      FROM DOMAINS
    UNION
      SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
             'USER-DEFINED TYPE', DTD_IDENTIFIER
      FROM METHOD_SPECIFICATIONS
    UNION
      SELECT PARAMETER_UDT_CATALOG, PARAMETER_UDT_SCHEMA, PARAMETER_UDT_NAME,
             'USER-DEFINED TYPE', DTD_IDENTIFIER
      FROM METHOD_SPECIFICATION_PARAMETERS
    UNION
      SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
             'ROUTINE', DTD_IDENTIFIER
      FROM PARAMETERS
    UNION
      SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
             'ROUTINE', DTD_IDENTIFIER
      FROM ROUTINES
      WHERE DTD_IDENTIFIER IS NOT NULL
    UNION
      SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
             USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', SOURCE_DTD_IDENTIFIER
      FROM USER_DEFINED_TYPES
      WHERE SOURCE_DTD_IDENTIFIER IS NOT NULL
    UNION
      SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
             USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', REF_DTD_IDENTIFIER
      FROM USER_DEFINED_TYPES
      WHERE REF_DTD_IDENTIFIER IS NOT NULL;

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
```

```
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.DATA_TYPE_PRIVILEGES.

5.26 DIRECT_SUPERTABLES view

Function

Identify the direct supertables related to a table that are defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW DIRECT_SUPERTABLES AS
    SELECT DS.TABLE_CATALOG, DS.TABLE_SCHEMA, DS.TABLE_NAME, DS.SUPERTABLE_NAME
    FROM DEFINITION_SCHEMA.DIRECT_SUPERTABLES AS DS
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( DS.TABLE_CATALOG, DS.TABLE_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
        DS.TABLE_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE DIRECT_SUPERTABLES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S081, "Subtables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.DIRECT_SUPERTABLES.

5.27 DIRECT_SUPERTYPES view

Function

Identify the direct supertypes related to a user-defined type that are defined in this catalog and owned by a given user or role.

Definition

```
CREATE VIEW DIRECT_SUPERTYPES AS
    SELECT USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           USER_DEFINED_TYPE_NAME AS UDT_NAME,
           SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
    FROM DEFINITION_SCHEMA.DIRECT_SUPERTYPES
    WHERE ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IN
          ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
                   UDTP.USER_DEFINED_TYPE_NAME
            FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
              JOIN
                DEFINITION_SCHEMA.SCHEMATA AS S
              ON ( ( UDTP.USER_DEFINED_TYPE_CATALOG,
                                      UDTP.USER_DEFINED_TYPE_SCHEMA )
                 = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
            WHERE ( S.SCHEMA_OWNER = CURRENT_USER
                 OR
                   S.SCHEMA_OWNER IN
                   ( SELECT ROLE_NAME
                     FROM ENABLED_ROLES ) ) )
      AND
        USER_DEFINED_TYPE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE DIRECT_SUPERTYPES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S024, "Enhanced structured types", conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTYPES.

5.28 DOMAIN_CONSTRAINTS view

Function

Identify the domain constraints of domains in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW DOMAIN_CONSTRAINTS AS
    SELECT DISTINCT
      CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
      DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
      IS_DEFERRABLE, INITIALLY_DEFERRED
    FROM DEFINITION_SCHEMA.DOMAIN_CONSTRAINTS
    WHERE ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN' ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
          FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
          WHERE ( UP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
              OR UP.GRANTEE IN
                 ( SELECT ROLE_NAME
                   FROM ENABLED_ROLES ) ) )
    AND CONSTRAINT_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE DOMAIN_CONSTRAINTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.DOMAIN_CONSTRAINTS.

5.29 DOMAINS view

This Subclause is modified by Subclause 21.5, "DOMAINS view", in ISO/IEC 9075-14.

Function

Identify the domains defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW DOMAINS AS
    SELECT DISTINCT
      D.DOMAIN_CATALOG, D.DOMAIN_SCHEMA, D.DOMAIN_NAME,
      DTD.DATA_TYPE, DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
      DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
      DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
      DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
      DTD.DATETIME_PRECISION, DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
      D.DOMAIN_DEFAULT, DTD.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER,
      DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
      DTD.DECLARED_NUMERIC_SCALE
    FROM DEFINITION_SCHEMA.DOMAINS AS D
      JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
      ON ( ( D.DOMAIN_CATALOG, D.DOMAIN_SCHEMA, D.DOMAIN_NAME,
             'DOMAIN', D.DTD_IDENTIFIER )
        = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA, DTD.OBJECT_NAME,
            DTD.OBJECT_TYPE, DTD.DTD_IDENTIFIER ) )
    WHERE ( ( D.DOMAIN_CATALOG, D.DOMAIN_SCHEMA, D.DOMAIN_NAME, 'DOMAIN' ) IN
            ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME, UP.OBJECT_TYPE
              FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
              WHERE ( UP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                      UP.GRANTEE IN
                      ( SELECT ER.ROLE_NAME
                        FROM ENABLED_ROLES AS ER ) ) )
          OR
            ( D.DOMAIN_CATALOG, D.DOMAIN_SCHEMA, D.DOMAIN_NAME ) IN
            ( SELECT C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME
              FROM COLUMNS AS C ) )
      AND
        D.DOMAIN_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE DOMAINS
```

```
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.DOMAINS.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.DOMAINS.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.30 ELEMENT_TYPES view

This Subclause is modified by Subclause 21.6, "ELEMENT_TYPES view", in ISO/IEC 9075-14.

Function

Identify the collection element types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ELEMENT_TYPES AS
    SELECT DISTINCT
        OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, ET.COLLECTION_TYPE_IDENTIFIER, DTD.DATA_TYPE,
        DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
        DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
        DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
        DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
        DTD.DATETIME_PRECISION, DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
        DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
        DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
        DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
        DTD.SCOPE_CATALOG, DTD.SCOPE_SCHEMA, DTD.SCOPE_NAME,
        DTD.MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
        DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
        DTD.DECLARED_NUMERIC_SCALE
    FROM DEFINITION_SCHEMA.ELEMENT_TYPES AS ET
      JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
      USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, DTD_IDENTIFIER )
    WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, ET.ROOT_DTD_IDENTIFIER ) IN
          ( SELECT DTP.OBJECT_CATALOG, DTP.OBJECT_SCHEMA, DTP.OBJECT_NAME,
                   DTP.OBJECT_TYPE, DTP.DTD_IDENTIFIER
            FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES AS DTP );

GRANT SELECT ON TABLE ELEMENT_TYPES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S091, "Basic array support", or Feature S271, "Basic multiset support", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ELEMENT_TYPES.

3)   Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.31 ENABLED_ROLES view

Function

Identify the enabled roles for the current SQL-session.

Definition

```
CREATE RECURSIVE VIEW ENABLED_ROLES ( ROLE_NAME ) AS
    VALUES ( CURRENT_ROLE )
  UNION
    SELECT RAD.ROLE_NAME
    FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_DESCRIPTORS RAD
      JOIN
        ENABLED_ROLES R
        ON RAD.GRANTEE = R.ROLE_NAME;

GRANT SELECT ON TABLE ENABLED_ROLES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.ENABLED_ROLES.

5.32   FIELDS view

This Subclause is modified by Subclause 21.7, "FIELDS view", in ISO/IEC 9075-14.

Function

Identify the field types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW FIELDS AS
    SELECT DISTINCT
        OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, F.ROW_IDENTIFIER, F.FIELD_NAME,
        F.ORDINAL_POSITION, DTD.DATA_TYPE,
        DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
        DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
        DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
        DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
        DTD.DATETIME_PRECISION, DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
        DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
        DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
        DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
        DTD.SCOPE_CATALOG, DTD.SCOPE_SCHEMA, DTD.SCOPE_NAME,
        DTD.MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
        DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
        DTD.DECLARED_NUMERIC_SCALE
    FROM DEFINITION_SCHEMA.FIELDS AS F
      JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
      USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, DTD_IDENTIFIER )
    WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, F.ROOT_DTD_IDENTIFIER ) IN
          ( SELECT DTP.OBJECT_CATALOG, DTP.OBJECT_SCHEMA, DTP.OBJECT_NAME,
                DTP.OBJECT_TYPE, DTP.DTD_IDENTIFIER
            FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES AS DTP );

GRANT SELECT ON TABLE FIELDS
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1)   Without Feature T051, "Row types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.FIELDS.

2)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.FIELDS.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.33 KEY_COLUMN_USAGE view

Function

Identify the columns defined in this catalog that are constrained as keys and that are accessible by a given user or role.

Definition

```
CREATE VIEW KEY_COLUMN_USAGE AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
           KCU1.TABLE_CATALOG, KCU1.TABLE_SCHEMA, KCU1.TABLE_NAME,
           KCU1.COLUMN_NAME, KCU1.ORDINAL_POSITION, KCU1.POSITION_IN_UNIQUE_CONSTRAINT
    FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU1
      JOIN
        INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC
      USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
    WHERE ( ( SELECT MAX ( KCU3.ORDINAL_POSITION )
              FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU3
              WHERE KCU3.CONSTRAINT_CATALOG = CONSTRAINT_CATALOG
                AND
                  KCU3.CONSTRAINT_SCHEMA = CONSTRAINT_SCHEMA
                AND
                  KCU3.CONSTRAINT_NAME = CONSTRAINT_NAME
            )
          = ( SELECT COUNT (*)
              FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KCU2
              WHERE ( KCU2.TABLE_CATALOG, KCU2.TABLE_SCHEMA,
                    KCU2.TABLE_NAME, KCU2.COLUMN_NAME )
                IN ( SELECT CP2.TABLE_CATALOG, CP2.TABLE_SCHEMA,
                          CP2.TABLE_NAME, CP2.COLUMN_NAME
                     FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP2
                     WHERE ( CP2.GRANTEE IN
                          ( 'PUBLIC', CURRENT_USER )
                       OR
                          CP2.GRANTEE IN
                          ( SELECT ROLE_NAME
                            FROM ENABLED_ROLES )
                     )
                   )
            AND
              KCU2.CONSTRAINT_CATALOG = CONSTRAINT_CATALOG
            AND
              KCU2.CONSTRAINT_SCHEMA = CONSTRAINT_SCHEMA
            AND
              KCU2.CONSTRAINT_NAME = CONSTRAINT_NAME
              )
          )
        AND
          CONSTRAINT_CATALOG
          = ( SELECT CATALOG_NAME
              FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

```
GRANT SELECT ON TABLE KEY_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.KEY_COLUMN_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.KEY_COLUMN_USAGE.

5.34   KEY_PERIOD_USAGE view

Function

Identify the periods defined in this catalog that participate in the definition of unique, primary, and foreign keys and that are accessible by a given user or role.

Definition

```
CREATE VIEW KEY_PERIOD_USAGE AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
           KPU1.TABLE_CATALOG, KPU1.TABLE_SCHEMA, KPU1.TABLE_NAME,
           KPU1.PERIOD_NAME
    FROM DEFINITION_SCHEMA.KEY_PERIOD_USAGE AS KPU1
      JOIN
        INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC
      USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
    WHERE CONSTRAINT_CATALOG
          = ( SELECT ISCN.CATALOG_NAME
                FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE KEY_PERIOD_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.KEY_PERIOD_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.KEY_PERIOD_USAGE.

3) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_PERIOD_USAGE.

5.35 METHOD_SPECIFICATION_PARAMETERS view

This Subclause is modified by Subclause 21.8, "METHOD_SPECIFICATION_PARAMETERS view", in ISO/IEC 9075-14.

Function

Identify the SQL parameters of method specifications described in the METHOD_SPECIFICATIONS view that are accessible to a given user or role.

Definition

```
CREATE VIEW METHOD_SPECIFICATION_PARAMETERS AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           P.ORDINAL_POSITION, P.PARAMETER_MODE, P.IS_RESULT,
           P.AS_LOCATOR, P.PARAMETER_NAME,
           P.FROM_SQL_SPECIFIC_CATALOG, P.FROM_SQL_SPECIFIC_SCHEMA,
           P.FROM_SQL_SPECIFIC_NAME, D.DATA_TYPE,
           D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
           D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
           D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
           D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
           D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
           D.USER_DEFINED_TYPE_CATALOG AS PARAMETER_UDT_CATALOG,
           D.USER_DEFINED_TYPE_SCHEMA AS PARAMETER_UDT_SCHEMA,
           D.USER_DEFINED_TYPE_NAME AS PARAMETER_UDT_NAME,
           D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
           D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER,
           D.DECLARED_DATA_TYPE, D.DECLARED_NUMERIC_PRECISION,
           D.DECLARED_NUMERIC_SCALE
    FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS AS P
      JOIN
           DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
        ON
           ( P.SPECIFIC_CATALOG, P.SPECIFIC_SCHEMA, P.SPECIFIC_NAME,
             'USER-DEFINED TYPE', P.DTD_IDENTIFIER )
         = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
             D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
      JOIN
         DEFINITION_SCHEMA.METHOD_SPECIFICATIONS AS M
      USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
            M.USER_DEFINED_TYPE_NAME ) IN
        ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
                 UDTP.USER_DEFINED_TYPE_NAME
          FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
          WHERE ( UDTP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UDTP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
             AND
```

```
        M.USER_DEFINED_TYPE_CATALOG
      = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE METHOD_SPECIFICATION_PARAMETERS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.36 METHOD_SPECIFICATIONS view

This Subclause is modified by Subclause 14.3, "METHOD_SPECIFICATIONS view", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 21.9, "METHOD_SPECIFICATIONS view", in ISO/IEC 9075-14.

Function

Identify the SQL-invoked methods in the catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW METHOD_SPECIFICATIONS AS
    SELECT M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
            M.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
            M.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
            M.USER_DEFINED_TYPE_NAME AS UDT_NAME,
            M.METHOD_NAME, IS_STATIC, IS_OVERRIDING, IS_CONSTRUCTOR,
            D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
            D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
            D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
            D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
            D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
            D.USER_DEFINED_TYPE_CATALOG AS RETURN_UDT_CATALOG,
            D.USER_DEFINED_TYPE_SCHEMA AS RETURN_UDT_SCHEMA,
            D.USER_DEFINED_TYPE_NAME AS RETURN_UDT_NAME,
            D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
            D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, M.METHOD_LANGUAGE,
            M.PARAMETER_STYLE, M.IS_DETERMINISTIC, M.SQL_DATA_ACCESS,
            M.IS_NULL_CALL,
            M.TO_SQL_SPECIFIC_CATALOG, M.TO_SQL_SPECIFIC_SCHEMA,
            M.TO_SQL_SPECIFIC_NAME,
            M.CREATED,
            DT.DATA_TYPE AS RESULT_CAST_FROM_DATA_TYPE,
            RESULT_CAST_AS_LOCATOR,
            DT.CHARACTER_MAXIMUM_LENGTH AS RESULT_CAST_CHAR_MAX_LENGTH,
            DT.CHARACTER_OCTET_LENGTH AS RESULT_CAST_CHAR_OCTET_LENGTH,
            DT.CHARACTER_SET_CATALOG AS RESULT_CAST_CHAR_SET_CATALOG,
            DT.CHARACTER_SET_SCHEMA AS RESULT_CAST_CHAR_SET_SCHEMA,
            DT.CHARACTER_SET_NAME AS RESULT_CAST_CHAR_SET_NAME,
            DT.COLLATION_CATALOG AS RESULT_CAST_COLLATION_CATALOG,
            DT.COLLATION_SCHEMA AS RESULT_CAST_COLLATION_SCHEMA,
            DT.COLLATION_NAME AS RESULT_CAST_COLLATION_NAME,
            DT.NUMERIC_PRECISION AS RESULT_CAST_NUMERIC_PRECISION,
            DT.NUMERIC_PRECISION_RADIX AS RESULT_CAST_NUMERIC_RADIX,
            DT.NUMERIC_SCALE AS RESULT_CAST_NUMERIC_SCALE,
            DT.DATETIME_PRECISION AS RESULT_CAST_DATETIME_PRECISION,
            DT.INTERVAL_TYPE AS RESULT_CAST_INTERVAL_TYPE,
            DT.INTERVAL_PRECISION AS RESULT_CAST_INTERVAL_PRECISION,
            DT.USER_DEFINED_TYPE_CATALOG AS RESULT_CAST_TYPE_UDT_CATALOG,
            DT.USER_DEFINED_TYPE_SCHEMA AS RESULT_CAST_TYPE_UDT_SCHEMA,
            DT.USER_DEFINED_TYPE_NAME AS RESULT_CAST_TYPE_UDT_NAME,
            DT.SCOPE_CATALOG AS RESULT_CAST_SCOPE_CATALOG,
            DT.SCOPE_SCHEMA AS RESULT_CAST_SCOPE_SCHEMA,
```

```
            DT.SCOPE_NAME AS RESULT_CAST_SCOPE_NAME,
            DT.MAXIMUM_CARDINALITY AS RESULT_CAST_MAX_CARDINALITY,
            DT.DTD_IDENTIFIER AS RESULT_CAST_DTD_IDENTIFIER,
            D.DECLARED_DATA_TYPE, D.DECLARED_NUMERIC_PRECISION,
            D.DECLARED_NUMERIC_SCALE
    FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATIONS AS M
      JOIN
            DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
        ON ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
             M.USER_DEFINED_TYPE_NAME,
             'USER-DEFINED TYPE', M.DTD_IDENTIFIER )
       = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA,
             D.OBJECT_NAME,
             D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
      LEFT JOIN
            DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DT
        ON ( M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
             'USER-DEFINED TYPE', RESULT_CAST_FROM_DTD_IDENTIFIER )
       = ( DT.OBJECT_CATALOG, DT.OBJECT_SCHEMA, DT.OBJECT_NAME,
             DT.OBJECT_TYPE, DT.DTD_IDENTIFIER )
    WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
            M.USER_DEFINED_TYPE_NAME ) IN
        ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
                 UDTP.USER_DEFINED_TYPE_NAME
          FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
          WHERE ( UDTP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  UDTP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
              AND
                M.USER_DEFINED_TYPE_CATALOG
              = ( SELECT CATALOG_NAME
                  FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE METHOD_SPECIFICATIONS
    TO PUBLIC WITH GRANT OPTION;
```

NOTE 5 — The METHOD_SPECIFICATIONS view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with "RESULT_CAST_" describes the data type specified in the <result cast>, if any, contained in the <method specification>, the other set of columns describes the data type specified in the <returns data type> contained in the <method specification>.

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPECIFICATIONS.

2) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPECIFICATIONS.

3) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.CREATED.

4)   Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.37 PARAMETERS view

This Subclause is modified by Subclause 19.5, "PARAMETERS view", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 21.10, "PARAMETERS view", in ISO/IEC 9075-14.

Function

Identify the SQL parameters of SQL-invoked routines defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW PARAMETERS AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           P.ORDINAL_POSITION, P.PARAMETER_MODE, P.IS_RESULT, P.AS_LOCATOR,
           P.PARAMETER_NAME, P.FROM_SQL_SPECIFIC_CATALOG,
           P.FROM_SQL_SPECIFIC_SCHEMA, P.FROM_SQL_SPECIFIC_NAME,
           P.TO_SQL_SPECIFIC_CATALOG, P.TO_SQL_SPECIFIC_SCHEMA, P.TO_SQL_SPECIFIC_NAME,
           DTD.DATA_TYPE, DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
           DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
           DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
           DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
           DTD.DATETIME_PRECISION,DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
           DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
           DTD.SCOPE_CATALOG, DTD.SCOPE_SCHEMA, DTD.SCOPE_NAME,
           DTD.MAXIMUM_CARDINALITY, DTD.DTD_IDENTIFIER,
           DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
           DTD.DECLARED_NUMERIC_SCALE,
           CASE
             WHEN EXISTS
               ( SELECT *
                 FROM DEFINITION_SCHEMA.SCHEMATA AS S
                 WHERE ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                   AND
                     ( S.SCHEMA_OWNER = CURRENT_USER
                     OR
                       S.SCHEMA_OWNER IN
                       ( SELECT ER.ROLE_NAME
                         FROM ENABLED_ROLES AS ER ) ) )
             THEN P.PARAMETER_DEFAULT
             ELSE NULL
           END AS PARAMETER_DEFAULT,
           DTD.TABLE_SEMANTICS, DTD.IS_PRUNABLE, DTD.HAS_PASS_THROUGH_COLUMNS
    FROM ( DEFINITION_SCHEMA.PARAMETERS AS P
         LEFT JOIN
           DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
           ON ( P.SPECIFIC_CATALOG, P.SPECIFIC_SCHEMA, P.SPECIFIC_NAME,
                'ROUTINE', P.DTD_IDENTIFIER )
            = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA, DTD.OBJECT_NAME,
                DTD.OBJECT_TYPE, DTD.DTD_IDENTIFIER ) )
```

```
        JOIN
            DEFINITION_SCHEMA.ROUTINES AS R
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    WHERE ( ( ( R.MODULE_CATALOG, R.MODULE_SCHEMA, R.MODULE_NAME ) IS NULL
            AND
                ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
                ( SELECT RP.SPECIFIC_CATALOG, RP.SPECIFIC_SCHEMA, RP.SPECIFIC_NAME
                    FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES AS RP
                    WHERE ( RP.GRANTEE IN
                            ( 'PUBLIC', CURRENT_USER )
                        OR
                            RP.GRANTEE IN
                            ( SELECT ER.ROLE_NAME
                                FROM ENABLED_ROLES AS ER ) ) ) ) ) )
        AND SPECIFIC_CATALOG
            = ( SELECT ISCN.CATALOG_NAME
                FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE PARAMETERS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.PARAMETERS.

2) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

3) Without Feature T522, "Default values for IN parameters of SQL-invoked procedures", Feature T523, "Default values for INOUT parameters of SQL-invoked procedures", or Feature T525, "Default values for parameters of SQL-invoked functions", conforming SQL language shall not reference INFORMA-TION_SCHEMA.PARAMETERS.PARAMETER_DEFAULT.

4) Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference INFORMATION_SCHEMA.PARAMETERS.TABLE_SEMANTICS, INFORMA-TION_SCHEMA.PARAMETERS.IS_PRUNABLE, or INFORMATION_SCHEMA.PARAMETERS.

5) Without Feature B204, "PRUNE WHEN EMPTY", conforming SQL language shall not reference INFORMATION_SCHEMA.PARAMETERS.IS_PRUNABLE.

6) Without Feature B205, "Pass-through columns", conforming SQL language shall not reference INFOR-MATION_SCHEMA.PARAMETERS.HAS_PASS_THROUGH_COLUMNS.

5.38 PERIODS view

Function

Identify the periods of tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW PERIODS AS
    SELECT P.TABLE_CATALOG, P.TABLE_SCHEMA, P.TABLE_NAME, P.PERIOD_NAME,
          CASE
            WHEN EXISTS ( SELECT *
                          FROM DEFINITION_SCHEMA.SCHEMATA AS S
                          WHERE ( P.TABLE_CATALOG, P.TABLE_SCHEMA )
                              = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                            AND
                                ( S. SCHEMA_OWNER = CURRENT_USER
                                  OR
                                  S. SCHEMA_OWNER IN
                                  ( SELECT ER.ROLE_NAME
                                    FROM ENABLED_ROLES AS ER ) ) )
            THEN P.START_COLUMN_NAME
            ELSE NULL
          END AS START_COLUMN_NAME,
        CASE
          WHEN EXISTS ( SELECT *
                        FROM DEFINITION_SCHEMA.SCHEMATA AS S
                        WHERE ( P.TABLE_CATALOG, P.TABLE_SCHEMA )
                            = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                          AND
                              ( S. SCHEMA_OWNER = CURRENT_USER
                                OR
                                S. SCHEMA_OWNER IN
                                ( SELECT ER.ROLE_NAME
                                  FROM ENABLED_ROLES AS ER ) ) )
            THEN P.END_COLUMN_NAME
            ELSE NULL
          END AS END_COLUMN_NAME
      FROM DEFINITION_SCHEMA.PERIODS AS P
    WHERE ( P.TABLE_CATALOG, P.TABLE_SCHEMA, P.TABLE_NAME ) IN
          ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
            FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
            WHERE ( TP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    TP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) )
            UNION
            SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
            FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
            WHERE ( CP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
```

```
                OR
                  CP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
    AND P.TABLE_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN);

GRANT SELECT ON TABLE PERIODS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference INFORMATION_SCHEMA.PERIODS.

2) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.PERIODS.

5.39 PRIVATE_PARAMETERS view

Function

Identify the private parameters of polymorphic table functions defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW PRIVATE_PARAMETERS AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           P.ORDINAL_POSITION, P.PARAMETER_NAME, P.FROM_SQL_SPECIFIC_CATALOG,
           P.FROM_SQL_SPECIFIC_SCHEMA, P.FROM_SQL_SPECIFIC_NAME,
           P.TO_SQL_SPECIFIC_CATALOG, P.TO_SQL_SPECIFIC_SCHEMA, P.TO_SQL_SPECIFIC_NAME,
           DTD.DATA_TYPE, DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
           DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
           DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
           DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
           DTD.DATETIME_PRECISION,DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
           DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
           DTD.SCOPE_CATALOG, DTD.SCOPE_SCHEMA, DTD.SCOPE_NAME,
           DTD.MAXIMUM_CARDINALITY, DTD.DTD_IDENTIFIER,
           DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
           DTD.DECLARED_NUMERIC_SCALE, P.PARAMETER_DEFAULT
           DTD.TABLE_SEMANTICS, DTD.IS_PRUNABLE, DTD.HAS_PASS_THROUGH_COLUMNS
    FROM ( DEFINITION_SCHEMA.PRIVATE_PARAMETERS AS P
        LEFT JOIN
           DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
           ON ( P.SPECIFIC_CATALOG, P.SPECIFIC_SCHEMA, P.SPECIFIC_NAME,
                'ROUTINE', P.DTD_IDENTIFIER )
            = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA, DTD.OBJECT_NAME,
                DTD.OBJECT_TYPE, DTD.DTD_IDENTIFIER ) )
      JOIN
         DEFINITION_SCHEMA.ROUTINES AS R
      USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    WHERE EXISTS ( SELECT *
                   FROM DEFINITION_SCHEMA.SCHEMATA AS S
                   WHERE ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
                       = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                       AND
                         ( S.SCHEMA_OWNER = CURRENT_USER
                     OR
                         S.SCHEMA_OWNER IN
                           ( SELECT ER.ROLE_NAME
                             FROM ENABLED_ROLES AS ER ) ) )
      AND SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE PRIVATE_PARAMETERS
```

```
TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
     TION_SCHEMA.PRIVATE_PARAMETERS.

2)   Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference
     INFORMATION_SCHEMA.PRIVATE_PARAMETERS.

3)   Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any
     of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and
     DECLARED_NUMERIC_SCALE.

4)   Without Feature T522, "Default values for IN parameters of SQL-invoked procedures", Feature T523,
     "Default values for INOUT parameters of SQL-invoked procedures", or Feature T525, "Default values
     for parameters of SQL-invoked functions", conforming SQL language shall not reference INFORMA-
     TION_SCHEMA.PRIVATE_PARAMETERS.PARAMETER_DEFAULT.

5)   Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference
     INFORMATION_SCHEMA.PRIVATE_PARAMETERS.TABLE_SEMANTICS, INFORMA-
     TION_SCHEMA.PRIVATE_PARAMETERS.IS_PRUNABLE, or INFORMATION_SCHEMA.PRI-
     VATE_PARAMETERS.

6)   Without Feature B204, "PRUNE WHEN EMPTY", conforming SQL language shall not reference
     INFORMATION_SCHEMA.PRIVATE_PARAMETERS.IS_PRUNABLE.

7)   Without Feature B205, "Pass-through columns", conforming SQL language shall not reference INFOR-
     MATION_SCHEMA.PRIVATE_PARAMETERS.HAS_PASS_THROUGH_COLUMNS.

5.40   REFERENCED_TYPES view

Function

Identify the referenced types of reference types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW REFERENCED_TYPES AS
    SELECT DISTINCT
        OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, RT.REFERENCE_TYPE_IDENTIFIER, DTD.DATA_TYPE,
        DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
        DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
        DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
        DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
        DTD.DATETIME_PRECISION, DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
        DTD.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
        DTD.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
        DTD.USER_DEFINED_TYPE_NAME AS UDT_NAME,
        DTD.SCOPE_CATALOG, DTD.SCOPE_SCHEMA, DTD.SCOPE_NAME,
        DTD.MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
        DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
        DTD.DECLARED_NUMERIC_SCALE
    FROM ( DEFINITION_SCHEMA.REFERENCED_TYPES AS RT
        JOIN
          DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
        USING ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, DTD_IDENTIFIER ) )
    WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, RT.ROOT_DTD_IDENTIFIER ) IN
        ( SELECT DTP.OBJECT_CATALOG, DTP.OBJECT_SCHEMA, DTP.OBJECT_NAME,
              DTP.OBJECT_TYPE, DTP.DTD_IDENTIFIER
          FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES AS DTP );

GRANT SELECT ON TABLE REFERENCED_TYPES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S041, "Basic reference types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.REFERENCED_TYPES.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.REFERENCED_TYPES.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.41   REFERENTIAL_CONSTRAINTS view

Function

Identify the referential constraints defined on tables in this catalog that are accssible to a given user or role.

Definition

```
CREATE VIEW REFERENTIAL_CONSTRAINTS AS
    SELECT DISTINCT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
           TC2.CONSTRAINT_CATALOG AS UNIQUE_CONSTRAINT_CATALOG,
           TC2.CONSTRAINT_SCHEMA AS UNIQUE_CONSTRAINT_SCHEMA,
           TC2.CONSTRAINT_NAME AS UNIQUE_CONSTRAINT_NAME,
           RC.MATCH_OPTION, RC.UPDATE_RULE, RC.DELETE_RULE
    FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
      JOIN
        INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC1
      USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
      LEFT JOIN
        INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS TC2
        ON ( ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
               RC.UNIQUE_CONSTRAINT_NAME )
           = ( TC2.CONSTRAINT_CATALOG, TC2.CONSTRAINT_SCHEMA, TC2.CONSTRAINT_NAME ) )
    WHERE CONSTRAINT_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE REFERENTIAL_CONSTRAINTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
     TION_SCHEMA.REFERENTIAL_CONSTRAINTS.

5.42   ROLE_COLUMN_GRANTS view

Function

Identifies the privileges on columns defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_COLUMN_GRANTS AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           COLUMN_NAME, PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
    WHERE ( GRANTEE IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES )
           OR
             GRANTOR IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES ) )
      AND TABLE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_COLUMN_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", and Feature T331, "Basic roles", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_COLUMN_GRANTS.

5.43 ROLE_ROUTINE_GRANTS view

Function

Identify the privileges on SQL-invoked routines defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_ROUTINE_GRANTS AS
    SELECT RP.GRANTOR, RP.GRANTEE,
            SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            R.ROUTINE_CATALOG, R.ROUTINE_SCHEMA, R.ROUTINE_NAME,
            RP.PRIVILEGE_TYPE, RP.IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES AS RP
      JOIN
        DEFINITION_SCHEMA.ROUTINES AS R
      USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    WHERE ( RP.GRANTEE IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER )
          OR
            RP.GRANTOR IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROLE_ROUTINE_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_ROUTINE_GRANTS.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_ROUTINE_GRANTS.

3) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_ROUTINE_GRANTS.

5.44 ROLE_TABLE_GRANTS view

Function

Identifies the privileges on tables defined in this catalog that are available to or granted by the currently applicable roles.

Definition

```
CREATE VIEW ROLE_TABLE_GRANTS AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           PRIVILEGE_TYPE, IS_GRANTABLE, WITH_HIERARCHY
    FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
    WHERE ( GRANTEE IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES )
          OR
            GRANTOR IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
      AND TABLE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_TABLE_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_TABLE_GRANTS.

2) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_TABLE_GRANTS.

5.45 ROLE_TABLE_METHOD_GRANTS view

Function

Identify the privileges on methods of tables of structured types defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_TABLE_METHOD_GRANTS AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
           TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
           SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.TABLE_METHOD_PRIVILEGES
    WHERE ( GRANTEE IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES )
          OR
             GRANTOR IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES ) )
      AND
          TABLE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_TABLE_METHOD_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.

2) Without Feature S024, "Enhanced structured types", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.

3) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_TABLE_METHOD_GRANTS.

5.46   ROLE_USAGE_GRANTS view

Function

Identify the USAGE privileges on objects defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_USAGE_GRANTS AS
    SELECT GRANTOR, GRANTEE,
           OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE,
           'USAGE' AS PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
    WHERE ( GRANTEE IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES )
          OR
            GRANTOR IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
      AND
          OBJECT_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_USAGE_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_USAGE_GRANTS.

2) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_USAGE_GRANTS.

5.47 ROLE_UDT_GRANTS view

Function

Identify the privileges on user-defined types defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_UDT_GRANTS AS
    SELECT GRANTOR, GRANTEE,
           USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           USER_DEFINED_TYPE_NAME AS UDT_NAME,
           PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
    WHERE ( GRANTEE IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES )
           OR
             GRANTOR IN
             ( SELECT ROLE_NAME
               FROM ENABLED_ROLES ) )
      AND
           USER_DEFINED_TYPE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_UDT_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_UDT_GRANTS.

2) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROLE_UDT_GRANTS.

5.48 ROUTINE_COLUMN_USAGE view

Function

Identify the columns owned by a given user or role on which SQL routines defined in this catalog are dependent.

Definition

```
CREATE VIEW ROUTINE_COLUMN_USAGE AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, R.ROUTINE_CATALOG,
           R.ROUTINE_SCHEMA, R.ROUTINE_NAME, RCU.TABLE_CATALOG, RCU.TABLE_SCHEMA,
           RCU.TABLE_NAME, RCU.COLUMN_NAME
    FROM ( DEFINITION_SCHEMA.ROUTINE_COLUMN_USAGE AS RCU
        JOIN
           DEFINITION_SCHEMA.ROUTINES AS R
           USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) )
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( RCU.TABLE_CATALOG, RCU.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
             ( SELECT ER.ROLE_NAME
               FROM ENABLED_ROLES AS ER ) )
      AND
          R.ROUTINE_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROUTINE_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_COLUMN_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_COLUMN_USAGE.

5.49 ROUTINE_PERIOD_USAGE view

Function

Identify the periods of tables owned by a given user or role on which SQL routines defined in this catalog are dependent.

Definition

```
CREATE VIEW ROUTINE_PERIOD_USAGE AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           R.ROUTINE_CATALOG, R.ROUTINE_SCHEMA, R.ROUTINE_NAME,
           RPU.TABLE_CATALOG, RPU.TABLE_SCHEMA, RPU.TABLE_NAME,
           RPU.PERIOD_NAME
    FROM ( DEFINITION_SCHEMA.ROUTINE_PERIOD_USAGE AS RPU
        JOIN
            DEFINITION_SCHEMA.ROUTINES AS R
            USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) )
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( RPU.TABLE_CATALOG, RPU.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
             ( SELECT ER.ROLE_NAME
                 FROM ENABLED_ROLES AS ER ) )
      AND
          R.ROUTINE_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROUTINE_PERIOD_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_PERIOD_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_PERIOD_USAGE.

3) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_PERIOD_USAGE.

4) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_PERIOD_USAGE.

5.50 ROUTINE_PRIVILEGES view

Function

Identify the privileges on SQL-invoked routines defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW ROUTINE_PRIVILEGES AS
    SELECT RP.GRANTOR, RP.GRANTEE, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           R.ROUTINE_CATALOG, R.ROUTINE_SCHEMA, R.ROUTINE_NAME,
           RP.PRIVILEGE_TYPE, RP.IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES AS RP
      JOIN
        DEFINITION_SCHEMA.ROUTINES AS R
        USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    WHERE ( RP.GRANTEE IN
            ( 'PUBLIC', CURRENT_USER )
          OR
            RP.GRANTEE IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER )
          OR
            RP.GRANTOR
          = CURRENT_USER
          OR
            RP.GRANTOR IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER )
        AND
          R.ROUTINE_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN ) );

GRANT SELECT ON TABLE ROUTINE_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_PRIVILEGES.

5.51 ROUTINE_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which an SQL routine defined in this catalog is dependent.

Definition

```
CREATE VIEW ROUTINE_ROUTINE_USAGE AS
    SELECT RRU.SPECIFIC_CATALOG, RRU.SPECIFIC_SCHEMA, RRU.SPECIFIC_NAME,
           RRU.ROUTINE_CATALOG, RRU.ROUTINE_SCHEMA, RRU.ROUTINE_NAME
    FROM DEFINITION_SCHEMA.ROUTINE_ROUTINE_USAGE AS RRU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( RRU.ROUTINE_CATALOG, RRU.ROUTINE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          RRU.SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROUTINE_ROUTINE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_ROUTINE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_ROUTINE_USAGE.

5.52   ROUTINE_SEQUENCE_USAGE view

Function

Identify each external sequence generator owned by a given user or role on which some SQL routine defined in this catalog is dependent.

Definition

```
CREATE VIEW ROUTINE_SEQUENCE_USAGE AS
    SELECT RSU.SPECIFIC_CATALOG, RSU.SPECIFIC_SCHEMA, RSU.SPECIFIC_NAME,
           RSU.SEQUENCE_CATALOG, RSU.SEQUENCE_SCHEMA, RSU.SEQUENCE_NAME
    FROM DEFINITION_SCHEMA.ROUTINE_SEQUENCE_USAGE AS RSU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( RSU.SPECIFIC_CATALOG, RSU.SPECIFIC_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
      AND
        RSU.SEQUENCE_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROUTINE_SEQUENCE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_SEQUENCE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_SEQUENCE_USAGE.

3) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE.

5.53 ROUTINE_TABLE_USAGE view

Function

Identify the tables owned by a given user or role on which SQL routines defined in this catalog are dependent.

Definition

```
CREATE VIEW ROUTINE_TABLE_USAGE AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           R.ROUTINE_CATALOG, R.ROUTINE_SCHEMA, R.ROUTINE_NAME,
           RTU.TABLE_CATALOG, RTU.TABLE_SCHEMA, RTU.TABLE_NAME
    FROM ( DEFINITION_SCHEMA.ROUTINE_TABLE_USAGE AS RTU
         JOIN
           DEFINITION_SCHEMA.ROUTINES AS R
           USING ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) )
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( RTU.TABLE_CATALOG, RTU.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
                FROM ENABLED_ROLES AS ER ) )
      AND
          SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
              FROM INFORMATION_SCHEMA.CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE ROUTINE_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_TABLE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ROUTINE_TABLE_USAGE.

5.54 ROUTINES view

This Subclause is modified by Subclause 19.7, "ROUTINES view", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 21.11, "ROUTINES view", in ISO/IEC 9075-14.

Function

Identify the SQL-invoked routines in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW ROUTINES AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_TYPE,
           MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           R.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           R.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           R.USER_DEFINED_TYPE_NAME AS UDT_NAME,
           D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
           D.CHARACTER_SET_CATALOG, D.CHARACTER_SET_SCHEMA, D.CHARACTER_SET_NAME,
           D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
           D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
           D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
           D.USER_DEFINED_TYPE_CATALOG AS TYPE_UDT_CATALOG,
           D.USER_DEFINED_TYPE_SCHEMA AS TYPE_UDT_SCHEMA,
           D.USER_DEFINED_TYPE_NAME AS TYPE_UDT_NAME,
           D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
           D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, ROUTINE_BODY,
           CASE
             WHEN EXISTS
               ( SELECT *
                 FROM DEFINITION_SCHEMA.SCHEMATA AS S
                 WHERE ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                   AND
                     ( SCHEMA_OWNER = CURRENT_USER
                     OR
                       SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN ROUTINE_DEFINITION
             ELSE NULL
           END AS ROUTINE_DEFINITION,
           EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE,
           IS_DETERMINISTIC, SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
           SCHEMA_LEVEL_ROUTINE, MAX_DYNAMIC_RESULT_SETS,
           IS_USER_DEFINED_CAST, IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE,
           TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME,
           AS_LOCATOR, CREATED, LAST_ALTERED, NEW_SAVEPOINT_LEVEL, IS_UDT_DEPENDENT,
           DT.DATA_TYPE AS RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
           DT.CHARACTER_MAXIMUM_LENGTH AS RESULT_CAST_CHAR_MAX_LENGTH,
           DT.CHARACTER_OCTET_LENGTH AS RESULT_CAST_CHAR_OCTET_LENGTH,
           DT.CHARACTER_SET_CATALOG AS RESULT_CAST_CHAR_SET_CATALOG,
```

```
                DT.CHARACTER_SET_SCHEMA AS RESULT_CAST_CHAR_SET_SCHEMA,
                DT.CHARACTER_SET_NAME AS RESULT_CAST_CHARACTER_SET_NAME,
                DT.COLLATION_CATALOG AS RESULT_CAST_COLLATION_CATALOG,
                DT.COLLATION_SCHEMA AS RESULT_CAST_COLLATION_SCHEMA,
                DT.COLLATION_NAME AS RESULT_CAST_COLLATION_NAME,
                DT.NUMERIC_PRECISION AS RESULT_CAST_NUMERIC_PRECISION,
                DT.NUMERIC_PRECISION_RADIX AS RESULT_CAST_NUMERIC_RADIX,
                DT.NUMERIC_SCALE AS RESULT_CAST_NUMERIC_SCALE,
                DT.DATETIME_PRECISION AS RESULT_CAST_DATETIME_PRECISION,
                DT.INTERVAL_TYPE AS RESULT_CAST_INTERVAL_TYPE,
                DT.INTERVAL_PRECISION AS RESULT_CAST_INTERVAL_PRECISION,
                DT.USER_DEFINED_TYPE_CATALOG AS RESULT_CAST_TYPE_UDT_CATALOG,
                DT.USER_DEFINED_TYPE_SCHEMA AS RESULT_CAST_TYPE_UDT_SCHEMA,
                DT.USER_DEFINED_TYPE_NAME AS RESULT_CAST_TYPE_UDT_NAME,
                DT.SCOPE_CATALOG AS RESULT_CAST_SCOPE_CATALOG,
                DT.SCOPE_SCHEMA AS RESULT_CAST_SCOPE_SCHEMA,
                DT.SCOPE_NAME AS RESULT_CAST_SCOPE_NAME,
                DT.MAXIMUM_CARDINALITY AS RESULT_CAST_MAX_CARDINALITY,
                DT.DTD_IDENTIFIER AS RESULT_CAST_DTD_IDENTIFIER,
                D.DECLARED_DATA_TYPE, D.DECLARED_NUMERIC_PRECISION,
                D.DECLARED_NUMERIC_SCALE,
                DT.DECLARED_DATA_TYPE AS RESULT_CAST_FROM_DECLARED_DATA_TYPE,
                DT.DECLARED_NUMERIC_PRECISION AS RESULT_CAST_DECLARED_NUMERIC_PRECISION,
                DT.DECLARED_NUMERIC_SCALE AS RESULT_CAST_DECLARED_NUMERIC_SCALE,
                RETURNS_ONLY_PASS_THROUGH, DESCRIBE_PROCEDURE_SPECIFIC_CATALOG,
                DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA, DESCRIBE_PROCEDURE_SPECIFIC_NAME,
                START_PROCEDURE_SPECIFIC_CATALOG, START_PROCEDURE_SPECIFIC_SCHEMA,
                START_PROCEDURE_SPECIFIC_NAME, FULFILL_PROCEDURE_SPECIFIC_CATALOG,
                FULFILL_PROCEDURE_SPECIFIC_SCHEMA, FULFILL_PROCEDURE_SPECIFIC_NAME,
                FINISH_PROCEDURE_SPECIFIC_CATALOG, FINISH_PROCEDURE_SPECIFIC_SCHEMA,
                FINISH_PROCEDURE_SPECIFIC_NAME
        FROM ( ( DEFINITION_SCHEMA.ROUTINES AS R
            LEFT JOIN
                DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
            ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', R.DTD_IDENTIFIER )
            = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
          LEFT JOIN
                DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DT
            ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', RESULT_CAST_FROM_DTD_IDENTIFIER )
            = ( DT.OBJECT_CATALOG, DT.OBJECT_SCHEMA, DT.OBJECT_NAME,
                DT.OBJECT_TYPE, DT.DTD_IDENTIFIER ) )
        WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IS NULL
                AND
                ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
                  ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
                    FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
                    WHERE ( GRANTEE IN
                            ( 'PUBLIC', CURRENT_USER )
                          OR
                            GRANTEE IN
                            ( SELECT ROLE_NAME
                              FROM ENABLED_ROLES ) ) )
          AND SPECIFIC_CATALOG
            = ( SELECT CATALOG_NAME
```

```
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROUTINES
    TO PUBLIC WITH GRANT OPTION;
```

NOTE 6 — The ROUTINES view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with "RESULT_CAST_" describes the data type specified in the <result cast>, if any, contained in the <SQL-invoked routine>, the other set of columns describes the data type specified in the <returns data type> contained in the <SQL-invoked routine>.

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.

2) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.CREATED.

3) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.LAST_ALTERED.

4) Without Feature T272, "Enhanced savepoint management", conforming SQL-language shall not reference INFORMATION_SCHEMA.ROUTINES.NEW_SAVEPOINT_LEVEL.

5) Without Feature T322, "Declared data type attributes", conforming SQL-language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, DECLARED_NUMERIC_SCALE, RESULT_CAST_FROM_DECLARED_DATA_TYPE, RESULT_CAST_DECLARED_NUMERIC_PRECISION, and RESULT_CAST_DECLARED_NUMERIC_SCALE.

6) Without Feature B200, "Polymorphic table functions", conforming SQL-language shall not reference any of the columns INFORMATION_SCHEMA.ROUTINES.RETURNS_ONLY_PASS_THROUGH, INFORMATION_SCHEMA.ROUTINES.DESCRIBE_PROCEDURE_SPECIFIC_CATALOG, INFORMATION_SCHEMA.ROUTINES.DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA, INFORMATION_SCHEMA.ROUTINES.DESCRIBE_PROCEDURE_SPECIFIC_NAME, INFORMATION_SCHEMA.ROUTINES.START_PROCEDURE_SPECIFIC_CATALOG, INFORMATION_SCHEMA.ROUTINES.START_PROCEDURE_SPECIFIC_SCHEMA, INFORMATION_SCHEMA.ROUTINES.START_PROCEDURE_SPECIFIC_NAME, INFORMATION_SCHEMA.ROUTINES.FULFILL_PROCEDURE_SPECIFIC_CATALOG, INFORMATION_SCHEMA.ROUTINES.FULFILL_PROCEDURE_SPECIFIC_SCHEMA, INFORMATION_SCHEMA.ROUTINES.FULFILL_PROCEDURE_SPECIFIC_NAME, INFORMATION_SCHEMA.ROUTINES.FINISH_PROCEDURE_SPECIFIC_CATALOG, INFORMATION_SCHEMA.ROUTINES.FINISH_PROCEDURE_SPECIFIC_SCHEMA, and INFORMATION_SCHEMA.ROUTINES.FINISH_PROCEDURE_SPECIFIC_NAME.

5.55 SCHEMATA view

Function

Identify the schemata in a catalog that are owned by a given user or role.

Definition

```
CREATE VIEW SCHEMATA AS
    SELECT CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER,
            DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
            DEFAULT_CHARACTER_SET_NAME, SQL_PATH
    FROM DEFINITION_SCHEMA.SCHEMATA
    WHERE ( SCHEMA_OWNER = CURRENT_USER
          OR
            SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
                FROM ENABLED_ROLES ) )
      AND
            CATALOG_NAME
        = ( SELECT CATALOG_NAME
              FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE SCHEMATA
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.SCHEMATA.

5.56 SEQUENCES view

Function

Identify the external sequence generators defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW SEQUENCES AS
    SELECT S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA, S.SEQUENCE_NAME,
           DTD.DATA_TYPE, DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX,
           DTD.NUMERIC_SCALE, S.START_VALUE, S.MINIMUM_VALUE,
           S.MAXIMUM_VALUE, S.INCREMENT, S.CYCLE_OPTION,
           DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
           DTD.DECLARED_NUMERIC_SCALE
    FROM DEFINITION_SCHEMA.SEQUENCES AS S
      JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
        ON ( ( S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA,
               S.SEQUENCE_NAME, 'SEQUENCE',
               S.DTD_IDENTIFIER )
           = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA,
               DTD.OBJECT_NAME, DTD.OBJECT_TYPE,
               DTD.DTD_IDENTIFIER ) )
    WHERE ( S.SEQUENCE_CATALOG, S.SEQUENCE_SCHEMA, S.SEQUENCE_NAME, 'SEQUENCE' ) IN
           ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME, UP.OBJECT_TYPE
             FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
             WHERE ( UP.GRANTEE IN
                     ( 'PUBLIC', CURRENT_USER )
                   OR
                     UP.GRANTEE IN
                     ( SELECT ER.ROLE_NAME
                       FROM ENABLED_ROLES AS ER ) ) )
      AND S.SEQUENCE_CATALOG
       = ( SELECT ISCN.CATALOG_NAME
           FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE SEQUENCES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.SEQUENCES.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.57   SQL_FEATURES view

Function

List the features and subfeatures of this standard, and indicate which of these the SQL-implementation supports.

Definition

```
CREATE VIEW SQL_FEATURES AS
    SELECT ID AS FEATURE_ID, NAME AS FEATURE_NAME, SUB_ID AS SUB_FEATURE_ID,
        SUB_NAME AS SUB_FEATURE_NAME, IS_SUPPORTED, IS_VERIFIED_BY, COMMENTS, TYPE
    FROM DEFINITION_SCHEMA.SQL_CONFORMANCE
    WHERE TYPE IN
        ( 'FEATURE', 'SUBFEATURE' );

GRANT SELECT ON TABLE SQL_FEATURES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.58   SQL_IMPLEMENTATION_INFO view

Function

List the SQL-implementation information items defined in this standard and, for each of these, indicate the value supported by the SQL-implementation.

Definition

```
CREATE VIEW SQL_IMPLEMENTATION_INFO AS
    SELECT IMPLEMENTATION_INFO_ID, IMPLEMENTATION_INFO_NAME,
           INTEGER_VALUE, CHARACTER_VALUE, COMMENTS
    FROM DEFINITION_SCHEMA.SQL_IMPLEMENTATION_INFO;

GRANT SELECT ON TABLE SQL_IMPLEMENTATION_INFO
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1)   Without Feature F502, "Enhanced documentation tables", conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO.

2)   Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.SQL_IMPLEMENTATION_INFO.

5.59 SQL_PARTS view

Function

List the parts of this standard, and indicate which of these the SQL-implementation supports.

Definition

```
CREATE VIEW SQL_PARTS AS
    SELECT ID AS PART, NAME, IS_SUPPORTED, IS_VERIFIED_BY, COMMENTS
    FROM DEFINITION_SCHEMA.SQL_CONFORMANCE
    WHERE TYPE = 'PART';

GRANT SELECT ON TABLE SQL_PARTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F502, "Enhanced documentation tables", conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_PARTS.

5.60   SQL_SIZING view

Function

List the sizing items defined in this standard and, for each of these, indicate the size supported by the SQL-implementation.

Definition

```
CREATE VIEW SQL_SIZING AS
    SELECT SIZING_ID, SIZING_NAME, SUPPORTED_VALUE, COMMENTS
    FROM DEFINITION_SCHEMA.SQL_SIZING;

GRANT SELECT ON TABLE SQL_SIZING
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.61   TABLE_CONSTRAINTS view

Function

Identify the table constraints defined on tables in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TABLE_CONSTRAINTS AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
            TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            CONSTRAINT_TYPE, IS_DEFERRABLE, INITIALLY_DEFERRED,
            ENFORCED
    FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS
    WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
              FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
              WHERE TP.PRIVILEGE_TYPE <> 'SELECT'
                AND
                    ( TP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                  OR
                      TP.GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES ) )
            UNION
              SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
              FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
              WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
                AND ( CP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                  OR
                      CP.GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES ) ) )
        AND
            CONSTRAINT_CATALOG
        = ( SELECT CATALOG_NAME FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_CONSTRAINTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

None.

5.62 TABLE_METHOD_PRIVILEGES view

Function

Identify the privileges on methods of tables of structured type defined in those catalogs that are available to or granted by a given user or role.

Definition

```
CREATE VIEW TABLE_METHOD_PRIVILEGES AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.TABLE_METHOD_PRIVILEGES
    WHERE ( GRANTEE IN
            ( 'PUBLIC', CURRENT_USER )
        OR
            GRANTEE IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES )
        OR
            GRANTOR
          = CURRENT_USER
        OR
            GRANTOR IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
      AND
        TABLE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_METHOD_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S024, "Enhanced structured types", conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TABLE_METHOD_PRIVILEGES.

5.63 TABLE_PRIVILEGES view

Function

Identify the privileges on tables defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW TABLE_PRIVILEGES AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           PRIVILEGE_TYPE, IS_GRANTABLE, WITH_HIERARCHY
    FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
    WHERE ( GRANTEE IN
           ( 'PUBLIC', CURRENT_USER )
        OR
           GRANTEE IN
           ( SELECT ROLE_NAME
             FROM ENABLED_ROLES )
        OR
           GRANTOR
         = CURRENT_USER
        OR
           GRANTOR IN
           ( SELECT ROLE_NAME
             FROM ENABLED_ROLES ) )
      AND
        TABLE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TABLE_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.TABLE_PRIVILEGES.

5.64   TABLES view

Function

Identify the tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TABLES AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_TYPE,
           SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
           USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
           USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO, IS_TYPED,
           COMMIT_ACTION
    FROM DEFINITION_SCHEMA.TABLES
    WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
          ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
            FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
            WHERE ( TP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                OR
                    GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) )
          UNION
            SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
            FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
            WHERE ( CP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                OR
                    CP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
    AND
        TABLE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TABLES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
    TION_SCHEMA.TABLES.

5.65 TRANSFORMS view

Function

Identify the transforms on user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRANSFORMS AS
    SELECT USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           USER_DEFINED_TYPE_NAME AS UDT_NAME,
           SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           GROUP_NAME, TRANSFORM_TYPE
    FROM DEFINITION_SCHEMA.TRANSFORMS
    WHERE ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IN
          ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
                   UDTP.USER_DEFINED_TYPE_NAME
            FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
            WHERE ( UDTP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    UDTP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
      AND
        USER_DEFINED_TYPE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRANSFORMS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature S241, "Transform functions", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.TRANSFORMS.

5.66 TRANSLATIONS view

Function

Identify the character transliterations defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRANSLATIONS AS
    SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
           SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
           SOURCE_CHARACTER_SET_NAME,
           TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
           TARGET_CHARACTER_SET_NAME,
           TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
           TRANSLATION_SOURCE_NAME
    FROM DEFINITION_SCHEMA.TRANSLATIONS
    WHERE ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME, 'TRANSLATION') IN
          ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA, UP.OBJECT_NAME, UP.OBJECT_TYPE
            FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
            WHERE ( UP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                      UP.GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES ) ) )
      AND
          TRANSLATION_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE TRANSLATIONS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRANSLATIONS.

2) Without Feature F695, "Translation support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRANSLATIONS.

3) Without Feature F696, "Additional translation documentation", conforming SQL language shall not reference TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA, or TRANSLA-TION_SOURCE_NAME.

5.67 TRIGGERED_UPDATE_COLUMNS view

Function

Identify the columns in this catalog that are identified by the explicit UPDATE trigger event columns of a trigger defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRIGGERED_UPDATE_COLUMNS AS
    SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
           EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
           EVENT_OBJECT_COLUMN
    FROM DEFINITION_SCHEMA.TRIGGERED_UPDATE_COLUMNS
    WHERE ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
            EVENT_OBJECT_TABLE, EVENT_OBJECT_COLUMN ) IN
          ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA,
                   CP.TABLE_NAME, CP.COLUMN_NAME
            FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
            WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
              AND
                ( CP.GRANTEE IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  CP.GRANTEE IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) )
    AND
       TRIGGER_CATALOG
     = ( SELECT CATALOG_NAME
         FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERED_UPDATE_COLUMNS
   TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.

2) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-MATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS.

5.68   TRIGGER_COLUMN_USAGE view

Function

Identify the columns on which triggers defined in this catalog and owned by a given user are dependent because of their reference by the search condition or in their appearance in a triggered SQL statement of a trigger owned by a given user or role.

Definition

```
CREATE VIEW TRIGGER_COLUMN_USAGE AS
    SELECT TCU.TRIGGER_CATALOG, TCU.TRIGGER_SCHEMA, TCU.TRIGGER_NAME,
           TCU.TABLE_CATALOG, TCU.TABLE_SCHEMA, TCU.TABLE_NAME, TCU.COLUMN_NAME
    FROM DEFINITION_SCHEMA.TRIGGER_COLUMN_USAGE AS TCU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
       ON ( ( TCU.TRIGGER_CATALOG, TCU.TRIGGER_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND TCU.TRIGGER_CATALOG
          = ( SELECT ISCN.CATALOG_NAME
              FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE TRIGGER_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_COLUMN_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGERED_COLUMN_USAGE.

3) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-MATION_SCHEMA.TRIGGER_COLUMN_USAGE.

5.69 TRIGGER_PERIOD_USAGE view

Function

Identify the periods in which triggers defined in this catalog and owned by a given user or role are dependent because of their reference by the search condition or in their appearance in a triggered SQL statement of a trigger owned by a given user or role.

Definition

```
CREATE VIEW TRIGGER_PERIOD_USAGE AS
    SELECT TPU.TRIGGER_CATALOG, TPU.TRIGGER_SCHEMA, TPU.TRIGGER_NAME,
            TPU.TABLE_CATALOG, TPU.TABLE_SCHEMA, TPU.TABLE_NAME, TPU.PERIOD_NAME
    FROM DEFINITION_SCHEMA.TRIGGER_PERIOD_USAGE AS TPU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( TPU.TRIGGER_CATALOG, TPU.TRIGGER_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
      AND TPU.TRIGGER_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE TRIGGER_PERIOD_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_PERIOD_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGERED_PERIOD_USAGE.

3) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_PERIOD_USAGE.

4) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_PERIOD_USAGE.

5.70   TRIGGER_ROUTINE_USAGE view

Function

Identify each SQL-invoked routine owned by a given user or role on which some trigger defined in this catalog is dependent.

Definition

```
CREATE VIEW TRIGGER_ROUTINE_USAGE AS
    SELECT TRU.TRIGGER_CATALOG, TRU.TRIGGER_SCHEMA, TRU.TRIGGER_NAME,
           TRU.SPECIFIC_CATALOG, TRU.SPECIFIC_SCHEMA, TRU.SPECIFIC_NAME
    FROM DEFINITION_SCHEMA.TRIGGER_ROUTINE_USAGE AS TRU
      JOIN
         DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( TRU.TRIGGER_CATALOG, TRU.TRIGGER_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
         OR
           S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          TRU.SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE TRIGGER_ROUTINE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_ROUTINE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_ROUTINE_USAGE.

3) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-MATION_SCHEMA.TRIGGER_ROUTINE_USAGE.

5.71 TRIGGER_SEQUENCE_USAGE view

Function

Identify each external sequence generator owned by a given user or role on which some trigger defined in this catalog is dependent.

Definition

```
CREATE VIEW TRIGGER_SEQUENCE_USAGE AS
    SELECT TSU.TRIGGER_CATALOG, TSU.TRIGGER_SCHEMA, TSU.TRIGGER_NAME,
           TSU.SEQUENCE_CATALOG, TSU.SEQUENCE_SCHEMA, TSU.SEQUENCE_NAME
    FROM DEFINITION_SCHEMA.TRIGGER_SEQUENCE_USAGE AS TSU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( TSU.TRIGGER_CATALOG, TSU.TRIGGER_SCHEMA )
            = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
           ( SELECT ER.ROLE_NAME
             FROM ENABLED_ROLES AS ER ) )
      AND
          TSU.SEQUENCE_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE TRIGGER_SEQUENCE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-
    TION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-
    TION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

3) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference
    INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

4) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-
    MATION_SCHEMA.TRIGGER_SEQUENCE_USAGE.

5.72 TRIGGER_TABLE_USAGE view

Function

Identify the tables on which triggers defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW TRIGGER_TABLE_USAGE AS
    SELECT TTU.TRIGGER_CATALOG, TTU.TRIGGER_SCHEMA, TTU.TRIGGER_NAME,
           TTU.TABLE_CATALOG, TTU.TABLE_SCHEMA, TTU.TABLE_NAME
    FROM DEFINITION_SCHEMA.TRIGGER_TABLE_USAGE AS TTU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
      ON ( ( TTU.TRIGGER_CATALOG, TTU.TRIGGER_SCHEMA )
         = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
      AND
        TTU.TRIGGER_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE TRIGGER_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_TABLE_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGER_TABLE_USAGE.

3) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-MATION_SCHEMA.TRIGGER_TABLE_USAGE.

5.73   TRIGGERS view

Function

Identify the triggers defined on tables in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW TRIGGERS AS
    SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
           EVENT_MANIPULATION,
           EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
           ACTION_ORDER,
           CASE
             WHEN EXISTS
               ( SELECT *
                 FROM DEFINITION_SCHEMA.SCHEMATA AS S
                 WHERE ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                   AND
                     ( S.SCHEMA_OWNER = CURRENT_USER
                   OR
                       S.SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN ACTION_CONDITION
             ELSE NULL
           END AS ACTION_CONDITION,
           CASE
             WHEN EXISTS
               ( SELECT *
                 FROM DEFINITION_SCHEMA.SCHEMATA AS S
                 WHERE ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                     = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                   AND
                     ( S.SCHEMA_OWNER = CURRENT_USER
                   OR
                       S.SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN ACTION_STATEMENT
             ELSE NULL
           END AS ACTION_STATEMENT,
           ACTION_ORIENTATION, ACTION_TIMING,
           ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
           ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW,
           CREATED
    FROM DEFINITION_SCHEMA.TRIGGERS
    WHERE ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE ) IN
           ( SELECT TP.TABLE_CATALOG, TP.TABLE_SCHEMA, TP.TABLE_NAME
             FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES AS TP
             WHERE TP.PRIVILEGE_TYPE <> 'SELECT'
               AND
```

```
                          ( TP.GRANTEE IN
                            ( 'PUBLIC', CURRENT_USER )
                        OR
                            TP.GRANTEE IN
                            ( SELECT ROLE_NAME
                              FROM ENABLED_ROLES ) )
              UNION
                SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME
                FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
                 WHERE CP.PRIVILEGE_TYPE <> 'SELECT'
                   AND
                       ( CP.GRANTEE IN
                         ( 'PUBLIC', CURRENT_USER )
                     OR
                         CP.GRANTEE IN
                         ( SELECT ROLE_NAME
                           FROM ENABLED_ROLES ) ) )
        AND
            TRIGGER_CATALOG
          = ( SELECT CATALOG_NAME
              FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIGGERS.

2) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS.TRIGGER_CREATED.

3) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFOR-MATION_SCHEMA.TRIGGERS.

5.74 UDT_PRIVILEGES view

Function

Identify the privileges on user-defined types defined in this catalog that are accessible to or granted by a given user or role.

Definition

```
CREATE VIEW UDT_PRIVILEGES AS
    SELECT GRANTOR, GRANTEE,
           USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
           USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
           USER_DEFINED_TYPE_NAME AS UDT_NAME,
           PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
    WHERE ( GRANTEE IN
            ( 'PUBLIC', CURRENT_USER )
        OR
          GRANTEE IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES )
        OR
          GRANTOR
        = CURRENT_USER
        OR
          GRANTOR IN
          ( SELECT ROLE_NAME
            FROM ENABLED_ROLES ) )
      AND
        USER_DEFINED_TYPE_CATALOG
      = ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE UDT_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-
   TION_SCHEMA.UDT_PRIVILEGES.

5.75 USAGE_PRIVILEGES view

Function

Identify the USAGE privileges on objects defined in this catalog that are available to or granted by a given user or role.

Definition

```
CREATE VIEW USAGE_PRIVILEGES AS
    SELECT GRANTOR, GRANTEE,
           OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
           OBJECT_TYPE, 'USAGE' AS PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
    WHERE ( GRANTEE IN
           ( 'PUBLIC', CURRENT_USER )
        OR
           GRANTEE IN
           ( SELECT ROLE_NAME
             FROM ENABLED_ROLES )
        OR
           GRANTOR
         = CURRENT_USER
        OR
           GRANTOR IN
           ( SELECT ROLE_NAME
             FROM ENABLED_ROLES ) )
      AND
           OBJECT_CATALOG
         = ( SELECT CATALOG_NAME
             FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE USAGE_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.USAGE_PRIVILEGES.

5.76   USER_DEFINED_TYPES view

This Subclause is modified by Subclause 14.6, "USER_DEFINED_TYPES view", in ISO/IEC 9075-13.

Function

Identify the user-defined types defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW USER_DEFINED_TYPES AS
    SELECT UDT.USER_DEFINED_TYPE_CATALOG, UDT.USER_DEFINED_TYPE_SCHEMA,
           UDT.USER_DEFINED_TYPE_NAME, UDT.USER_DEFINED_TYPE_CATEGORY,
           UDT.IS_INSTANTIABLE, UDT.IS_FINAL, UDT.ORDERING_FORM,
           UDT.ORDERING_CATEGORY, UDT.ORDERING_ROUTINE_CATALOG,
           UDT.ORDERING_ROUTINE_SCHEMA, UDT.ORDERING_ROUTINE_NAME, UDT.REFERENCE_TYPE,
           DTD.DATA_TYPE, DTD.CHARACTER_MAXIMUM_LENGTH, DTD.CHARACTER_OCTET_LENGTH,
           DTD.CHARACTER_SET_CATALOG, DTD.CHARACTER_SET_SCHEMA, DTD.CHARACTER_SET_NAME,
           DTD.COLLATION_CATALOG, DTD.COLLATION_SCHEMA, DTD.COLLATION_NAME,
           DTD.NUMERIC_PRECISION, DTD.NUMERIC_PRECISION_RADIX, DTD.NUMERIC_SCALE,
           DTD.DATETIME_PRECISION, DTD.INTERVAL_TYPE, DTD.INTERVAL_PRECISION,
           UDT.SOURCE_DTD_IDENTIFIER, UDT.REF_DTD_IDENTIFIER,
           DTD.DECLARED_DATA_TYPE, DTD.DECLARED_NUMERIC_PRECISION,
           DTD.DECLARED_NUMERIC_SCALE, DTD.MAXIMUM_CARDINALITY
    FROM ( DEFINITION_SCHEMA.USER_DEFINED_TYPES AS UDT
        LEFT JOIN
           DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS DTD
           ON ( ( UDT.USER_DEFINED_TYPE_CATALOG, UDT.USER_DEFINED_TYPE_SCHEMA,
                  UDT.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                  UDT.SOURCE_DTD_IDENTIFIER )
              = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA,
                  DTD.OBJECT_NAME, DTD.OBJECT_TYPE,
                  DTD.DTD_IDENTIFIER )
            OR
                ( UDT.USER_DEFINED_TYPE_CATALOG, UDT.USER_DEFINED_TYPE_SCHEMA,
                  UDT.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                  UDT.REF_DTD_IDENTIFIER )
              = ( DTD.OBJECT_CATALOG, DTD.OBJECT_SCHEMA,
                  DTD.OBJECT_NAME, DTD.OBJECT_TYPE,
                  DTD.DTD_IDENTIFIER ) ) )
    WHERE ( UDT.USER_DEFINED_TYPE_CATALOG, UDT.USER_DEFINED_TYPE_SCHEMA,
            UDT.USER_DEFINED_TYPE_NAME ) IN
          ( SELECT UDTP.USER_DEFINED_TYPE_CATALOG, UDTP.USER_DEFINED_TYPE_SCHEMA,
                   UDTP.USER_DEFINED_TYPE_NAME
            FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES AS UDTP
            WHERE ( UDTP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    UDTP.GRANTEE IN
                    ( SELECT ER.ROLE_NAME
                      FROM ENABLED_ROLES AS ER ) ) )
      AND
          UDT.USER_DEFINED_TYPE_CATALOG
```

```
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE USER_DEFINED_TYPES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.USER_DEFINED_TYPES.

2) Without Feature S401, "Distinct types based on array types", conforming SQL language shall not reference INFORMATION_SCHEMA.USER_DEFINED_TYPES.MAXIMUM_CARDINALITY.

3) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of the columns DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE.

5.77 VIEW_COLUMN_USAGE view

Function

Identify the columns on which viewed tables defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW VIEW_COLUMN_USAGE AS
    SELECT VCU.VIEW_CATALOG, VCU.VIEW_SCHEMA, VCU.VIEW_NAME,
          VCU.TABLE_CATALOG, VCU.TABLE_SCHEMA, VCU.TABLE_NAME, VCU.COLUMN_NAME
    FROM DEFINITION_SCHEMA.VIEW_COLUMN_USAGE AS VCU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( VCU.TABLE_CATALOG, VCU.TABLE_SCHEMA )
            = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
          S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
      AND
        VCU.VIEW_CATALOG
      = ( SELECT ISCN.CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE VIEW_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.VIEW_COLUMN_USAGE.

5.78 VIEW_PERIOD_USAGE view

Function

Identify the periods on which viewed tables defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW VIEW_PERIOD_USAGE AS
    SELECT VPU.VIEW_CATALOG, VPU.VIEW_SCHEMA, VPU.VIEW_NAME,
           VPU.TABLE_CATALOG, VPU.TABLE_SCHEMA, VPU.TABLE_NAME, VPU.PERIOD_NAME
    FROM DEFINITION_SCHEMA.VIEW_PERIOD_USAGE AS VPU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( VPU.TABLE_CATALOG, VPU.TABLE_SCHEMA )
           = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          VPU.VIEW_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE VIEW_PERIOD_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.VIEW_PERIOD_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMA-TION_SCHEMA.VIEW_PERIOD_USAGE.

3) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_PERIOD_USAGE.

4) Without Feature T181, "Application-time period tables", conforming SQL language shall not reference INFORMATION_SCHEMA.VIEW_PERIOD_USAGE.

5.79   VIEW_ROUTINE_USAGE view

Function

Identify each routine owned by a given user or role on which a view defined in this catalog is dependent.

Definition

```
CREATE VIEW VIEW_ROUTINE_USAGE AS
    SELECT VRU.TABLE_CATALOG, VRU.TABLE_SCHEMA, VRU.TABLE_NAME,
           VRU.SPECIFIC_CATALOG, VRU.SPECIFIC_SCHEMA, VRU.SPECIFIC_NAME
    FROM DEFINITION_SCHEMA.VIEW_ROUTINE_USAGE AS VRU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
       ON ( ( VRU.TABLE_CATALOG, VRU.TABLE_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
      AND
          VRU.SPECIFIC_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE VIEW_ROUTINE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.VIEW_ROUTINE_USAGE.

5.80 VIEW_TABLE_USAGE view

Function

Identify the tables on which viewed tables defined in this catalog and owned by a given user or role are dependent.

Definition

```
CREATE VIEW VIEW_TABLE_USAGE AS
    SELECT VTU.VIEW_CATALOG, VTU.VIEW_SCHEMA, VTU.VIEW_NAME,
           VTU.TABLE_CATALOG, VTU.TABLE_SCHEMA, VTU.TABLE_NAME
    FROM DEFINITION_SCHEMA.VIEW_TABLE_USAGE AS VTU
      JOIN
        DEFINITION_SCHEMA.SCHEMATA AS S
        ON ( ( VTU.TABLE_CATALOG, VTU.TABLE_SCHEMA )
          = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
WHERE ( S.SCHEMA_OWNER = CURRENT_USER
      OR
        S.SCHEMA_OWNER IN
        ( SELECT ER.ROLE_NAME
          FROM ENABLED_ROLES AS ER ) )
      AND
          VTU.VIEW_CATALOG
        = ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE VIEW_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.VIEW_TABLE_USAGE.

5.81   VIEWS view

Function

Identify the viewed tables defined in this catalog that are accessible to a given user or role.

Definition

```
CREATE VIEW VIEWS AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          CASE
            WHEN EXISTS
                ( SELECT *
                  FROM DEFINITION_SCHEMA.SCHEMATA AS S
                  WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
                      = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                    AND
                        ( S.SCHEMA_OWNER = CURRENT_USER
                      OR
                          S.SCHEMA_OWNER IN
                          ( SELECT ER.ROLE_NAME
                            FROM ENABLED_ROLES AS ER ) ) )
            THEN V.VIEW_DEFINITION
            ELSE NULL
          END AS VIEW_DEFINITION,
          V.CHECK_OPTION, V.IS_UPDATABLE, T.IS_INSERTABLE_INTO AS INSERTABLE_INTO,
          V.IS_TRIGGER_UPDATABLE, V.IS_TRIGGER_DELETABLE, V.IS_TRIGGER_INSERTABLE_INTO
    FROM DEFINITION_SCHEMA.VIEWS AS V
    JOIN INFORMATION_SCHEMA.TABLES AS T
    USING (TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME);

GRANT SELECT ON TABLE VIEWS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature T213, "INSTEAD OF triggers", conforming SQL language shall not reference any of the columns IS_TRIGGER_UPDATABLE, IS_TRIGGER_DELETABLE, IS_TRIG-GER_INSERTABLE_INTO.

5.82   Short name views

This Subclause is modified by Subclause 19.8, "Short name views", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 24.14, "Short name views", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 14.7, "Short name views", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 21.15, "Short name views", in ISO/IEC 9075-14.

Function

Provide alternative views that use only identifiers that do not require Feature F391, "Long identifiers".

Definition

```
CREATE VIEW CATALOG_NAME
        ( CATALOG_NAME ) AS
    SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME;

GRANT SELECT ON TABLE CATALOG_NAME
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ADMIN_ROLE_AUTHS
        ( GRANTEE, ROLE_NAME, IS_GRANTABLE ) AS
    SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS;

GRANT SELECT ON TABLE ADMIN_ROLE_AUTHS
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ATTRIBUTES_S
        ( UDT_CATALOG,         UDT_SCHEMA,          UDT_NAME,
          ATTRIBUTE_NAME,      ORDINAL_POSITION,    ATTRIBUTE_DEFAULT,
          DATA_TYPE,           CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,
          CHAR_SET_CATALOG,    CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,
          COLLATION_CATALOG,   COLLATION_SCHEMA,    COLLATION_NAME,
          NUMERIC_PRECISION,   NUMERIC_PREC_RADIX,  NUMERIC_SCALE,
          DATETIME_PRECISION,  INTERVAL_TYPE,       INTERVAL_PRECISION,
          ATT_UDT_CAT,         ATT_UDT_SCHEMA,      ATT_UDT_NAME,
          SCOPE_CATALOG,       SCOPE_SCHEMA,        SCOPE_NAME,
          MAX_CARDINALITY,     DTD_IDENTIFIER,      IS_DERIVED_REF_ATT,
          DECLARED_DATA_TYPE,  DECLARED_NUM_PREC,   DECLARED_NUM_SCALE) AS
    SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
          ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
          DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
          NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
          DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
          ATTRIBUTE_UDT_CATALOG, ATTRIBUTE_UDT_SCHEMA, ATTRIBUTE_UDT_NAME,
          SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
          MAXIMUM_CARDINALITY, DTD_IDENTIFIER, IS_DERIVED_REFERENCE_ATTRIBUTE,
```

```
          DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
          DECLARED_NUMERIC_SCALE
     FROM INFORMATION_SCHEMA.ATTRIBUTES;

GRANT SELECT ON TABLE ATTRIBUTES_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW CHARACTER_SETS_S
        ( CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,
          CHAR_REPERTOIRE,    FORM_OF_USE,
          DEF_COLLATE_CAT,    DEF_COLLATE_SCHEMA, DEF_COLLATE_NAME ) AS
     SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          CHARACTER_REPERTOIRE, FORM_OF_USE,
          DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, DEFAULT_COLLATE_NAME
     FROM INFORMATION_SCHEMA.CHARACTER_SETS;

GRANT SELECT ON TABLE CHARACTER_SETS_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW COLLATION_APPLIC_S
        ( COLLATION_CATALOG,  COLLATION_SCHEMA,  COLLATION_NAME,
          CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,   CHARACTER_SET_NAME ) AS
     SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
          CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME
     FROM INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY;

GRANT SELECT ON TABLE COLLATION_APPLIC_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW COL_COL_USAGE
        ( TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
          COLUMN_NAME,        DEPENDENT_COLUMN ) AS
     SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          COLUMN_NAME, DEPENDENT_COLUMN
     FROM INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE;

GRANT SELECT ON TABLE COL_COL_USAGE
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW COL_DOMAIN_USAGE
        ( DOMAIN_CATALOG,     DOMAIN_SCHEMA,       DOMAIN_NAME,
          TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
          COLUMN_NAME ) AS
     SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
          TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
     FROM INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE;

GRANT SELECT ON TABLE COL_DOMAIN_USAGE
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW COLUMNS_S
        ( TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
```

```
            COLUMN_NAME,           ORDINAL_POSITION,    COLUMN_DEFAULT,
            IS_NULLABLE,           DATA_TYPE,           CHAR_MAX_LENGTH,
            CHAR_OCTET_LENGTH,     NUMERIC_PRECISION,   NUMERIC_PREC_RADIX,
            NUMERIC_SCALE,         DATETIME_PRECISION,  INTERVAL_TYPE,
            INTERVAL_PRECISION,    CHAR_SET_CATALOG,    CHAR_SET_SCHEMA,
            CHARACTER_SET_NAME,    COLLATION_CATALOG,   COLLATION_SCHEMA,
            COLLATION_NAME,        DOMAIN_CATALOG,      DOMAIN_SCHEMA,
            DOMAIN_NAME,           UDT_CATALOG,         UDT_SCHEMA,
            UDT_NAME,              SCOPE_CATALOG,       SCOPE_SCHEMA,
            SCOPE_NAME,            MAX_CARDINALITY,     DTD_IDENTIFIER,
            IS_SELF_REF,           IS_IDENTITY,         ID_GENERATION,
            ID_START,              ID_INCREMENT,        ID_MAXIMUM,
            ID_MINIMUM,            ID_CYCLE,            IS_GENERATED,
            GENERATION_EXPR,       IS_SYSPER_START,     IS_SYSPER_END,
            SYSPER_TSTMP_GEN,      IS_UPDATABLE,        DECLARED_DATA_TYPE,
            DECLARED_NUM_PREC,     DECLARED_NUM_SCALE) AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            COLUMN_NAME, ORDINAL_POSITION, COLUMN_DEFAULT,
            IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
            CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
            NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
            INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
            COLLATION_NAME, DOMAIN_CATALOG, DOMAIN_SCHEMA,
            DOMAIN_NAME, UDT_CATALOG, UDT_SCHEMA,
            UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
            SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
            IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
            IDENTITY_START, IDENTITY_INCREMENT, IDENTITY_MAXIMUM,
            IDENTITY_MINIMUM, IDENTITY_CYCLE, IS_GENERATED,
            GENERATION_EXPRESSION, IS_SYSTEM_TIME_PERIOD_START, IS_SYSTEM_TIME_PERIOD_END,
            SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION, IS_UPDATABLE, DECLARED_DATA_TYPE,
            DECLARED_NUMERIC_PRECISION, DECLARED_NUMERIC_SCALE
    FROM INFORMATION_SCHEMA.COLUMNS;

GRANT SELECT ON TABLE COLUMNS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW CONSTR_ROUT_USE_S
        ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,  CONSTRAINT_NAME,
          SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,    SPECIFIC_NAME ) AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
          SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
    FROM INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE;

GRANT SELECT ON TABLE CONSTR_ROUT_USE_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW CONSTR_COL_USAGE
        ( TABLE_CATALOG,       TABLE_SCHEMA,        TABLE_NAME,
          COLUMN_NAME,         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
          CONSTRAINT_NAME ) AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          COLUMN_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
          CONSTRAINT_NAME
```

```
    FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE;

GRANT SELECT ON TABLE CONSTR_COL_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW CONSTR_PER_USAGE
        ( TABLE_CATALOG,       TABLE_SCHEMA,        TABLE_NAME,
          PERIOD_NAME,         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
          CONSTRAINT_NAME ) AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          PERIOD_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
          CONSTRAINT_NAME
    FROM INFORMATION_SCHEMA.CONSTRAINT_PERIOD_USAGE;

GRANT SELECT ON TABLE CONSTR_PER_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW CONSTR_TABLE_USAGE
        ( TABLE_CATALOG,       TABLE_SCHEMA,        TABLE_NAME,
          CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,  CONSTRAINT_NAME ) AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
    FROM INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE;

GRANT SELECT ON TABLE CONSTR_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW DOMAINS_S
        ( DOMAIN_CATALOG,      DOMAIN_SCHEMA,       DOMAIN_NAME,
          DATA_TYPE,           CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,
          CHAR_SET_CATALOG,    CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,
          COLLATION_CATALOG,   COLLATION_SCHEMA,    COLLATION_NAME,
          NUMERIC_PRECISION,   NUMERIC_PREC_RADIX, NUMERIC_SCALE,
          DATETIME_PRECISION,  INTERVAL_TYPE,       INTERVAL_PRECISION,
          DOMAIN_DEFAULT,      MAX_CARDINALITY,     DTD_IDENTIFIER,
          DECLARED_DATA_TYPE, DECLARED_NUM_PREC,   DECLARED_NUM_SCALE) AS
    SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
          DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
          NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
          DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
          DOMAIN_DEFAULT, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
          DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
          DECLARED_NUMERIC_SCALE
    FROM INFORMATION_SCHEMA.DOMAINS;

GRANT SELECT ON TABLE DOMAINS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ELEMENT_TYPES_S
        ( OBJECT_CATALOG,      OBJECT_SCHEMA,       OBJECT_NAME,
          OBJECT_TYPE,         COLLECTION_TYPE_ID, DATA_TYPE,
```

```
            CHAR_MAX_LENGTH,      CHAR_OCTET_LENGTH,    CHAR_SET_CATALOG,
            CHAR_SET_SCHEMA,      CHARACTER_SET_NAME,   COLLATION_CATALOG,
            COLLATION_SCHEMA,     COLLATION_NAME,       NUMERIC_PRECISION,
            NUMERIC_PREC_RADIX, NUMERIC_SCALE,          DATETIME_PRECISION,
            INTERVAL_TYPE,        INTERVAL_PRECISION, UDT_CATALOG,
            UDT_SCHEMA,           UDT_NAME,             SCOPE_CATALOG,
            SCOPE_SCHEMA,         SCOPE_NAME,           MAX_CARDINALITY,
            DTD_IDENTIFIER,
            DECLARED_DATA_TYPE, DECLARED_NUM_PREC,  DECLARED_NUM_SCALE) AS
    SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER, DATA_TYPE,
            CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
            CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
            COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
            NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
            INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
            UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
            SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
            DTD_IDENTIFIER, DECLARED_DATA_TYPE,
            DECLARED_NUMERIC_PRECISION, DECLARED_NUMERIC_SCALE
    FROM INFORMATION_SCHEMA.ELEMENT_TYPES;

GRANT SELECT ON TABLE ELEMENT_TYPES_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW FIELDS_S
        ( OBJECT_CATALOG,      OBJECT_SCHEMA,      OBJECT_NAME,
            OBJECT_TYPE,         ROW_IDENTIFIER,     FIELD_NAME,
            ORDINAL_POSITION,    DATA_TYPE,          CHAR_MAX_LENGTH,
            CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,
            CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
            COLLATION_NAME,      NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
            NUMERIC_SCALE,       DATETIME_PRECISION, INTERVAL_TYPE,
            INTERVAL_PRECISION, UDT_CATALOG,         UDT_SCHEMA,
            UDT_NAME,            SCOPE_CATALOG,      SCOPE_SCHEMA,
            SCOPE_NAME,          MAX_CARDINALITY,    DTD_IDENTIFIER,
            DECLARED_DATA_TYPE, DECLARED_NUM_PREC,  DECLARED_NUM_SCALE) AS
    SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
            ORDINAL_POSITION, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
            CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
            COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
            NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
            INTERVAL_PRECISION, UDT_CATALOG, UDT_SCHEMA,
            UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
            SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
            DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE
    FROM INFORMATION_SCHEMA.FIELDS;

GRANT SELECT ON TABLE FIELDS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW KEY_COLUMN_USAGE_S
```

```
        ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,  CONSTRAINT_NAME,
          TABLE_CATALOG,      TABLE_SCHEMA,       TABLE_NAME,
          COLUMN_NAME,        ORDINAL_POSITION,  POSITION_IN_UC) AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
          TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          COLUMN_NAME, ORDINAL_POSITION, POSITION_IN_UNIQUE_CONSTRAINT
    FROM INFORMATION_SCHEMA. KEY_COLUMN_USAGE;

GRANT SELECT ON TABLE KEY_COLUMN_USAGE_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW METHOD_SPECS
        ( SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          UDT_CATALOG,        UDT_SCHEMA,         UDT_NAME,
          METHOD_NAME,        IS_STATIC,          IS_OVERRIDING,
          IS_CONSTRUCTOR,     DATA_TYPE,          CHAR_MAX_LENGTH,
          CHAR_OCTET_LENGTH,  CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,
          CHARACTER_SET_NAME, COLLATION_CATALOG,  COLLATION_SCHEMA,
          COLLATION_NAME,     NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,
          NUMERIC_SCALE,      DATETIME_PRECISION, INTERVAL_TYPE,
          INTERVAL_PRECISION, RETURN_UDT_CATALOG, RETURN_UDT_SCHEMA,
          RETURN_UDT_NAME,    SCOPE_CATALOG,      SCOPE_SCHEMA,
          SCOPE_NAME,         MAX_CARDINALITY,    DTD_IDENTIFIER,
          METHOD_LANGUAGE,    PARAMETER_STYLE,    IS_DETERMINISTIC,
          SQL_DATA_ACCESS,    IS_NULL_CALL,       TO_SQL_SPEC_CAT,
          TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME,   AS_LOCATOR,
          CREATED,            RC_FROM_DATA_TYPE,  RC_AS_LOCATOR,
          RC_CHAR_MAX_LENGTH, RC_CHAR_OCT_LENGTH, RC_CHAR_SET_CAT,
          RC_CHAR_SET_SCHEMA, RC_CHAR_SET_NAME,   RC_COLLATION_CAT,
          RC_COLLATION_SCH,   RC_COLLATION_NAME,  RC_NUMERIC_PREC,
          RC_NUMERIC_RADIX,   RC_NUMERIC_SCALE,   RC_DATETIME_PREC,
          RC_INTERVAL_TYPE,   RC_INTERVAL_PREC,   RC_TYPE_UDT_CAT,
          RC_TYPE_UDT_SCHEMA, RC_TYPE_UDT_NAME,   RC_SCOPE_CATALOG,
          RC_SCOPE_SCHEMA,    RC_SCOPE_NAME,      RC_MAX_CARDINALITY,
          RC_DTD_IDENTIFIER,  DECLARED_DATA_TYPE, DECLARED_NUM_PREC,
          DECLARED_NUM_SCALE) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
          UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
          METHOD_NAME, IS_STATIC, IS_OVERRIDING,
          IS_CONSTRUCTOR, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
          CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
          COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
          NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
          INTERVAL_PRECISION, RETURN_UDT_CATALOG, RETURN_UDT_SCHEMA,
          RETURN_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
          SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
          METHOD_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
          SQL_DATA_ACCESS, IS_NULL_CALL, TO_SQL_SPECIFIC_CATALOG,
          TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, AS_LOCATOR,
          CREATED, RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
          RESULT_CAST_CHAR_MAX_LENGTH, RESULT_CAST_CHAR_OCTET_LENGTH,
          RESULT_CAST_CHAR_SET_CATALOG, RESULT_CAST_CHAR_SET_SCHEMA,
          RESULT_CAST_CHAR_SET_NAME,
          RESULT_CAST_COLLATION_CATALOG, RESULT_CAST_COLLATION_SCHEMA,
          RESULT_CAST_COLLATION_NAME, RESULT_CAST_NUMERIC_PRECISION,
```

```
                RESULT_CAST_NUMERIC_RADIX, RESULT_CAST_NUMERIC_SCALE,
                RESULT_CAST_DATETIME_PRECISION, RESULT_CAST_INTERVAL_TYPE,
                RESULT_CAST_INTERVAL_PRECISION, RESULT_CAST_TYPE_UDT_CATALOG,
                RESULT_CAST_TYPE_UDT_SCHEMA, RESULT_CAST_TYPE_UDT_NAME,
                RESULT_CAST_SCOPE_CATALOG, RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,
                RESULT_CAST_MAX_CARDINALITY, RESULT_CAST_DTD_IDENTIFIER,
                DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
                DECLARED_NUMERIC_SCALE
        FROM INFORMATION_SCHEMA.METHOD_SPECIFICATIONS;

GRANT SELECT ON TABLE METHOD_SPECS
        TO PUBLIC WITH GRANT OPTION;


CREATE VIEW METHOD_SPEC_PARAMS
                ( SPECIFIC_CATALOG,     SPECIFIC_SCHEMA,     SPECIFIC_NAME,
                ORDINAL_POSITION,    PARAMETER_MODE,       IS_RESULT,
                AS_LOCATOR,          PARAMETER_NAME,       FROM_SQL_SPEC_CAT,
                FROM_SQL_SPEC_SCH,   FROM_SQL_SPEC_NAME, DATA_TYPE,
                CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,    CHAR_SET_CATALOG,
                CHAR_SET_SCHEMA,     CHARACTER_SET_NAME, COLLATION_CATALOG,
                COLLATION_SCHEMA,    COLLATION_NAME,       NUMERIC_PRECISION,
                NUMERIC_PREC_RADIX, NUMERIC_SCALE,        DATETIME_PRECISION,
                INTERVAL_TYPE,       INTERVAL_PRECISION, PARM_UDT_CATALOG,
                PARM_UDT_SCHEMA,     PARM_UDT_NAME,        SCOPE_CATALOG,
                SCOPE_SCHEMA,        SCOPE_NAME,           MAX_CARDINALITY,
                DTD_IDENTIFIER,
                DECLARED_DATA_TYPE, DECLARED_NUM_PREC,  DECLARED_NUM_SCALE) AS
        SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
                AS_LOCATOR, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
                FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, DATA_TYPE,
                CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
                CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
                COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
                NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
                INTERVAL_TYPE, INTERVAL_PRECISION, PARAMETER_UDT_CATALOG,
                PARAMETER_UDT_SCHEMA, PARAMETER_UDT_NAME, SCOPE_CATALOG,
                SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
                DTD_IDENTIFIER,
                DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
                DECLARED_NUMERIC_SCALE
        FROM INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS;

GRANT SELECT ON TABLE METHOD_SPEC_PARAMS
        TO PUBLIC WITH GRANT OPTION;


CREATE VIEW PARAMETERS_S
                ( SPECIFIC_CATALOG,     SPECIFIC_SCHEMA,     SPECIFIC_NAME,
                ORDINAL_POSITION,    PARAMETER_MODE,       IS_RESULT,
                AS_LOCATOR,          PARAMETER_NAME,       FROM_SQL_SPEC_CAT,
                FROM_SQL_SPEC_SCH,   FROM_SQL_SPEC_NAME, TO_SQL_SPEC_CAT,
                TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME,     DATA_TYPE,
                CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,    CHAR_SET_CATALOG,
                CHAR_SET_SCHEMA,     CHARACTER_SET_NAME, COLLATION_CATALOG,
                COLLATION_SCHEMA,    COLLATION_NAME,       NUMERIC_PRECISION,
```

```
            NUMERIC_PREC_RADIX, NUMERIC_SCALE,       DATETIME_PRECISION,
            INTERVAL_TYPE,      INTERVAL_PRECISION, UDT_CATALOG,
            UDT_SCHEMA,         UDT_NAME,            SCOPE_CATALOG,
            SCOPE_SCHEMA,       SCOPE_NAME,          MAX_CARDINALITY,
            DTD_IDENTIFIER,     DECLARED_DATA_TYPE, DECLARED_NUM_PREC,
            DECLARED_NUM_SCALE, PARAMETER_DEFAULT,  TABLE_SEMANTICS,
            IS_PRUNABLE,        HAS_PASS_THRU_COLS ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
            AS_LOCATOR, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
            FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, TO_SQL_SPECIFIC_CATALOG,
            TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, DATA_TYPE,
            CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
            CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
            COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
            NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
            INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
            UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
            SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
            DTD_IDENTIFIER, DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE, PARAMETER_DEFAULT, TABLE_SEMANTICS,
            IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS
    FROM INFORMATION_SCHEMA.PARAMETERS;

GRANT SELECT ON TABLE PARAMETERS_S
    TO PUBLIC WITH GRANT OPTION;



CREATE VIEW PRIVATE_PARAMS_S
        ( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,     SPECIFIC_NAME,
          ORDINAL_POSITION,    PARAMETER_NAME,      FROM_SQL_SPEC_CAT,
          FROM_SQL_SPEC_SCH,   FROM_SQL_SPEC_NAME,  TO_SQL_SPEC_CAT,
          TO_SQL_SPEC_SCHEMA,  TO_SQL_SPEC_NAME,    DATA_TYPE,
          CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,   CHAR_SET_CATALOG,
          CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,  COLLATION_CATALOG,
          COLLATION_SCHEMA,    COLLATION_NAME,      NUMERIC_PRECISION,
          NUMERIC_PREC_RADIX,  NUMERIC_SCALE,       DATETIME_PRECISION,
          INTERVAL_TYPE        INTERVAL_PRECISION, UDT_CATALOG,
          UDT_SCHEMA,          UDT_NAME,            SCOPE_CATALOG,
          SCOPE_SCHEMA,        SCOPE_NAME,          MAX_CARDINALITY,
          DTD_IDENTIFIER,      DECLARED_DATA_TYPE, DECLARED_NUM_PREC,
          DECLARED_NUM_SCALE, PARAMETER_DEFAULT,  TABLE_SEMANTICS,
          IS_PRUNABLE,         HAS_PASS_THRU_COLS ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            ORDINAL_POSITION, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
            FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, TO_SQL_SPECIFIC_CATALOG,
            TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, DATA_TYPE,
            CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
            CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
            COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
            NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
            INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
            UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
            SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
            DTD_IDENTIFIER, DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE, PARAMETER_DEFAULT, TABLE_SEMANTICS,
```

```
        IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS
    FROM INFORMATION_SCHEMA.PRIVATE_PARAMETERS;

GRANT SELECT ON TABLE PRIVATE_PARAMS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW REFERENCED_TYPES_S
        ( OBJECT_CATALOG,     OBJECT_SCHEMA,      OBJECT_NAME,
        OBJECT_TYPE,         REFERENCE_TYPE_ID,  DATA_TYPE,
        CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,  CHAR_SET_CATALOG,
        CHAR_SET_SCHEMA,     CHARACTER_SET_NAME, COLLATION_CATALOG,
        COLLATION_SCHEMA,    COLLATION_NAME,     NUMERIC_PRECISION,
        NUMERIC_PREC_RADIX,  NUMERIC_SCALE,      DATETIME_PRECISION,
        INTERVAL_TYPE,       INTERVAL_PRECISION, UDT_CATALOG,
        UDT_SCHEMA,          UDT_NAME,           SCOPE_CATALOG,
        SCOPE_SCHEMA,        SCOPE_NAME,         MAX_CARDINALITY,
        DTD_IDENTIFIER,
        DECLARED_DATA_TYPE,  DECLARED_NUM_PREC,  DECLARED_NUM_SCALE) AS
    SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
        CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
        CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
        COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
        NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
        INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
        UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
        SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
        DTD_IDENTIFIER,
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
        DECLARED_NUMERIC_SCALE
    FROM INFORMATION_SCHEMA.REFERENCED_TYPES;

GRANT SELECT ON TABLE REFERENCED_TYPES_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW REF_CONSTRAINTS
        ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
        UNIQUE_CONSTR_CAT,  UNIQUE_CONSTR_SCH,  UNIQUE_CONSTR_NAME,
        MATCH_OPTION,       UPDATE_RULE,        DELETE_RULE ) AS
    SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
        UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA, UNIQUE_CONSTRAINT_NAME,
        MATCH_OPTION, UPDATE_RULE, DELETE_RULE
    FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS;

GRANT SELECT ON TABLE REF_CONSTRAINTS
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROLE_ROUT_GRANTS
        ( GRANTOR,            GRANTEE,            SPECIFIC_CATALOG,
        SPECIFIC_SCHEMA,    SPECIFIC_NAME,      ROUTINE_CATALOG,
        ROUTINE_SCHEMA,     ROUTINE_NAME,       PRIVILEGE_TYPE,
        IS_GRANTABLE ) AS
    SELECT GRANTOR, GRANTEE, SPECIFIC_CATALOG,
        SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_CATALOG,
```

```
            ROUTINE_SCHEMA, ROUTINE_NAME, PRIVILEGE_TYPE,
            IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS;

GRANT SELECT ON TABLE ROLE_ROUT_GRANTS
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROL_TAB_METH_GRNTS
        ( GRANTOR,           GRANTEE,           TABLE_CATALOG,
          TABLE_SCHEMA,      TABLE_NAME,        SPECIFIC_CATALOG,
          SPECIFIC_SCHEMA,   SPECIFIC_NAME,     IS_GRANTABLE ) AS
    SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
          TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
          SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM INFORMATION_SCHEMA.ROLE_TABLE_METHOD_GRANTS;

GRANT SELECT ON TABLE ROL_TAB_METH_GRNTS
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROUT_ROUT_USAGE_S
        ( SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          ROUTINE_CATALOG,    ROUTINE_SCHEMA,     ROUTINE_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
          ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_ROUTINE_USAGE;

GRANT SELECT ON TABLE ROUT_ROUT_USAGE_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROUT_SEQ_USAGE_S
        ( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          SEQUENCE_CATALOG,    SEQUENCE_SCHEMA,    SEQUENCE_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
          SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_SEQUENCE_USAGE;

GRANT SELECT ON TABLE ROUT_SEQ_USAGE_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROUTINE_COL_USAGE
        ( SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          ROUTINE_CATALOG,    ROUTINE_SCHEMA,     ROUTINE_NAME,
          TABLE_CATALOG,      TABLE_SCHEMA,       TABLE_NAME,
          COLUMN_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
          ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
          TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          COLUMN_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_COLUMN_USAGE;

GRANT SELECT ON TABLE ROUTINE_COL_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

```
CREATE VIEW ROUTINE_PER_USAGE
        ( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          ROUTINE_CATALOG,     ROUTINE_SCHEMA,     ROUTINE_NAME,
          TABLE_CATALOG,       TABLE_SCHEMA,       TABLE_NAME,
          PERIOD_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           PERIOD_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_PERIOD_USAGE;

GRANT SELECT ON TABLE ROUTINE_PER_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROUT_TABLE_USAGE
        ( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          ROUTINE_CATALOG,     ROUTINE_SCHEMA,     ROUTINE_NAME,
          TABLE_CATALOG,       TABLE_SCHEMA,       TABLE_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
           ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
    FROM INFORMATION_SCHEMA.ROUTINE_TABLE_USAGE;

GRANT SELECT ON TABLE ROUT_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW ROUTINES_S
        ( SPECIFIC_CATALOG,    SPECIFIC_SCHEMA,    SPECIFIC_NAME,
          ROUTINE_CATALOG,     ROUTINE_SCHEMA,     ROUTINE_NAME,
          ROUTINE_TYPE,        MODULE_CATALOG,     MODULE_SCHEMA,
          MODULE_NAME,         UDT_CATALOG,        UDT_SCHEMA,
          UDT_NAME,            DATA_TYPE,          CHAR_MAX_LENGTH,
          CHAR_OCTET_LENGTH,   CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,
          CHARACTER_SET_NAME,  COLLATION_CATALOG,  COLLATION_SCHEMA,
          COLLATION_NAME,      NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,
          NUMERIC_SCALE,       DATETIME_PRECISION, INTERVAL_TYPE,
          INTERVAL_PRECISION,  TYPE_UDT_CATALOG,   TYPE_UDT_SCHEMA,
          TYPE_UDT_NAME,       SCOPE_CATALOG,      SCOPE_SCHEMA,
          SCOPE_NAME,          MAX_CARDINALITY,    DTD_IDENTIFIER,
          ROUTINE_BODY,        ROUTINE_DEFINITION, EXTERNAL_NAME,
          EXTERNAL_LANGUAGE,   PARAMETER_STYLE,    IS_DETERMINISTIC,
          SQL_DATA_ACCESS,     IS_NULL_CALL,       SQL_PATH,
          SCH_LEVEL_ROUTINE,   MAX_DYN_RESLT_SETS, IS_USER_DEFND_CAST,
          IS_IMP_INVOCABLE,    SECURITY_TYPE,      TO_SQL_SPEC_CAT,
          TO_SQL_SPEC_SCHEMA,  TO_SQL_SPEC_NAME,   AS_LOCATOR,
          CREATED,             LAST_ALTERED,       NEW_SAVEPOINT_LVL,
          IS_UDT_DEPENDENT,    RC_FROM_DATA_TYPE,  RC_AS_LOCATOR,
          RC_CHAR_MAX_LENGTH,  RC_CHAR_OCT_LENGTH, RC_CHAR_SET_CAT,
          RC_CHAR_SET_SCHEMA,  RC_CHAR_SET_NAME,   RC_COLLATION_CAT,
          RC_COLLATION_SCH,    RC_COLLATION_NAME,  RC_NUM_PREC,
          RC_NUMERIC_RADIX,    RC_NUMERIC_SCALE,   RC_DATETIME_PREC,
          RC_INTERVAL_TYPE,    RC_INTERVAL_PREC,   RC_TYPE_UDT_CAT,
          RC_TYPE_UDT_SCHEMA,  RC_TYPE_UDT_NAME,   RC_SCOPE_CATALOG,
```

```
            RC_SCOPE_SCHEMA,     RC_SCOPE_NAME,        RC_MAX_CARDINALITY,
            RC_DTD_IDENTIFIER,
            DECLARED_DATA_TYPE, DECLARED_NUM_PREC,    DECLARED_NUM_SCALE,
            RC_FROM_DEC_DTYPE,  RC_FROM_DEC_PREC,     RC_FROM_DEC_SCALE,
            RET_ONLY_PASS_THRU, DESCRIBE_CATALOG,     DESCRIBE_SCHEMA,
            DESCRIBE_NAME,      START_CATALOG,        START_SCHEMA,
            START_NAME,         FULFILL_CATALOG,      FULFILL_SCHEMA,
            FULFILL_NAME,       FINISH_CATALOG,       FINISH_SCHEMA,
            FINISH_NAME ) AS
    SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
            ROUTINE_TYPE, MODULE_CATALOG, MODULE_SCHEMA,
            MODULE_NAME, UDT_CATALOG, UDT_SCHEMA,
            UDT_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
            CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
            COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
            NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
            INTERVAL_PRECISION, TYPE_UDT_CATALOG, TYPE_UDT_SCHEMA,
            TYPE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
            SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
            ROUTINE_BODY, ROUTINE_DEFINITION, EXTERNAL_NAME,
            EXTERNAL_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
            SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
            SCHEMA_LEVEL_ROUTINE, MAX_DYNAMIC_RESULT_SETS, IS_USER_DEFINED_CAST,
            IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE, TO_SQL_SPECIFIC_CATALOG,
            TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, AS_LOCATOR,
            CREATED, LAST_ALTERED, NEW_SAVEPOINT_LEVEL,
            IS_UDT_DEPENDENT, RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
            RESULT_CAST_CHAR_MAX_LENGTH, RESULT_CAST_CHAR_OCTET_LENGTH,
                RESULT_CAST_CHAR_SET_CATALOG,
            RESULT_CAST_CHAR_SET_SCHEMA, RESULT_CAST_CHARACTER_SET_NAME,
                RESULT_CAST_COLLATION_CATALOG,
            RESULT_CAST_COLLATION_SCHEMA, RESULT_CAST_COLLATION_NAME,
                RESULT_CAST_NUMERIC_PRECISION,
            RESULT_CAST_NUMERIC_RADIX, RESULT_CAST_NUMERIC_SCALE,
                RESULT_CAST_DATETIME_PRECISION,
            RESULT_CAST_INTERVAL_TYPE, RESULT_CAST_INTERVAL_PRECISION,
                RESULT_CAST_TYPE_UDT_CATALOG,
            RESULT_CAST_TYPE_UDT_SCHEMA, RESULT_CAST_TYPE_UDT_NAME,
                RESULT_CAST_SCOPE_CATALOG,
            RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,
                RESULT_CAST_MAX_CARDINALITY,
            RESULT_CAST_DTD_IDENTIFIER,
            DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE,
            RESULT_CAST_FROM_DECLARED_DATA_TYPE,
            RESULT_CAST_FROM_DECLARED_NUMERIC_PRECISION,
            RESULT_CAST_FROM_DECLARED_NUMERIC_SCALE,
            RETURNS_ONLY_PASS_THROUGH,
            DESCRIBE_PROCEDURE_SPECIFIC_CATALOG,
            DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA,
            DESCRIBE_PROCEDURE_SPECIFIC_NAME,
            START_PROCEDURE_SPECIFIC_CATALOG,
            START_PROCEDURE_SPECIFIC_SCHEMA,
            START_PROCEDURE_SPECIFIC_NAME,
            FULFILL_PROCEDURE_SPECIFIC_CATALOG,
```

```
            FULFILL_PROCEDURE_SPECIFIC_SCHEMA,
            FULFILL_PROCEDURE_SPECIFIC_NAME,
            FINISH_PROCEDURE_SPECIFIC_CATALOG,
            FINISH_PROCEDURE_SPECIFIC_SCHEMA,
            FINISH_PROCEDURE_SPECIFIC_NAME
     FROM INFORMATION_SCHEMA.ROUTINES;

GRANT SELECT ON TABLE ROUTINES_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW SCHEMATA_S
        ( CATALOG_NAME,        SCHEMA_NAME,        SCHEMA_OWNER,
          DEF_CHAR_SET_CAT,  DEF_CHAR_SET_SCH,  DEF_CHAR_SET_NAME,
          SQL_PATH ) AS
     SELECT CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER,
          DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
          DEFAULT_CHARACTER_SET_NAME, SQL_PATH
     FROM INFORMATION_SCHEMA.SCHEMATA;

GRANT SELECT ON TABLE SCHEMATA_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW SEQUENCES_S
        ( SEQUENCE_CATALOG,   SEQUENCE_SCHEMA,    SEQUENCE_NAME,
          DATA_TYPE,          NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,
          NUMERIC_SCALE,      START_VALUE,        MINIMUM_VALUE,
          MAXIMUM_VALUE,      INCREMENT,          CYCLE_OPTION,
          DECLARED_DATA_TYPE, DECLARED_NUM_PREC,  DECLARED_NUM_SCALE) AS
     SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME,
          DATA_TYPE, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
          NUMERIC_SCALE, START_VALUE, MINIMUM_VALUE,
          MAXIMUM_VALUE, INCREMENT, CYCLE_OPTION,
          DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
          DECLARED_NUMERIC_SCALE
     FROM INFORMATION_SCHEMA.SEQUENCES;

GRANT SELECT ON TABLE SEQUENCES_S
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW SQL_IMPL_INFO
        ( IMPL_INFO_ID,      IMPL_INFO_NAME,     INTEGER_VALUE,
          CHARACTER_VALUE,   COMMENTS ) AS
     SELECT IMPLEMENTATION_INFO_ID, IMPLEMENTATION_INFO_NAME, INTEGER_VALUE,
          CHARACTER_VALUE, COMMENTS
     FROM INFORMATION_SCHEMA.SQL_IMPLEMENTATION_INFO;

GRANT SELECT ON TABLE SQL_IMPL_INFO
     TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TABLE_METHOD_PRIVS
        ( GRANTOR,            GRANTEE,            TABLE_CATALOG,
          TABLE_SCHEMA,       TABLE_NAME,         SPECIFIC_CATALOG,
          SPECIFIC_SCHEMA,    SPECIFIC_NAME,       IS_GRANTABLE ) AS
```

```
       SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
              TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
              SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
       FROM INFORMATION_SCHEMA.TABLE_METHOD_PRIVILEGES;

GRANT SELECT ON TABLE TABLE_METHOD_PRIVS
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TABLES_S
       ( TABLE_CATALOG,      TABLE_SCHEMA,       TABLE_NAME,
         TABLE_TYPE,         SELF_REF_COL_NAME,  REF_GENERATION,
         UDT_CATALOG,        UDT_SCHEMA,         UDT_NAME,
         IS_INSERTABLE_INTO, IS_TYPED,           COMMIT_ACTION ) AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           TABLE_TYPE, SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
           USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME,
           IS_INSERTABLE_INTO, IS_TYPED, COMMIT_ACTION
    FROM INFORMATION_SCHEMA.TABLES;

GRANT SELECT ON TABLE TABLES_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRANSLATIONS_S
       ( TRANS_CATALOG,      TRANSLATION_SCHEMA, TRANSLATION_NAME,
         SRC_CHAR_SET_CAT,   SRC_CHAR_SET_SCH,   SRC_CHAR_SET_NAME,
         TGT_CHAR_SET_CAT,   TGT_CHAR_SET_SCH,   TGT_CHAR_SET_NAME,
         TRANS_SRC_CATALOG,  TRANS_SRC_SCHEMA,   TRANS_SRC_NAME ) AS
    SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
           SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
           SOURCE_CHARACTER_SET_NAME,
           TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
           TARGET_CHARACTER_SET_NAME,
           TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
           TRANSLATION_SOURCE_NAME
    FROM INFORMATION_SCHEMA.TRANSLATIONS;

GRANT SELECT ON TABLE TRANSLATIONS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_ROUT_USAGE_S
       ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,     TRIGGER_NAME,
         SPECIFIC_CATALOG,   SPECIFIC_SCHEMA,    SPECIFIC_NAME ) AS
    SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
           SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
    FROM INFORMATION_SCHEMA.TRIGGER_ROUTINE_USAGE;

GRANT SELECT ON TABLE TRIG_ROUT_USAGE_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_SEQ_USAGE_S
       ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,     TRIGGER_NAME,
         SEQUENCE_CATALOG,   SEQUENCE_SCHEMA,    SEQUENCE_NAME ) AS
    SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
```

```
            SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
      FROM INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE;

GRANT SELECT ON TABLE TRIG_SEQ_USAGE_S
      TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_UPDATE_COLS
          ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
            EVENT_OBJECT_CAT,   EVENT_OBJECT_SCH,    EVENT_OBJECT_TABLE,
            EVENT_OBJECT_COL ) AS
      SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
            EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
            EVENT_OBJECT_COLUMN
      FROM INFORMATION_SCHEMA.TRIGGERED_UPDATE_COLUMNS;

GRANT SELECT ON TABLE TRIG_UPDATE_COLS
      TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_COLUMN_USAGE
          ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
            TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
            COLUMN_NAME ) AS
      SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
            TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
      FROM INFORMATION_SCHEMA.TRIGGER_COLUMN_USAGE;

GRANT SELECT ON TABLE TRIG_COLUMN_USAGE
      TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_PER_USAGE
          ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
            TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
            PERIOD_NAME ) AS
      SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
            TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            PERIOD_NAME
      FROM INFORMATION_SCHEMA.TRIGGER_PERIOD_USAGE;

GRANT SELECT ON TABLE TRIG_PER_USAGE
      TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIG_TABLE_USAGE
          ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
            TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME ) AS
      SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
            TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
      FROM INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE;

GRANT SELECT ON TABLE TRIG_TABLE_USAGE
      TO PUBLIC WITH GRANT OPTION;


CREATE VIEW TRIGGERS_S
```

```
           ( TRIGGER_CATALOG,    TRIGGER_SCHEMA,      TRIGGER_NAME,
             EVENT_MANIPULATION, EVENT_OBJECT_CAT,    EVENT_OBJECT_SCH,
             EVENT_OBJECT_TABLE, ACTION_ORDER,        ACTION_CONDITION,
             ACTION_STATEMENT,   ACTION_ORIENTATION, ACTION_TIMING,
             ACT_REF_OLD_TABLE,  ACT_REF_NEW_TABLE,  ACT_REF_OLD_ROW,
             ACT_REF_NEW_ROW,    CREATED ) AS
      SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
             EVENT_MANIPULATION, EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
             EVENT_OBJECT_TABLE, ACTION_ORDER, ACTION_CONDITION,
             ACTION_STATEMENT, ACTION_ORIENTATION, ACTION_TIMING,
             ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
             ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW, CREATED
      FROM INFORMATION_SCHEMA.TRIGGERS;

GRANT SELECT ON TABLE TRIGGERS_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW UDT_S
           ( UDT_CATALOG,        UDT_SCHEMA,          UDT_NAME,
             UDT_CATEGORY,       IS_INSTANTIABLE,     IS_FINAL,
             ORDERING_FORM,      ORDERING_CATEGORY,   ORDERING_ROUT_CAT,
             ORDERING_ROUT_SCH,  ORDERING_ROUT_NAME,  REFERENCE_TYPE,
             DATA_TYPE,          CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH,
             CHAR_SET_CATALOG,   CHAR_SET_SCHEMA,     CHARACTER_SET_NAME,
             COLLATION_CATALOG,  COLLATION_SCHEMA,    COLLATION_NAME,
             NUMERIC_PRECISION,  NUMERIC_PREC_RADIX,  NUMERIC_SCALE,
             DATETIME_PRECISION, INTERVAL_TYPE,       INTERVAL_PRECISION,
             SOURCE_DTD_ID,      REF_DTD_IDENTIFIER,  DECLARED_DATA_TYPE,
             DECLARED_NUM_PREC,  DECLARED_NUM_SCALE, MAX_CARDINALITY) AS
      SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME,
             USER_DEFINED_TYPE_CATEGORY, IS_INSTANTIABLE, IS_FINAL,
             ORDERING_FORM, ORDERING_CATEGORY, ORDERING_ROUTINE_CATALOG,
             ORDERING_ROUTINE_SCHEMA, ORDERING_ROUTINE_NAME, REFERENCE_TYPE,
             DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
             CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
             COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
             NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
             DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
             SOURCE_DTD_IDENTIFIER, REF_DTD_IDENTIFIER,
             DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
             DECLARED_NUMERIC_SCALE, MAXIMUM_CARDINALITY
      FROM INFORMATION_SCHEMA.USER_DEFINED_TYPES;

GRANT SELECT ON TABLE UDT_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW VIEWS_S
           ( TABLE_CATALOG,      TABLE_SCHEMA,        TABLE_NAME,
             VIEW_DEFINITION,    CHECK_OPTION,        IS_UPDATABLE,
             IS_INSERTABLE_INTO, IS_TRIG_UPDATABLE,   IS_TRIG_DELETABLE,
             IS_TRIG_INS_INTO) AS
      SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
             VIEW_DEFINITION, CHECK_OPTION, IS_UPDATABLE,
             INSERTABLE_INTO, IS_TRIGGER_UPDATABLE,
             IS_TRIGGER_DELETABLE, IS_TRIGGER_INSERTABLE_INTO
```

```
    FROM INFORMATION_SCHEMA.VIEWS;

GRANT SELECT ON TABLE VIEWS_S
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.

2) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS_S.

3) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.COL_DOMAINS_USAGE.

4) Without Feature F341, "Usage tables", conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.

5) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS.

6) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.COL_DOMAIN_USAGE.

7) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_COL_USAGE.

8) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.CONST_TABLE_USAGE.

9) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.KEY_COLUMN_USAGE_S.

10) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINE_COL_USAGE.

11) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_TABLE_USAGE.

12) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_ROUT_USAGE_S.

13) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTR_ROUT_USE_S.

14) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.

15) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.

16) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.

17) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRIG_SEQ_USAGE_S.

18) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COL_COL_USAGE.

19) Without Feature F502, "Enhanced documentation tables", conforming SQL language shall not reference INFORMATION_SCHEMA.SQL_IMPL_INFO.

20) Without Feature F690, "Collation support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.COLLATIONS_S.

21) Without Feature F695, "Translation support", conforming SQL language shall not reference INFORMA-TION_SCHEMA.TRANSLATIONS_S.

22) Without Feature F696, "Additional translation documentation", conforming SQL language shall not reference TRANSLATIONS_S.TRANS_SRC_CATALOG, TRANSLATIONS_S.TRANS_SRC_SCHEMA, or TRANSLATIONS_S.TRANS_SRC_NAME.

23) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.ATTRIBUTES_S.

24) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPECS.

25) Without Feature S023, "Basic structured types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.METHOD_SPEC_PARAMS.

26) Without Feature S024, "Enhanced structured types", conforming SQL language shall not reference INFORMATION_SCHEMA.TABLE_METHOD_PRIVS.

27) Without Feature S024, "Enhanced structured types", conforming SQL language shall not reference INFORMATION_SCHEMA.ROL_TAB_METH_GRNTS.

28) Without Feature S041, "Basic reference types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.REFERENCED_TYPES_S.

29) Without Feature S091, "Basic array support" or Feature S271, "Basic multiset support", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES_S.

30) Without Feature S401, "Distinct types based on array types", conforming SQL language shall not reference INFORMATION_SCHEMA.UDT_S.MAX_CARDINALITY.

31) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPEC.CREATED.

32) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.CREATED.

33) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.LAST_ALTERED.

34) Without Feature T011, "Timestamp in Information Schema", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS_S.CREATED.

35) Without Feature T051, "Row types", conforming SQL language shall not reference INFORMA-TION_SCHEMA.FIELDS_S.

36) Without Feature T111, "Updatable joins, unions, and columns", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS_S.IS_UPDATABLE.

37) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMATION_SCHEMA.COL_COL_USAGE.

38) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS_S.IS_GENERATED

39) Without Feature T175, "Generated columns", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS_S.GENERATION_EXPR.

40) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUT_SEQ_USAGE_S.

41) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference INFORMATION_SCHEMA.SEQUENCES_S.

42) Without Feature T176, "Sequence generator support", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGER_SEQ_USAGE_S.

43) Without Feature T180, "System-versioned tables", conforming SQL language shall not reference any of the columns COLUMNS_S.IS_SYSPER_START, COLUMNS_S.IS_SYSPER_END, and COLUMNS_S.SYSPER_TSTMP_GEN.

44) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_UPDATE_COLS

45) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference the INFORMATION_SCHEMA.TRIG_TABLE_USAGE view.

46) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_ROUT_USAGE_S.

47) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_SEQ_USAGE_S.

48) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIGGERS_S.

49) Without Feature T211, "Basic trigger capability", conforming SQL language shall not reference INFORMATION_SCHEMA.TRIG_COLUMN_USAGE.

50) Without Feature T213, "INSTEAD OF triggers", conforming SQL language shall not reference any of VIEWS_S.IS_TRIG_UPDATABLE, VIEWS_S.IS_TRIG_DELETABLE, VIEWS_S.IS_TRIG_INS_INTO.

51) Without Feature T272, "Enhanced savepoint management", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES_S.NEW_SAVEPOINT_LEVEL.

52) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMATION_SCHEMA.ADMIN_ROLE_AUTHS.

53) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_ROUT_GRANTS.

54) Without Feature T331, "Basic roles", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_TAB_METH_GRNTS.

55) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of ATTRIBUTES_S.DECLARED_DATA_TYPE, ATTRIBUTES_S.DEC_NUMERIC_PREC and ATTRIBUTES_S.DEC_NUM_SCALE.

56) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of COLUMNS_S.DECLARED_DATA_TYPE, COLUMNS_S.DEC_NUMERIC_PREC and COLUMNS_S.DEC_NUM_SCALE.

57) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of DOMAINS_S.DECLARED_DATA_TYPE, DOMAINS_S.DEC_NUMERIC_PREC and DOMAINS_S.DEC_NUM_SCALE.

58) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of ELEMENT_TYPES_S.DECLARED_DATA_TYPE, ELEMENT_TYPES_S.DEC_NUMERIC_PREC and ELEMENT_TYPES_S.DEC_NUM_SCALE.

59) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of FIELDS_S.DECLARED_DATA_TYPE, FIELDS_S.DEC_NUMERIC_PREC and FIELDS_S.DEC_NUM_SCALE.

60) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of METHOD_SPECS.DECLARED_DATA_TYPE, METHOD_SPECS.DEC_NUMERIC_PREC and METHOD_SPECS.DEC_NUM_SCALE.

61) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of METHOD_SPEC_PARAMS.DECLARED_DATA_TYPE, METHOD_SPEC_PARAMS.DEC_NUMERIC_PREC and METHOD_SPEC_PARAMS.DEC_NUM_SCALE.

62) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of PARAMETERS_S.DECLARED_DATA_TYPE, PARAMETERS_S.DEC_NUMERIC_PREC and PARAMETERS_S.DEC_NUM_SCALE.

63) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of REFERENCED_TYPES_S.DECLARED_DATA_TYPE, REFERENCED_TYPES_S.DEC_NUMERIC_PREC and REFERENCED_TYPES_S.DEC_NUM_SCALE.

64) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of ROUTINES_S.DECLARED_DATA_TYPE, ROUTINES_S.DEC_NUMERIC_PREC and ROUTINES_S.DEC_NUM_SCALE.

65) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of SEQUENCES_S.DECLARED_DATA_TYPE, SEQUENCES_S.DEC_NUMERIC_PREC and SEQUENCES_S.DEC_NUM_SCALE.

66) Without Feature T322, "Declared data type attributes", conforming SQL language shall not reference any of UDT_S.DECLARED_DATA_TYPE, UDT_S.DEC_NUMERIC_PREC and UDT_S.DEC_NUM_SCALE.

67) Without Feature T522, "Default values for IN parameters of SQL-invoked procedures", Feature T523, "Default values for INOUT parameters of SQL-invoked procedures", or Feature T525, "Default values for parameters of SQL-invoked functions", conforming SQL language shall not reference INFORMA-TION_SCHEMA.PARAMETERS_S.PARAMETER_DEFAULT.

68) Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference any of INFORMATION_SCHEMA.PARAMETERS_S.TABLE_SEMANTICS, INFORMA-

TION_SCHEMA.PARAMETERS_S.IS_PRUNABLE, or INFORMATION_SCHEMA.PARAME-
TERS_S.HAS_PASS_THRU_COLS.

69) Without Feature B204, "PRUNE WHEN EMPTY", conforming SQL language shall not reference
INFORMATION_SCHEMA.PARAMETERS_S.IS_PRUNABLE.

70) Without Feature B205, "Pass-through columns", conforming SQL language shall not reference INFOR-
MATION_SCHEMA.PARAMETERS_S.HAS_PASS_THRU_COLS.

71) Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference
INFORMATION_SCHEMA.PRIVATE_PARAMS_S.

72) Without Feature B200, "Polymorphic table functions", conforming SQL language shall not reference any
of INFORMATION_SCHEMA.ROUTINES_S. RET_ONLY_PASS_THRU, INFORMA-
TION_SCHEMA.ROUTINES_S.DESCRIBE_CATALOG, INFORMATION_SCHEMA.ROU-
TINES_S.DESCRIBE_SCHEMA, INFORMATION_SCHEMA.ROUTINES_S.DESCRIBE_NAME,
INFORMATION_SCHEMA.ROUTINES_S.START_CATALOG, INFORMATION_SCHEMA.ROU-
TINES_S.START_SCHEMA, INFORMATION_SCHEMA.ROUTINES_S.START_NAME, INFORMA-
TION_SCHEMA.ROUTINES_S.FULFILL_CATALOG, INFORMATION_SCHEMA.ROUTINES_S.FUL-
FILL_SCHEMA, INFORMATION_SCHEMA.ROUTINES_S.FULFILL_NAME, INFORMA-
TION_SCHEMA.ROUTINES_S.FINISH_CATALOG, INFORMATION_SCHEMA.ROUTINES_S.FIN-
ISH_SCHEMA, or INFORMATION_SCHEMA.ROUTINES_S.FINISH_NAME.

73) Without Feature B205, "Pass-through columns", conforming SQL language shall not reference INFOR-
MATION_SCHEMA.ROUTINES_S.RET_ONLY_PASS_THRU.

(Blank page)

6 Definition Schema

This Clause is modified by Clause 9, "Definition Schema", in ISO/IEC 9075-3.
This Clause is modified by Clause 20, "Definition Schema", in ISO/IEC 9075-4.
This Clause is modified by Clause 25, "Definition Schema", in ISO/IEC 9075-9.
This Clause is modified by Clause 15, "Definition Schema", in ISO/IEC 9075-10.
This Clause is modified by Clause 15, "Definition Schema", in ISO/IEC 9075-13.
This Clause is modified by Clause 22, "Definition Schema", in ISO/IEC 9075-14.

6.1   DEFINITION_SCHEMA Schema

Function

Create the schema that is to contain the base tables that underlie the Information Schema

Definition

```
CREATE SCHEMA DEFINITION_SCHEMA
    AUTHORIZATION DEFINITION_SCHEMA
```

Description

None.

6.2   EQUAL_KEY_DEGREES assertion

Function

The assertion EQUAL_KEY_DEGREES ensures that every foreign key is of the same degree as the corresponding unique constraint.

Definition

```
CREATE ASSERTION EQUAL_KEY_DEGREES
    CHECK
      ( NOT EXISTS
        ( SELECT *
          FROM ( SELECT COUNT ( DISTINCT FK.COLUMN_NAME ),
                        COUNT ( DISTINCT PK.COLUMN_NAME )
                FROM KEY_COLUMN_USAGE AS FK,
                     REFERENTIAL_CONSTRAINTS AS RF,
                     KEY_COLUMN_USAGE AS PK
                WHERE ( FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA,
                        FK.CONSTRAINT_NAME ) =
                        ( RF.CONSTRAINT_CATALOG, RF.CONSTRAINT_SCHEMA,
                          RF.CONSTRAINT_NAME )
                  AND
                        ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                          PK.CONSTRAINT_NAME ) =
                        ( RF.UNIQUE_CONSTRAINT_CATALOG, RF.UNIQUE_CONSTRAINT_SCHEMA,
                          RF.UNIQUE_CONSTRAINT_NAME )
                GROUP BY
                   RF.CONSTRAINT_CATALOG, RF.CONSTRAINT_SCHEMA, RF.CONSTRAINT_NAME )
                   AS R ( FK_DEGREE, PK_DEGREE )
          WHERE FK_DEGREE <> PK_DEGREE ) )
```

6.3   KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion

Function

The assertion KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 ensures that every unique or primary key constraint has at least one unique column and that every referential constraint has at least one referencing column.

Definition

```
CREATE ASSERTION KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1
    CHECK
      ( NOT EXISTS
        ( SELECT *
          FROM TABLE_CONSTRAINTS
            FULL OUTER JOIN
              KEY_COLUMN_USAGE
            USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
          WHERE COLUMN_NAME IS NULL
            AND
              CONSTRAINT_TYPE IN
              ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) )
```

6.4 UNIQUE_CONSTRAINT_NAME assertion

Function

The UNIQUE_CONSTRAINT_NAME assertion ensures that the same combination of <schema name> and
<constraint name> is not used by more than one constraint.

> NOTE 7 — The UNIQUE_CONSTRAINT_NAME assertion avoids the need for separate checks on DOMAINS, TABLE_CON-
> STRAINTS, and ASSERTIONS.

Definition

```
CREATE ASSERTION UNIQUE_CONSTRAINT_NAME
    CHECK ( 1 =
      ( SELECT MAX ( OCCURRENCES )
        FROM ( SELECT COUNT (*) AS OCCURRENCES
               FROM ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                      FROM DOMAIN_CONSTRAINTS
                    UNION ALL
                      SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                      FROM TABLE_CONSTRAINTS
                    UNION ALL
                      SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                      FROM ASSERTIONS )
               GROUP BY
                   CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) ) )
```

6.5 ASSERTIONS base table

Function

The ASSERTIONS table has one row for each assertion. It effectively contains a representation of the assertion descriptors.

Definition

```
CREATE TABLE ASSERTIONS
    (
    CONSTRAINT_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_DEFERRABLE         INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT ASSERTIONS_IS_DEFERRABLE_NOT_NULL
        NOT NULL,
    INITIALLY_DEFERRED    INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT ASSERTIONS_INITIALLY_DEFERRED_NOT_NULL
        NOT NULL,

    CONSTRAINT ASSERTIONS_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

    CONSTRAINT ASSERTIONS_FOREIGN_KEY_CHECK_CONSTRAINTS
      FOREIGN KEY (CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
        REFERENCES CHECK_CONSTRAINTS,

    CONSTRAINT ASSERTIONS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT ASSERTIONS_DEFERRED_CHECK
      CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN
              ( VALUES ( 'NO',  'NO' ),
                      ( 'YES', 'NO' ),
                      ( 'YES', 'YES' ) ) )
    )
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion being described.

2) The values of IS_DEFERRABLE have the following meanings:

YES	The assertion is deferrable.
NO	The assertion is not deferrable.

3) The values of INITIALLY_DEFERRED have the following meanings:

YES	The assertion is initially deferred.
NO	The assertion is initially immediate.

6.6 ATTRIBUTES base table

Function

The ATTRIBUTES base table contains one row for each attribute. It effectively contains a representation of the attribute descriptors.

Definition

```
CREATE TABLE ATTRIBUTES (
    UDT_CATALOG                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    UDT_SCHEMA                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    UDT_NAME                             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ATTRIBUTE_NAME                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION                     INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT ORDINAL_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT ATTRIBUTES_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT ATTRIBUTES_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM ATTRIBUTES
                          GROUP BY UDT_CATALOG, UDT_SCHEMA, UDT_NAME ) ),
    DTD_IDENTIFIER                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ATTRIBUTE_DEFAULT                    INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_DERIVED_REFERENCE_ATTRIBUTE       INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_NOT_NULL
        NOT NULL,

    CONSTRAINT ATTRIBUTES_PRIMARY_KEY
      PRIMARY KEY ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME, ATTRIBUTE_NAME ),

    CONSTRAINT ATTRIBUTES_UNIQUE
      UNIQUE ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME, ORDINAL_POSITION ),

    CONSTRAINT ATTRIBUTES_CHECK_DATA_TYPE
      CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
              'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN

                ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                         OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) ),

    CONSTRAINT ATTRIBUTES_UDT_IS_STRUCTURED_CHECK
      CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME ) IN
              ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                       USER_DEFINED_TYPE_NAME
                FROM USER_DEFINED_TYPES
                WHERE USER_DEFINED_TYPE_CATEGORY = 'STRUCTURED' ) )
```

)

Description

1) The values of UDT_CATALOG, UDT_SCHEMA, and UDT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type containing the attribute being described.

2) The value of ATTRIBUTE_NAME is the name of the attribute being described.

3) The values of UDT_CATALOG, UDT_SCHEMA, UDT_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the attribute.

4) The value of ORDINAL_POSITION is the ordinal position of the attribute in the user-defined type.

5) The value of ATTRIBUTE_DEFAULT is null if the attribute being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of ATTRIBUTE_DEFAULT is "TRUNCATED". Otherwise, the value of ATTRIBUTE_DEFAULT is a character representation of the default value for the column that obeys the rules specified for <default option> in Subclause 11.5, "<default clause>".

   NOTE 8 — "TRUNCATED" is different from other values like CURRENT_USER or CURRENT_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.

6) The values of IS_DERIVED_REFERENCE_ATTRIBUTE have the following meanings:

YES	The attribute is used in the definition of a derived representation for the reference type corresponding to the structured type to which the attribute belongs.
NO	The attribute is not used in the definition of a derived representation for the reference type corresponding to the structured type to which the attribute belongs.

6.7   AUTHORIZATIONS base table

Function

The AUTHORIZATIONS table has one row for each <role name> and one row for each <user identifier> referenced in the Information Schema. These are the <role name>s and <user identifier>s that may grant privileges as well as those that may create a schema, or currently own a schema created through a <schema definition>.

Definition

```
CREATE TABLE AUTHORIZATIONS (
    AUTHORIZATION_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT AUTHORIZATIONS_AUTHORIZATION_NAME_VALUES
        CHECK ( AUTHORIZATION_NAME NOT IN ( '_SYSTEM', 'PUBLIC' ) ),
    AUTHORIZATION_TYPE      INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT AUTHORIZATIONS_AUTHORIZATION_TYPE_NOT_NULL
        NOT NULL
      CONSTRAINT AUTHORIZATIONS_AUTHORIZATION_TYPE_CHECK
        CHECK ( AUTHORIZATION_TYPE IN ( 'USER', 'ROLE' ) ),

    CONSTRAINT AUTHORIZATIONS_PRIMARY_KEY
      PRIMARY KEY (AUTHORIZATION_NAME)
    )
```

Description

1)   The values of AUTHORIZATION_TYPE have the following meanings:

USER	The value of AUTHORIZATION_NAME is a known <user identifier>.
ROLE	The value of AUTHORIZATION_NAME is a <role name> defined by a <role definition>.

6.8 CATALOG_NAME base table

Function

The CATALOG_NAME table identifies the catalog or catalogs that are described by this Definition Schema.

Definition

```
CREATE TABLE CATALOG_NAME (
    CATALOG_NAME                    INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CATALOG_NAME_PRIMARY_KEY
      PRIMARY KEY ( CATALOG_NAME )
    )
```

Description

1) The values of CATALOG_NAME are the names of the catalogs that are described by this Definition Schema.

Table Population

1) There is one row in this table for the catalog in which this Definition Schema exists. In that row CATA-LOG_NAME, is the name of that catalog.

6.9 CHARACTER_ENCODING_FORMS base table

Function

The CHARACTER_ENCODING_FORMS table has one row for each character encoding form descriptor.

Definition

```
CREATE TABLE CHARACTER_ENCODING_FORMS (
    CHARACTER_REPERTOIRE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_ENCODING_FORM_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CHARACTER_ENCODING_FORMS_PRIMARY_KEY
      PRIMARY KEY ( CHARACTER_ENCODING_FORM_NAME, CHARACTER_REPERTOIRE_NAME ),

    CONSTRAINT CHARACTER_ENCODING_FORMS_FOREIGN_KEY_CHARACTER_REPERTOIRES
      FOREIGN KEY ( CHARACTER_REPERTOIRE_NAME )
        REFERENCES CHARACTER_REPERTOIRES
  )
```

Description

1) The value of CHARACTER_ENCODING_FORM_NAME is the name of the character encoding form being described.

2) The value of CHARACTER_REPERTOIRE_NAME is the name of the character repertoire to which this character encoding form applies.

Table Population

1) There is one row in this table for the character encoding form SQL_TEXT. In that row, CHARAC-TER_ENCODING_FORM_NAME and CHARACTER_REPERTOIRE_NAME are 'SQL_TEXT' and 'SQL_TEXT', respectively.

2) There is one row in this table for the character encoding form SQL_IDENTIFIER. In that row, CHARAC-TER_ENCODING_FORM_NAME and CHARACTER_REPERTOIRE_NAME are 'SQL_IDENTIFIER' and 'SQL_IDENTIFIER', respectively.

3) There is one row in this table for the character encoding form SQL_CHARACTER. In that row, CHAR-ACTER_ENCODING_FORM_NAME and CHARACTER_REPERTOIRE_NAME are 'SQL_CHARAC-TER' and 'SQL_CHARACTER', respectively.

4) If the implementation supports one or more of the standard-defined character encoding forms GRAPHIC_IRV, LATIN1, ISO8BIT, UTF32, UTF16, or UTF8, then there is one row in this table for each of those character encoding forms supported. In such a row, CHARACTER_ENCODING_FORM_NAME and CHARACTER_REPERTOIRE_NAME have, respectively, the following values.

Case:

    a)   GRAPHIC_IRV: 'GRAPHIC_IRV' and 'GRAPHIC_IRV'.

    b)   LATIN1: 'LATIN1' and 'LATIN1'.

    c)   ISO8BIT: 'ISO8BIT' and 'ISO8BIT'.

    d)   UTF32: 'UTF32' and 'UCS'.

    e)   UTF16: 'UCS16' and 'UCS'.

    f)   UTF8: 'UTF8' and 'UCS'.

5)   There is one row in this table for each implementation-defined character encoding form. The contents of that row are implementation-defined.

6.10   CHARACTER_REPERTOIRES base table

Function

The CHARACTER_REPERTOIRES table has one row for each character repertoire descriptor.

Definition

```
CREATE TABLE CHARACTER_REPERTOIRES (
    CHARACTER_REPERTOIRE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATION_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATION_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_SCHEMA_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATION_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_REPERTOIRES_DEFAULT_COLLATION_NAME_NOT_NULL
        NOT NULL,

    CONSTRAINT CHARACTER_REPERTOIRES_PRIMARY_KEY
      PRIMARY KEY ( CHARACTER_REPERTOIRE_NAME ),

    CONSTRAINT CHARACTER_REPERTOIRES_FOREIGN_KEY_COLLATIONS
      FOREIGN KEY ( DEFAULT_COLLATION_CATALOG, DEFAULT_COLLATION_SCHEMA,
                    DEFAULT_COLLATION_NAME )
        REFERENCES COLLATIONS
    )
```

Description

1) The value of DEFAULT_COLLATION_CATALOG, DEFAULT_COLLATION_SCHEMA, and DEFAULT_COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default collation of the character repertoire being described.

2) The value of CHARACTER_REPERTOIRE_NAME is the name of the character repertoire being described.

Table Population

1) There is one row in this table for the character repertoire SQL_TEXT. In that row:

   a) CHARACTER_REPERTOIRE_NAME is 'SQL_TEXT'.

   b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

2) There is one row in this table for the character repertoire SQL_IDENTIFIER. In that row:

    a)   CHARACTER_REPERTOIRE_NAME is 'SQL_IDENTIFIER'.

    b)   DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COL-LATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

3)   There is one row in this table for the character repertoire SQL_CHARACTER. In that row:

    a)   CHARACTER_REPERTOIRE_NAME is 'SQL_CHARACTER'.

    b)   DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COL-LATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

4)   If the implementation supports one or more of the standard-defined character repertoires GRAPHIC_IRV, LATIN1, ISO8BIT, or UCS, then there is one row in this table for each of those character repertoires supported. In such a row:

    a)   DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA are the name of the catalog and 'INFORMATION_SCHEMA' respectively.

    b)   CHARACTER_REPERTOIRE_NAME and DEFAULT_COLLATE_NAME have, respectively, the following values.

       Case:

       i)   GRAPHIC_IRV: 'GRAPHIC_IRV' and 'GRAPHIC_IRV'.

       ii)   LATIN1: 'LATIN1' and 'LATIN1'.

       iii) ISO8BIT: 'ISO8BIT' and 'ISO8BIT'.

       iv) UTF32: 'UTF32' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

       v)   UTF16: 'UTF16' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

       vi) UTF8: 'UTF8' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

5)   There is one row in this table for each implementation-defined character repertoire. The contents of that row are implementation-defined.

6.11 CHARACTER_SETS base table

Function

The CHARACTER_SETS table has one row for each character set descriptor.

Definition

```
CREATE TABLE CHARACTER_SETS (
    CHARACTER_SET_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_REPERTOIRE         INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_SETS_CHARACTER_REPERTOIRE_NOT_NULL
        NOT NULL,
    FORM_OF_USE                  INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_SETS_FORM_OF_USE_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_SCHEMA_NOT_NULL
        NOT NULL,
    DEFAULT_COLLATE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_NAME_NOT_NULL
        NOT NULL,

    CONSTRAINT CHARACTER_SETS_PRIMARY_KEY
      PRIMARY KEY ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ),

    CONSTRAINT CHARACTER_SETS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT CHARACTER_SETS_FOREIGN_KEY_CHARACTER_ENCODING_FORMS
      FOREIGN KEY (FORM_OF_USE, CHARACTER_REPERTOIRE )
        REFERENCES CHARACTER_ENCODING_FORMS,

    CONSTRAINT CHARACTER_SETS_CHECK_REFERENCES_COLLATIONS
      CHECK ( DEFAULT_COLLATE_CATALOG NOT IN
          ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
          ( DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA,
            DEFAULT_COLLATE_NAME ) IN
          ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
            FROM COLLATIONS ) )
```

)

Description

1) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARAC-TER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set being described.

2) The value of CHARACTER_REPERTOIRE is the name of the character repertoire of the character set being described.

3) The value of FORM_OF_USE is the name of the character encoding form used by the character set being described.

4) The values of DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the explicit or implicit default collation for the character set.

Table Population

1) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_TEXT. In that row:

    a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

    b) CHARACTER_REPERTOIRE and FORM_OF_USE are 'SQL_TEXT' and 'SQL_TEXT', respectively.

    c) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COL-LATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

2) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_IDENTIFIER. In that row:

    a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

    b) CHARACTER_REPERTOIRE and FORM_OF_USE are 'SQL_IDENTIFIER' and 'SQL_IDENTIFIER', respectively.

    c) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COL-LATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

3) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_CHARACTER. In that row:

    a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

    b) CHARACTER_REPERTOIRE and FORM_OF_USE are 'SQL_CHARACTER' and 'SQL_CHARAC-TER', respectively.

c) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COL-LATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

4) If the implementation supports one of the standard-defined character sets GRAPHIC_IRV, ASCII_GRAPHIC, LATIN1, ISO8BIT, ASCII_FULL, UTF32, UTF16, or UTF8, then there is one row in this table for each of those character repertoires supported. In such a row:

   a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, DEFAULT_COLLATE_CATA-LOG, DEFAULT_COLLATE_SCHEMA are the name of the catalog, 'INFORMATION_SCHEMA', the name of the catalog, and 'INFORMATION_SCHEMA' respectively.

   b) CHARACTER_SET_NAME, CHARACTER_REPERTOIRE, FORM_OF_USE, and DEFAULT_COLLATE_NAME have, respectively, the following values.

   Case:

   i) GRAPHIC_IRV: 'GRAPHIC_IRV', 'GRAPHIC_IRV', 'GRAPHIC_IRV', and 'GRAPHIC_IRV'.

   ii) ASCII_GRAPHIC: 'GRAPHIC_IRV', 'GRAPHIC_IRV', 'GRAPHIC_IRV', and 'GRAPHIC_IRV'.

   iii) LATIN1: 'LATIN1', 'LATIN1', 'LATIN1' and 'LATIN1'.

   iv) ISO8BIT: 'ISO8BIT', 'ISO8BIT', 'ISO8BIT', and 'ISO8BIT'.

   v) ASCII_FULL: 'ISO8BIT', 'ISO8BIT', 'ISO8BIT', and 'ISO8BIT'.

   vi) UTF32: 'UTF32', 'UCS', 'UCS32' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

   vii) UTF16: 'UCS16', 'UCS', 'UCS16' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

   viii) UTF8: 'UTF8', 'UCS', 'UCS8' and an implementation-defined choice of either 'UCS_BASIC' or 'UNICODE'.

5) There is one row in this table for each implementation-defined character set. The contents of that row are implementation-defined.

6.12 CHECK_COLUMN_USAGE base table

Function

The CHECK_COLUMN_USAGE table has one row for each column identified by a <column reference> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Definition

```
CREATE TABLE CHECK_COLUMN_USAGE (
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CHECK_COLUMN_USAGE_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

    CONSTRAINT CHECK_COLUMN_USAGE_FOREIGN_KEY_CHECK_TABLE_USAGE
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME)
        REFERENCES CHECK_TABLE_USAGE,

    CONSTRAINT CHECK_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
      CHECK ( TABLE_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
          OR
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM COLUMNS ) )
    )
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.

6.13 CHECK_CONSTRAINT_ROUTINE_USAGE base table

Function

The CHECK_CONSTRAINT_ROUTINE_USAGE base table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an <assertion definition>, a <domain constraint>, or a <table constraint definition>.

Definition

```
CREATE TABLE CHECK_CONSTRAINT_ROUTINE_USAGE (
    CONSTRAINT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
      CHECK ( SPECIFIC_CATALOG NOT IN
            ( SELECT CATALOG_NAME
              FROM SCHEMATA )
          OR
            ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
            ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
              FROM ROUTINES ) ),

    CONSTRAINT CHECK_CONSTRAINT_ROUTINE_USAGE_FOREIGN_KEY_CHECK_CONSTRAINTS
      FOREIGN KEY (CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
        REFERENCES CHECK_CONSTRAINTS
    )
```

Description

1) The CHECK_CONSTRAINT_ROUTINE_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an <assertion definition> or in the <check constraint definition> contained in either a <domain constraint> or a <table constraint definition>.

2) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion or check constraint being described.

3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

6.14 CHECK_CONSTRAINTS base table

Function

The CHECK_CONSTRAINTS table has one row for each domain constraint, table check constraint, and assertion.

Definition

```
CREATE TABLE CHECK_CONSTRAINTS (
    CONSTRAINT_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHECK_CLAUSE       INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT CHECK_CONSTRAINTS_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

    CONSTRAINT CHECK_CONSTRAINTS_SOURCE_CHECK
      CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
              ( SELECT *
                FROM ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                       FROM ASSERTIONS
                     UNION
                       SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                       FROM TABLE_CONSTRAINTS
                     UNION
                       SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                       FROM DOMAIN_CONSTRAINTS ) ) )
    )
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) Case:

   a) If the character representation of the <search condition> contained in the <check constraint definition>, <domain constraint>, or <assertion definition> that defined the check constraint being described can be represented without truncation, then the value of CHECK_CLAUSE is that character representation.

   b) Otherwise, the value of CHECK_CLAUSE is the null value.

   NOTE 9 — Any implicit column references that were contained in the <search condition> associated with a <check constraint definition> or an <assertion definition> are replaced by explicit column references in CHECK_CONSTRAINTS.

6.15 CHECK_PERIOD_USAGE base table

Function

The CHECK_PERIOD_USAGE table has one row for each period identified by either SYSTEM_TIME or an <application time period name> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Definition

```
CREATE TABLE CHECK_PERIOD_USAGE (
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PERIOD_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CHECK_PERIOD_USAGE_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

    CONSTRAINT CHECK_PERIOD_USAGE_FOREIGN_KEY_CHECK_TABLE_USAGE
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME)
        REFERENCES CHECK_TABLE_USAGE,

    CONSTRAINT CHECK_PERIOD_USAGE_CHECK_REFERENCES_PERIODS
      CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME FROM SCHEMATA )
            OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME
                FROM PERIODS ) )
)
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and PERIOD_NAME are the catalog name, unqualified schema name, qualified identifier, and period name, respectively, of a period identified by either SYSTEM_TIME or an <application time period name> contained in the <search condition> of the constraint being described.

6.16 CHECK_TABLE_USAGE base table

Function

The CHECK_TABLE_USAGE table has one row for each table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of a check constraint, domain constraint, or assertion.

Definition

```
CREATE TABLE CHECK_TABLE_USAGE
    (
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT CHECK_TABLE_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT CHECK_TABLE_USAGE_FOREIGN_KEY_CHECK_CONSTRAINTS
    FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
      REFERENCES CHECK_CONSTRAINTS,

    CONSTRAINT CHECK_TABLE_USAGE_CHECK_REFERENCES_TABLES
        CHECK ( TABLE_CATALOG NOT IN
                ( SELECT CATALOG_NAME FROM SCHEMATA )
             OR
                ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLES ) ) )
    )
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of the constraint being described.

6.17 COLLATIONS base table

Function

The COLLATIONS table has one row for each character collation descriptor.

Definition

```
CREATE TABLE COLLATIONS (
    COLLATION_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PAD_ATTRIBUTE                 INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT PAD_ATTRIBUT_NOT_NULL
        NOT NULL
      CONSTRAINT COLLATIONS_PAD_ATTRIBUTE_CHECK
        CHECK ( PAD_ATTRIBUTE IN
                ( 'NO PAD', 'PAD SPACE' ) ),
    CHARACTER_REPERTOIRE_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT CHARACTER_REPERTOIRE_NAME_NOT_NULL
        NOT NULL,

    CONSTRAINT COLLATIONS_PRIMARY_KEY
      PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ),

    CONSTRAINT COLLATIONS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT COLLATIONS_FOREIGN_KEY_CHARACTER_REPERTOIRE
      FOREIGN KEY ( CHARACTER_REPERTOIRE_NAME )
        REFERENCES CHARACTER_REPERTOIRES
    )
```

Description

1) The values of COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation being described.

2) The values of PAD_ATTRIBUTE have the following meanings:

NO PAD	The collation being described has the NO PAD characteristic.
PAD SPACE	The collation being described has the PAD SPACE characteristic.

3) The value of CHARACTER_REPERTOIRE_NAME is the name of the charater repertoire to which the collation being described is applicable.

Definition Schema 155

Table Population

1) There is one row in this table for the collation INFORMATION_SCHEMA.SQL_TEXT. In that row:

   a) COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

   b) PAD_ATTRIBUTE is implementation-defined.

   c) CHARACTER_REPERTOIRE_NAME is 'SQL_TEXT'.

2) There is one row in this table for the collation INFORMATION_SCHEMA.SQL_IDENTIFIER. In that row:

   a) COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

   b) PAD_ATTRIBUTE is implementation-defined.

   c) CHARACTER_REPERTOIRE_NAME is 'SQL_IDENTIFIER'.

3) There is one row in this table for the collation INFORMATION_SCHEMA.SQL_CHARACTER. In that row:

   a) COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

   b) PAD_ATTRIBUTE is implementation-defined.

   c) CHARACTER_REPERTOIRE_NAME is 'SQL_CHARACTER'.

4) If the implementation supports one of the standard-defined collations INFORMA-TION_SCHEMA.GRAPHIC_IRV, INFORMATION_SCHEMA.LATIN1, INFORMA-TION_SCHEMA.ISO8BIT, INFORMATION_SCHEMA.UCS_BASIC, or INFORMA-TION_SCHEMA.UNICODE, then there is one row in this table for each of those collations supported. In such a row:

   a) COLLATION_CATALOG and COLLATION_SCHEMA are the name of the catalog and 'INFOR-MATION_SCHEMA' respectively.

   b) PAD_ATTRIBUTE is implementation-defined.

   c) COLLATION_NAME and CHARACTER_REPERTOIRE_NAME have, respectively, the following values.

   Case:

   i) GRAPHIC_IRV: 'GRAPHIC_IRV' and 'GRAPHIC_IRV'.

   ii) LATIN1: 'LATIN1' and 'LATIN1'.

   iii) ISO8BIT: 'ISO8BIT' and 'ISO8BIT'.

   iv) UCS_BASIC: 'UCS_BASIC' and 'UCS'.

   v) UNICODE: 'UNICODE' and 'UCS'.

5)   There is one row in this table for each implementation-defined collation. The contents of that row are implementation-defined.

6.18 COLLATION_CHARACTER_SET_APPLICABILITY base table

Function

The COLLATION_CHARACTER_SET_APPLICABILITY table has one row for each applicability of a collation to a character set.

Definition

```
CREATE TABLE COLLATION_CHARACTER_SET_APPLICABILITY (
    COLLATION_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_PRIMARY_KEY
      PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
                    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ),

    CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_FOREIGN_KEY_COLLATIONS
      FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
        REFERENCES COLLATIONS,

    CONSTRAINT COLLATION_CHARACTER_SET_APPLICABILITY_CHECK_REFERENCES_CHARACTER_SETS
      CHECK ( CHARACTER_SET_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
          OR
            ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ) IN
              ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME
                FROM CHARACTER_SETS ) )
)
```

Description

1) The values of COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation whose applicability is being described.

2) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set to which the collation is applicable.

Table Population

1) There is one row in this table for collation INFORMATION_SCHEMA.SQL_TEXT. In that row:

  a)   COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

  b)   CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

2)   There is one row in this table for collation INFORMATION_SCHEMA.SQL_IDENTIFIER. In that row:

  a)   COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

  b)   CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

3)   There is one row in this table for collation INFORMATION_SCHEMA.SQL_CHARACTER. In that row:

  a)   COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

  b)   CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

4)   If the implementation supports one of the standard-defined collations INFORMA-TION_SCHEMA.GRAPHIC_IRV, INFORMATION_SCHEMA.LATIN1, INFORMA-TION_SCHEMA.ISO8BIT, INFORMATION_SCHEMA.UCS_BASIC, or INFORMA-TION_SCHEMA.UNICODE, then there is one row in this table for each of those collations supported. In such a row:

  a)   COLLATION_CATALOG, COLLATION_SCHEMA, CHARACTER_SET_CATALOG, and CHARACTER_SET_SCHEMA are the name of the catalog and 'INFORMATION_SCHEMA', the name of the catalog, and 'INFORMATION_SCHEMA' respectively.

  b)   COLLATION_NAME and CHARACTER_SET_NAME have, respectively, the following values.

      Case:

      i)   GRAPHIC_IRV: 'GRAPHIC_IRV' and 'GRAPHIC_IRV'.

      ii)   LATIN1: 'LATIN1' and 'LATIN1'.

      iii) ISO8BIT: 'ISO8BIT' and 'ISO8BIT'.

      iv)   UCS_BASIC: 'UCS_BASIC' and 'UCS'.

      v)   UNICODE: 'UNICODE' and 'UCS'.

5)   There is one row in this table for each additional applicability of a collation to a character set. The contents of these rows are implementation-defined.

6.19 COLUMN_COLUMN_USAGE base table

Function

The COLUMN_COLUMN_USAGE table has one row for each case where a generated column depends on a base column.

Definition

```
CREATE TABLE COLUMN_COLUMN_USAGE (
    TABLE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DEPENDENT_COLUMN INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT COLUMN_COLUMN_USAGE_PRIMARY_KEY
      PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                    COLUMN_NAME, DEPENDENT_COLUMN ),

    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_1
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                    COLUMN_NAME ) REFERENCES COLUMNS,

    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_2
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                    DEPENDENT_COLUMN ) REFERENCES COLUMNS
)
```

Description

1) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and DEPENDENT_COLUMN
   are the catalog name, unqualified schema name and qualified identifier, respectively, of a generated column
   GC.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and COLUMN_NAME are the
   catalog name, unqualified schema name and qualified identifier, respectively, of a column on which *GC*
   depends.

6.20   COLUMN_PRIVILEGES base table

Function

The COLUMN_PRIVILEGES table has one row for each column privilege descriptor. It effectively contains a representation of the column privilege descriptors.

Definition

```
CREATE TABLE COLUMN_PRIVILEGES (
    GRANTOR           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE    INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT COLUMN_PRIVILEGE_TYPE_CHECK
        CHECK ( PRIVILEGE_TYPE IN
               ( 'SELECT', 'INSERT', 'UPDATE', 'REFERENCES' ) ),
    IS_GRANTABLE   INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT COLUMN_PRIVILEGE_IS_GRANTABLE_NOT_NULL
        NOT NULL,

    CONSTRAINT COLUMN_PRIVILEGE_PRIMARY_KEY
      PRIMARY KEY ( GRANTOR, GRANTEE,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                    PRIVILEGE_TYPE, COLUMN_NAME ),

    CONSTRAINT COLUMN_PRIVILEGE_FOREIGN_KEY_COLUMNS
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
        REFERENCES COLUMNS,

    CONSTRAINT COLUMN_PRIVILEGES_GRANTOR_CHECK
      CHECK ( GRANTOR = '_SYSTEM'
            OR
              GRANTOR IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) ),

    CONSTRAINT COLUMN_PRIVILEGES_GRANTEE_CHECK
      CHECK ( GRANTEE = 'PUBLIC'
            OR
              GRANTEE IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) )
```

)

Description

1) The value of GRANTOR is the <authorization identifier> of the user or role who granted column privileges, on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME, to the user or role identified by the value of GRANTEE for the column privilege being described, or "_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the column privilege being described is granted.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the column on which the privilege being described was granted.

4) The values of PRIVILEGE_TYPE have the following meanings:

SELECT	The user has SELECT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
INSERT	The user has INSERT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
UPDATE	The user has UPDATE privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
REFER-ENCE	The user has REFERENCES privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.

5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.21 COLUMNS base table

Function

The COLUMNS table has one row for each column. It effectively contains a representation of the column descriptors.

Definition

```
CREATE TABLE COLUMNS (
    TABLE_CATALOG                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME                            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION                      INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT COLUMNS_ORDINAL_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT COLUMNS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT COLUMNS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM COLUMNS
                          GROUP BY
                            TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) ),
    DTD_IDENTIFIER                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_CATALOG                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_DEFAULT                        INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_NULLABLE                           INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT COLUMNS_IS_NULLABLE_NOT_NULL
        NOT NULL,
    IS_SELF_REFERENCING                   INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT COLUMNS_IS_SELF_REFERENCING_NOT_NULL
        NOT NULL,
    IS_IDENTITY                           INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT COLUMNS_IS_IDENTITY_NOT_NULL
        NOT NULL,
    IDENTITY_GENERATION                   INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT COLUMNS_IDENTITY_GENERATION_CHECK
        CHECK ( IDENTITY_GENERATION IN
          ( 'ALWAYS', 'BY DEFAULT' ) ),
    IDENTITY_START                        INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_INCREMENT                    INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MAXIMUM                       INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MINIMUM                       INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_CYCLE                         INFORMATION_SCHEMA.YES_OR_NO,
    IS_GENERATED                          INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT COLUMNS_IS_GENERATED_NOT_NULL
        NOT NULL
      CONSTRAINT COLUMNS_IS_GENERATED_CHECK
        CHECK ( IS_GENERATED IN
            ( 'NEVER', 'ALWAYS' ) ),
```

```
GENERATION_EXPRESSION                     INFORMATION_SCHEMA.CHARACTER_DATA,
IS_SYSTEM_TIME_PERIOD_START               INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT IS_SYSTEM_TIME_PERIOD_START_NOT_NULL
    NOT NULL,
IS_SYSTEM_TIME_PERIOD_END                 INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT IS_SYSTEM_TIME_PERIOD_END_NOT_NULL
    NOT NULL,
SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION   INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT COLUMNS_SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION_CHECK
    CHECK ( SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION IN ( 'ALWAYS' ) ),
IS_UPDATABLE                              INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT COLUMNS_IS_UPDATABLE_NOT_NULL
    NOT NULL,

CONSTRAINT COLUMNS_PRIMARY_KEY
  PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

CONSTRAINT COLUMNS_UNIQUE
  UNIQUE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, ORDINAL_POSITION ),

CONSTRAINT COLUMNS_FOREIGN_KEY_TABLES
  FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
    REFERENCES TABLES,

CONSTRAINT COLUMNS_CHECK_REFERENCES_DOMAIN
  CHECK ( DOMAIN_CATALOG NOT IN
          ( SELECT CATALOG_NAME
            FROM SCHEMATA )
        OR
          ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN
          ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
            FROM DOMAINS ) ),

CONSTRAINT COLUMNS_CHECK_IDENTITY_COMBINATIONS
  CHECK ( ( IS_IDENTITY = 'NO' ) =
        ( ( IDENTITY_GENERATION, IDENTITY_START, IDENTITY_INCREMENT,
            IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE ) IS NULL ) ),

CONSTRAINT COLUMNS_CHECK_GENERATION_COMBINATIONS
  CHECK ( ( IS_GENERATED = 'NEVER' ) =
        ( GENERATION_EXPRESSION IS NULL ) ),

CONSTRAINT COLUMNS_CHECK_DATA_TYPE
  CHECK ( DOMAIN_CATALOG NOT IN
          ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
          ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
            IS NOT NULL
          AND
            ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
              'TABLE', DTD_IDENTIFIER ) NOT IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                     OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR ) )
        OR
          ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
            IS NULL
```

```
        AND
          ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            'TABLE', DTD_IDENTIFIER ) IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                   OBJECT_TYPE, DTD_IDENTIFIER
            FROM DATA_TYPE_DESCRIPTOR ) ) )
  )
```

Description

1) Case:

   a) If a column is described by a column descriptor included in a table descriptor, then the table descriptor and the column descriptor are associated with that column.

   b) If a column is described by a column descriptor included in a view descriptor, then the view descriptor and the corresponding column descriptor of the table of the <query expression> are associated with that column.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the column being described.

3) The value of COLUMN_NAME is the name of the column being described.

4) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the column.

5) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are null if the column being described is not defined using a <domain name>. Otherwise, the values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the domain used by the column being described.

6) The value of ORDINAL_POSITION is the ordinal position of the column in the table.

7) Let DC be the descriptor of the column being described. If DC includes a <default option>, then let DO be that <default option>. The value of COLUMN_DEFAULT is

   Case:

   a) If DC does not include a <default option>, then the null value.

   b) If CHARACTER_LENGTH(DO) > ML, then "TRUNCATED".

   c) Otherwise, DO.

8) The values of IS_NULLABLE have the following meanings:

YES	The column is possibly nullable.
NO	The column is known not nullable.

9) The values of IS_SELF_REFERENCING have the following meanings:

YES	The column is a self-referencing column.
NO	The column is not a self-referencing column.

10) The values of IS_IDENTITY have the following meanings:

YES	The column is an identity column.
NO	The column is not an identity column.

11) The values of IDENTITY_GENERATION have the following meanings:

ALWAYS	The column is an identity column whose values are always generated.
BY DEFAULT	The column is an identity column whose values are generated by default.
null	The column is not an identity column.

12) The value of IDENTITY_START is null if the column is not an identity column; otherwise, it is a character representation of the start value of the column being described.

13) The value of IDENTITY_INCREMENT is null if the column is not an identity column; otherwise, it is a character representation of the increment of the column being described.

14) The value of IDENTITY_MAXIMUM is null if the column is not an identity column; otherwise, it is a character representation of the maximum value of the column being described.

15) The value of IDENTITY_MINIMUM is null if the column is not an identity column; otherwise, it is a character representation of the minimum value of the column being described.

16) The value of IDENTITY_CYCLE is null if the column is not an identity column; otherwise, it is either YES or NO. The values of IDENTITY_CYCLE have the following meanings:

YES	The cycle option of the column is CYCLE.
NO	The cycle option of the column is NO CYCLE.
null	The column is not an identity column.

17) The values of IS_GENERATED have the following meanings:

NEVER	The column is not a generated column.
ALWAYS	The column is generated and stored.

18) The value of GENERATION_EXPRESSION is the text of the <generation expression> specified in the <column definition> when the column identified by COLUMN_NAME is defined.

19) The values of IS_UPDATABLE have the following meanings:

YES	The column is updatable.
NO	The column is not updatable.

20) The values of IS_SYSTEM_TIME_PERIOD_START have the following meanings:

YES	The column is a system-time-period start column.
NO	The column is not a system-time period start column.

21) The values of IS_SYSTEM_TIME_PERIOD_END have the following meanings:

YES	The column is a system-time period end column.
NO	The column is a system-time period end column.

22) The values of SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION have the following meanings:

ALWAYS	The column is a system-time period start column or a system-time period end column whose values are always generated.
null	The column is not a system-time period start column or a system-time period end column.

6.22   DATA_TYPE_DESCRIPTOR base table

This Subclause is modified by Subclause 25.2, "DATA_TYPE_DESCRIPTOR base table", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 22.1, "DATA_TYPE_DESCRIPTOR base table", in ISO/IEC 9075-14.

Function

The DATA_TYPE_DESCRIPTOR table has one row for each usage of a datatype as identified by ISO/IEC 9075. It effectively contains a representation of the data type descriptors.

Definition

```
CREATE TABLE DATA_TYPE_DESCRIPTOR (
    OBJECT_CATALOG                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                         INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_OBJECT_TYPE
        CHECK ( OBJECT_TYPE IN
              ( 'TABLE', 'DOMAIN', 'USER-DEFINED TYPE',
                'ROUTINE', 'SEQUENCE' ) ),
    DTD_IDENTIFIER                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DATA_TYPE                           INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT DATA_TYPE_DESCRIPTOR_OBJECTDATA_TYPE_NOT_NULL
        NOT NULL,
    CHARACTER_SET_CATALOG               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_SCHEMA                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_MAXIMUM_LENGTH            INFORMATION_SCHEMA.CARDINAL_NUMBER,
    CHARACTER_OCTET_LENGTH              INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COLLATION_CATALOG                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_SCHEMA                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_NAME                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    NUMERIC_PRECISION                   INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_PRECISION_RADIX             INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_SCALE                       INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DECLARED_DATA_TYPE                  INFORMATION_SCHEMA.CHARACTER_DATA,
    DECLARED_NUMERIC_PRECISION          INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DECLARED_NUMERIC_SCALE              INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DATETIME_PRECISION                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
    INTERVAL_TYPE                       INFORMATION_SCHEMA.CHARACTER_DATA,
    INTERVAL_PRECISION                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
    USER_DEFINED_TYPE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_CATALOG                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_SCHEMA                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_NAME                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MAXIMUM_CARDINALITY                 INFORMATION_SCHEMA.CARDINAL_NUMBER,
    TABLE_SEMANTICS                     INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT DATA_TYPE_DESCRIPTOR_TABLE_SEMANTICS_CHECK
        CHECK TABLE_SEMANTICS IN ( 'ROW', 'SET' ),
    IS_PRUNABLE                         INFORMATION_SCHEMA.YES_OR_NO,
```

```
HAS_PASS_THROUGH_COLUMNS              INFORMATION_SCHEMA.YES_OR_NO,

CONSTRAINT DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS
  CHECK ( ( DATA_TYPE IN
            ( 'CHARACTER', 'CHARACTER VARYING', 'CHARACTER LARGE OBJECT' )
          AND
            ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
              CHARACTER_SET_NAME ) IS NOT NULL
          AND
            ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
          AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
              DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
              DECLARED_NUMERIC_SCALE ) IS NULL
          AND
            DATETIME_PRECISION IS NULL
          AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME ) IS NULL
          AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
            MAXIMUM_CARDINALITY IS NULL
          AND
            ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
        OR
          ( DATA_TYPE IN
            ( 'BINARY', 'BINARY VARYING', 'BINARY LARGE OBJECT' )
          AND
            ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
          AND
            ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
              CHARACTER_SET_NAME, COLLATION_CATALOG,
              COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
          AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
              DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
              DECLARED_NUMERIC_SCALE ) IS NULL
          AND
            DATETIME_PRECISION IS NULL
          AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME ) IS NULL
          AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
            MAXIMUM_CARDINALITY IS NULL
          AND
            ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
        OR
          ( DATA_TYPE IN
            ( 'SMALLINT', 'INTEGER', 'BIGINT' )
          AND
```

```
          DECLARED_DATA_TYPE IN
            ( 'SMALLINT', 'INTEGER', 'BIGINT', 'NUMERIC', 'DECIMAL' )
        AND
          ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
            COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
        AND
          NUMERIC_PRECISION_RADIX IN ( 2, 10 )
        AND
          NUMERIC_PRECISION IS NOT NULL
        AND
          NUMERIC_SCALE = 0
        AND
          ( DECLARED_NUMERIC_SCALE IS NULL OR DECLARED_NUMERIC_SCALE = 0 )
        AND
          DATETIME_PRECISION IS NULL
        AND
          ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
        AND
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IS NULL
        AND
          ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
        AND
          MAXIMUM_CARDINALITY IS NULL
        AND
          ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
        ( DATA_TYPE IN
            ( 'NUMERIC', 'DECIMAL' )
        AND
          DECLARED_DATA_TYPE IN
            ( 'SMALLINT', 'INTEGER', 'BIGINT', 'NUMERIC', 'DECIMAL' )
        AND
          ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
            COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
        AND
          NUMERIC_PRECISION_RADIX = 10
        AND
          ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
        AND
          DATETIME_PRECISION IS NULL
        AND
          ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
        AND
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IS NULL
        AND
          ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
        AND
          MAXIMUM_CARDINALITY IS NULL
        AND
          ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
        ( DATA_TYPE IN
            ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
```

```
AND
  DECLARED_DATA_TYPE IN
    ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
AND
  ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
    CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
AND
  NUMERIC_PRECISION IS NOT NULL
AND
  NUMERIC_PRECISION_RADIX = 2
AND
  NUMERIC_SCALE IS NULL
AND
  DATETIME_PRECISION IS NULL
AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME ) IS NULL
AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
  MAXIMUM_CARDINALITY IS NULL
AND
  ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
OR
  ( DATA_TYPE = 'DECFLOAT'
AND
  DECLARED_DATA_TYPE = 'DECFLOAT'
AND
  ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
    CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
AND
  NUMERIC_PRECISION IS NOT NULL
AND
  NUMERIC_PRECISION_RADIX = 10
AND
  NUMERIC_SCALE IS NULL
AND
  DATETIME_PRECISION IS NULL
AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
    USER_DEFINED_TYPE_NAME ) IS NULL
AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
  MAXIMUM_CARDINALITY IS NULL
AND
  ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
OR
  ( DATA_TYPE IN
    ( 'DATE', 'TIME', 'TIMESTAMP',
      'TIME WITH TIME ZONE', 'TIMESTAMP WITH TIME ZONE' )
```

```
        AND
          ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
            COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
        AND
          ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
            DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE ) IS NULL
        AND
          DATETIME_PRECISION IS NOT NULL
        AND
          ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
        AND
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IS NULL
        AND
          ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
        AND
          MAXIMUM_CARDINALITY IS NULL
        AND
          ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
        ( DATA_TYPE = 'INTERVAL'
        AND
          ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
            COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
        AND
          ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
            DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
            DECLARED_NUMERIC_SCALE ) IS NULL
        AND
          DATETIME_PRECISION IS NOT NULL
        AND
          INTERVAL_TYPE IN
            ( 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND',
              'YEAR TO MONTH', 'DAY TO HOUR', 'DAY TO MINUTE',
              'DAY TO SECOND', 'HOUR TO MINUTE',
              'HOUR TO SECOND', 'MINUTE TO SECOND' )
        AND
          INTERVAL_PRECISION IS NOT NULL
        AND
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IS NULL
        AND
          ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
        AND
          MAXIMUM_CARDINALITY IS NULL
        AND
          ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
        ( DATA_TYPE = 'BOOLEAN'
        AND
          ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
            CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
            COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
        AND
```

```
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
        DECLARED_NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
      MAXIMUM_CARDINALITY IS NULL
    AND
      ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
  OR
    ( DATA_TYPE = 'USER-DEFINED'
    AND
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
        CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
        COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
        DECLARED_NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NOT NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
      MAXIMUM_CARDINALITY IS NULL
    AND
      ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
  OR
    ( DATA_TYPE = 'REF'
    AND
      ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
    AND
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, COLLATION_CATALOG,
        COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
        DECLARED_NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
```

```
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME ) IS NOT NULL
          AND
              MAXIMUM_CARDINALITY IS NULL
          AND
              ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
          ( DATA_TYPE = 'ARRAY'
          AND
              ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
                CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
                COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
          AND
              ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
                DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
                DECLARED_NUMERIC_SCALE ) IS NULL
          AND
              DATETIME_PRECISION IS NULL
          AND
              ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME ) IS NULL
          AND
              ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
              MAXIMUM_CARDINALITY IS NOT NULL
          AND
              ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
          ( DATA_TYPE = 'MULTISET'
          AND
              ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
                CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
                COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
          AND
              ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
                DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
                DECLARED_NUMERIC_SCALE ) IS NULL
          AND
              DATETIME_PRECISION IS NULL
          AND
              ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME ) IS NULL
          AND
              ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
              MAXIMUM_CARDINALITY IS NULL
          AND
              ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
      OR
          ( DATA_TYPE = 'ROW'
          AND
```

```
         ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
           CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
           CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
           COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
       AND
         ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
           DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
           DECLARED_NUMERIC_SCALE ) IS NULL
       AND
         DATETIME_PRECISION IS NULL
       AND
         ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
       AND
         ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
           USER_DEFINED_TYPE_NAME ) IS NULL
       AND
         ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
       AND
         MAXIMUM_CARDINALITY IS NULL
       AND
         ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
     OR
      ( DATA_TYPE = 'TABLE'
       AND
         ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
           CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
           CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
           COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
       AND
         ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
           DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
           DECLARED_NUMERIC_SCALE ) IS NULL
       AND
         DATETIME_PRECISION IS NULL
       AND
         ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
       AND
         ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
           USER_DEFINED_TYPE_NAME ) IS NULL
       AND
         ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
       AND
         MAXIMUM_CARDINALITY IS NULL )
     OR
      ( DATA_TYPE = 'DESCRIPTOR'
       AND
         ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
           CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
           CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
           COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
       AND
         ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
           DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
           DECLARED_NUMERIC_SCALE ) IS NULL
       AND
         DATETIME_PRECISION IS NULL
       AND
```

```
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
          AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME ) IS NULL
          AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
            MAXIMUM_CARDINALITY IS NULL
          AND
            ( TABLE_SEMANTICS, IS_PRUNABLE, HAS_PASS_THROUGH_COLUMNS ) IS NULL )
        OR
          ( DATA_TYPE NOT IN
            ( 'CHARACTER', 'CHARACTER VARYING',
              'CHARACTER LARGE OBJECT', 'BINARY',
              'BINARY VARYING', 'BINARY LARGE OBJECT',
              'NUMERIC', 'DECIMAL', 'SMALLINT', 'INTEGER', 'BIGINT',
              'FLOAT', 'REAL', 'DOUBLE PRECISION', 'DECFLOAT',
              'DATE', 'TIME', 'TIMESTAMP',
              'INTERVAL', 'BOOLEAN', 'USER-DEFINED',
              'REF', 'ROW', 'ARRAY', 'MULTISET',
              'TABLE', 'DESCRIPTOR' ) ) ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_UDT
    CHECK ( USER_DEFINED_TYPE_CATALOG <>
            ANY ( SELECT CATALOG_NAME
                  FROM SCHEMATA )
        OR
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ) IN
          ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                   USER_DEFINED_TYPE_NAME
            FROM USER_DEFINED_TYPES ) ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_PRIMARY_KEY
    PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, DTD_IDENTIFIER ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_COLLATION_CHARACTER_SET_APPLICABILITY

    CHECK ( CHARACTER_SET_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
          OR
            COLLATION_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
          OR
            ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
                    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IN
            ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
                    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
              FROM COLLATION_CHARACTER_SET_APPLICABILITY ) ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA )
      REFERENCES SCHEMATA
```

)

Description

1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, and OBJECT_NAME are the fully qualified name of the object (table, domain, SQL-invoked routine, user-defined type, or sequence generator) whose descriptor includes the data type descriptor, and OBJECT_TYPE is 'TABLE', 'DOMAIN', 'ROUTINE', 'USER-DEFINED TYPE', or 'SEQUENCE', as the case may be.

2) The value of DTD_IDENTIFIER is the implementation-dependent value that uniquely identifies the data type descriptor among all data type descriptors of the schema object identified by OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and OBJECT_TYPE.

3) Let *DTN* be the data type name specified in the declaration of the data type being described. The value of DATA_TYPE is

   Case:

   a) If the data type being described is an exact numeric type or an approximate numeric type, then the name of the normal form of *DTN*.

   b) Otherwise, *DTN*.

   NOTE 10 — The normal form of a an exact numeric data type or an approximate numeric data type is defined in Subclause 6.1, "<data type>" of Part 2 of this International Standard.

4) The value of DECLARED_DATA_TYPE is

   Case:

   a) If the SQL-Implementation knows the data type name specified in the declaration of the data type being described, then *DTN*.

   b) Otherwise, the null value.

   NOTE 11 — The declared data type of a numeric item may be unknown if, for example, the SQL-Implementation has been upgraded to support Feature T322 but manages schema items declared before the upgrade was installed.

5) Case:

   a) If the data type being described is a character string type, then the values of CHARACTER_MAXIMUM_LENGTH and CHARACTER_OCTET_LENGTH are, respectively, the length or maximum length in characters and the length or maximum length in octets of the data type being described.

   b) If the data type being described is a binary string type, then the values of CHARACTER_MAXIMUM_LENGTH and CHARACTER_OCTET_LENGTH are the length or maximum length in octets of the data type being described.

   c) If the data type being described is a reference type, then the values of CHARACTER_MAXIMUM_LENGTH and CHARACTER_OCTET_LENGTH are the length in octets of the data type being described.

   d) Otherwise, the values of CHARACTER_MAXIMUM_LENGTH and CHARACTER_OCTET_LENGTH are the null value.

6) Case:

    a)   If the data type being described is a character string type, then the values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA and COLLATION_NAME are, respectively, the qualified name of the character set and applicable collation, if any.

    b)   Otherwise, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA and COLLATION_NAME are the null value.

7) For a numeric type, the values of NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, and NUMERIC_SCALE are, respectively, the implemented precision, the radix of the precision and the scale of the data type being described.

8) For a numeric type, the values of DECLARED_NUMERIC_PRECISION, and DECLARED_NUMERIC_SCALE are, respectively,

Case:

    a)   If they are known to the SQL-Implementation, the declared precision and declared scale of the data type being described.

    b)   Otherwise, the null value.

        NOTE 12 — The declared precision or scale of a numeric item may be unknown if, for example, the SQL-Implementation has been upgraded to support Feature T322 but manages schema items declared before the upgrade was installed, or if it was not specified in the declaration of the item.

9) For a datetime or interval type, the value of DATETIME_PRECISION is the fractional seconds precision of the data type being described.

10) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA and USER_DEFINED_TYPE_NAME are, the fully qualified name of the user-defined type or the referenced structured type if it is a reference type, if specified, of the data type being described.

11) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_TYPE are the value for <interval qualifier> (as specified in Table 30, "Codes used for <interval qualifier>s in Dynamic SQL", in [ISO9075-2]) for the data type being described; otherwise, INTERVAL_TYPE is the null value.

12) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_PRECISION are the interval leading field precision of the data type being described; otherwise, INTERVAL_PRECISION is the null value.

13) Case:

    a)   If DATA_TYPE is 'USER-DEFINED', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the user-defined type being described.

    b)   If the DATA_TYPE is 'REF', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the referenced structured type of the reference type being described.

    c)   Otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

14) If DATA_TYPE is 'REF', then the values of SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are the fully qualified name of the referenceable table, if any; otherwise, the values of SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are the null value.

15) If DATA_TYPE is the name of some character string type and OBJECT_SCHEMA is 'INFORMA-TION_SCHEMA', then the values for CHARACTER_MAXIMUM_LENGTH, CHARAC-TER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARAC-TER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are implementation-defined.

16) If DATA_TYPE is 'ARRAY', then the value of MAXIMUM_CARDINALITY is the maximum cardinality of the array type being described. Otherwise, the value of MAXIMUM_CARDINALITY is the null value.

17) If DATA_TYPE is 'ROW' then the data type being described is a row type.

18) Case:

   a)   If the data type being described is a generic table parameter type of a polymorphic table function, then

   i)   DATA_TYPE is 'TABLE'.

   ii)   If the generic table parameter has row semantics, then TABLE_SEMANTICS is 'ROW'; other-wise, it is 'SET'.

   iii)   If the generic table paramater has set semantics and PRUNE WHEN EMPTY is specified, then IS_PRUNABLE is 'YES'; otherwise, it is 'NO'.

   iv)   If the generic table parameter has pass-through columns, then HAS_PASS_THROUGH_COLUMNS is 'YES'; otherwise, it is 'NO'.

   b)   If the data type being described is the return type of a polymorphic table function, then DATA_TYPE is 'TABLE', and TABLE_SEMANTICS, IS_PRUNABLE and HAS_PASS_THROUGH_COLUMNS are the null value.

   c)   Otherwise, TABLE_SEMANTICS, IS_PRUNABLE, and HAS_PASS_THROUGH_COLUMNS are the null value.

19) If the data type being described is the parameter type of a descriptor parameter of a polymorphic table function, then DATA_TYPE is 'DESCRIPTOR'; otherwise, it is not 'DESCRIPTOR'.

6.23 DIRECT_SUPERTABLES base table

Function

The DIRECT_SUPERTABLES base table contains one row for each direct subtable-supertable relationship.

Definition

```
CREATE TABLE DIRECT_SUPERTABLES (
      TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SUPERTABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

      CONSTRAINT DIRECT_SUPERTABLES_PRIMARY_KEY
        PRIMARY KEY (TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, SUPERTABLE_NAME ),

      CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_TABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
          REFERENCES TABLES,

      CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_SUPERTABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, SUPERTABLE_NAME )
          REFERENCES TABLES,

      CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_SAME_TABLES
        CHECK ( TABLE_NAME <> SUPERTABLE_NAME ),

      CONSTRAINT DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA,
                  SUPERTABLE_NAME, TABLE_NAME ) NOT IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA,
                         TABLE_NAME, SUPERTABLE_NAME
                  FROM DIRECT_SUPERTABLES ) ),

      CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_ALSO_INDIRECT
        CHECK (
          NOT EXISTS (
            WITH RECURSIVE SUPER
                ( TYPE, SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME,
                  SUPERTABLE_NAME )
                AS
                ( SELECT 0, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                           SUPERTABLE_NAME
                  FROM DIRECT_SUPERTABLES
                UNION
                  SELECT 1, S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME,
                           D.SUPERTABLE_NAME
                  FROM SUPER AS S
                JOIN
                      DIRECT_SUPERTABLES AS D
                  ON ( S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME )
                   = ( D.TABLE_CATALOG, D.TABLE_SCHEMA, D.TABLE_NAME ) )
```

```
             SELECT SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME, SUPERTABLE_NAME
             FROM SUPER
             WHERE TYPE = 1
           INTERSECT
             SELECT *
             FROM DIRECT_SUPERTABLES ) )
     )
```

Description

1) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the fully qualified name of the subtable.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, and SUPERTABLE_NAME are the fully qualified name of the direct supertable.

6.24 DIRECT_SUPERTYPES base table

Function

The DIRECT_SUPERTYPES base table contains one row for each direct subtype-supertype relationship.

Definition

```
CREATE TABLE DIRECT_SUPERTYPES (
     USER_DEFINED_TYPE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
     USER_DEFINED_TYPE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
     USER_DEFINED_TYPE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SUPERTYPE_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SUPERTYPE_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SUPERTYPE_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,

     CONSTRAINT DIRECT_SUPERTYPES_PRIMARY_KEY
       PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                     USER_DEFINED_TYPE_NAME,
                     SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ),

     CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_1
       FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                     USER_DEFINED_TYPE_NAME )
         REFERENCES USER_DEFINED_TYPES,

     CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_2
       FOREIGN KEY ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME )
         REFERENCES USER_DEFINED_TYPES,

     CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_SAME_TYPES
       CHECK ( ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                 USER_DEFINED_TYPE_NAME ) <>
               ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) ),

     CONSTRAINT DIRECT_SUPERTYPES_CHECK_NO_REFLEXIVITY
       CHECK ( ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME,
                 USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                 USER_DEFINED_TYPE_NAME ) NOT IN
               ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                        USER_DEFINED_TYPE_NAME,
                        SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA,
                        SUPERTYPE_NAME
                 FROM DIRECT_SUPERTYPES ) ),

     CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_ALSO_INDIRECT
       CHECK (
         NOT EXISTS (
             WITH RECURSIVE SUPER
                 ( TYPE, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                   USER_DEFINED_TYPE_NAME,
                   SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) AS
                 ( SELECT 0, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
```

```
                  USER_DEFINED_TYPE_NAME,
                  SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
           FROM DIRECT_SUPERTYPES
        UNION
          SELECT 1, S.USER_DEFINED_TYPE_CATALOG, S.USER_DEFINED_TYPE_SCHEMA,
                  S.USER_DEFINED_TYPE_NAME,
                  D.SUPERTYPE_CATALOG, D.SUPERTYPE_SCHEMA, D.SUPERTYPE_NAME
          FROM SUPER AS S
            JOIN
              DIRECT_SUPERTYPES AS D
             ON ( D.USER_DEFINED_TYPE_CATALOG, D.USER_DEFINED_TYPE_SCHEMA,
                  D.USER_DEFINED_TYPE_NAME ) =
                ( S.SUPERTYPE_CATALOG, S.SUPERTYPE_SCHEMA,
                  S.SUPERTYPE_NAME ) )
      SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
             USER_DEFINED_TYPE_NAME,
             SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
      FROM SUPER
      WHERE TYPE = 1
    INTERSECT
      SELECT *
      FROM DIRECT_SUPERTYPES ) )
)
```

Description

1) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the fully qualified name of the user-defined type that is a direct subtype.

2) The values of SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, and SUPERTYPE_NAME are the fully qualified name of the user-defined type that is the direct supertype.

6.25 DOMAIN_CONSTRAINTS base table

Function

The DOMAIN_CONSTRAINTS table has one row for each domain constraint associated with a domain. It effectively contains a representation of the domain constraint descriptors.

Definition

```
CREATE TABLE DOMAIN_CONSTRAINTS (
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DOMAIN_CATALOG_NOT_NULL
        NOT NULL,
    DOMAIN_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DOMAIN_SCHEMA_NOT_NULL
        NOT NULL,
    DOMAIN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DOMAIN_NAME_NOT_NULL
        NOT NULL,
    IS_DEFERRABLE           INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT IS_DEFERRABLE_NOT_NULL
        NOT NULL,
    INITIALLY_DEFERRED      INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT INITIALLY_DEFERRED_NOT_NULL
        NOT NULL,

    CONSTRAINT DOMAIN_CONSTRAINTS_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_CHECK_CONSTRAINTS
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
        REFERENCES CHECK_CONSTRAINTS,

    CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_DOMAINS
      FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
        REFERENCES DOMAINS,

    CONSTRAINT DOMAIN_CONSTRAINTS_CHECK_DEFERRABLE
      CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN
              ( VALUES ( 'NO',  'NO' ),
                       ( 'YES', 'NO' ),
                       ( 'YES', 'YES' ) ) ),

    CONSTRAINT DOMAIN_CONSTRAINTS_CHECK_SCHEMA_IDENTITY
      CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
            = ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) )
```

)

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the fully qualified name of the domain constraint.

2) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA and DOMAIN_NAME are the fully qualified name of the domain in which the domain constraint is defined.

3) The values of IS_DEFERRABLE have the following meanings:

YES	The domain constraint is deferrable.
NO	The domain constraint is not deferrable.

4) The values of INITIALLY_DEFERRED have the following meanings:

YES	The domain constraint is initially deferred.
NO	The domain constraint is initially immediate.

6.26 DOMAINS base table

Function

The DOMAINS table has one row for each domain. It effectively contains a representation of the domain descriptors.

Definition

```
CREATE TABLE DOMAINS (
    DOMAIN_CATALOG                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER                  INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DTD_IDENTIFIER_NOT_NULL
          NOT NULL,
    DOMAIN_DEFAULT                  INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT DOMAINS_PRIMARY_KEY
      PRIMARY KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ),

    CONSTRAINT DOMAINS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) REFERENCES SCHEMATA,

    CONSTRAINT DOMAIN_CHECK_DATA_TYPE
      CHECK ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
                'DOMAIN', DTD_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                       OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) )
  )
```

Description

1) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are the fully qualified name of the domain.

2) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, and DTD_IDENTIFIER are the values of DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the domain.

3) The value of DOMAIN_DEFAULT is null if the domain being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of DOMAIN_DEFAULT is "TRUNCATED". Otherwise, the value of DOMAIN_DEFAULT is a character representation of the default value for the domain that obeys the rules specified for <default option> in Subclause 11.5, "<default clause>".

> NOTE 13 — "TRUNCATED" is different from other values like CURRENT_USER or CURRENT_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.

6.27 ELEMENT_TYPES base table

Function

The ELEMENT_TYPES table has one row for each collection type. It effectively contains a representation of the element descriptor of the collection type.

Definition

```
CREATE TABLE ELEMENT_TYPES (
    OBJECT_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLECTION_TYPE_IDENTIFIER    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ELEMENT_TYPES_PRIMARY_KEY
      PRIMARY KEY (OBJECT_CATALOG, OBJECT_SCHEMA,
                   OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ),

    CONSTRAINT ELEMENT_TYPES_CHECK_COLLECTION_TYPE
      CHECK (
        ( OBJECT_CATALOG, OBJECT_SCHEMA,
          OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ) IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                   OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
            FROM DATA_TYPE_DESCRIPTOR
            WHERE DATA_TYPE IN ( 'ARRAY', 'MULTISET' ) ) ),

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
    )
```

Description

1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and COLLECTION_TYPE_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the collection type whose element type is being described.

2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,

OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the element type of the collection type.

3) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the element type.

NOTE 14 — The root data type indicates the row of DATA_TYPE_DESCRIPTOR that directly belongs to some column, attribute, field, *etc.*. The root data type is used to identify the access rights a user has on a specific row of the base table ELEMENT_TYPES.

6.28 FIELDS base table

Function

The FIELDS table has one row for each field of each row type. It effectively contains a representation of the field descriptors.

Definition

```
CREATE TABLE FIELDS (
    OBJECT_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROW_IDENTIFIER              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FIELD_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION            INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT FIELDS_ORDINAL_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT FIELDS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT FIELDS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM FIELDS
                          GROUP BY OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                                   OBJECT_TYPE, ROW_IDENTIFIER ) ),
    DTD_IDENTIFIER              INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT FIELDS_PRIMARY_KEY
      PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME ),

    CONSTRAINT FIELDS_UNIQUE
      UNIQUE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
               OBJECT_TYPE, ROW_IDENTIFIER, ORDINAL_POSITION ),

    CONSTRAINT FIELDS_CHECK_ROW_TYPE
      CHECK (
        ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, ROW_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                 OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR
          WHERE DATA_TYPE = 'ROW') ),

    CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
```

```
                    OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
   REFERENCES DATA_TYPE_DESCRIPTOR
 )
```

Description

1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROW_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the row type containing the field being described.

2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the field type.

   NOTE 15 — The root data type indicates the row of DATA_TYPE_DESCRIPTOR that directly belongs to some column, attribute, field, *etc.*. The root data type is used to identify the access rights a user has on a specific row of the base table FIELDS.

3) The value of FIELD_NAME is the name of the field being described.

4) The value of ORDINAL_POSITION is the ordinal position of the field in the row type.

5) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the field being described.

6.29   KEY_COLUMN_USAGE base table

Function

The KEY_COLUMN_USAGE table has one or more rows for each row in the TABLE_CONSTRAINTS table that has a CONSTRAINT_TYPE of "UNIQUE", "PRIMARY KEY", or "FOREIGN KEY". The rows list the columns that constitute each unique constraint, and the referencing columns in each foreign key constraint.

Definition

```
CREATE TABLE KEY_COLUMN_USAGE (
    CONSTRAINT_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_COLUMN_TABLE_CATALOG_NOT_NULL
        NOT NULL,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_COLUMN_TABLE_SCHEMA_NOT_NULL
        NOT NULL,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_COLUMN_TABLE_NAME_NOT_NULL
        NOT NULL,
    COLUMN_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION       INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT KEY_COLUMN_ORDINAL_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                        FROM KEY_COLUMN_USAGE
                        GROUP BY CONSTRAINT_CATALOG,
                                 CONSTRAINT_SCHEMA,
                                 CONSTRAINT_NAME ) ),
    POSITION_IN_UNIQUE_CONSTRAINT INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_GREATER_THAN_ZERO_CHECK
        CHECK (POSITION_IN_UNIQUE_CONSTRAINT > 0),

    CONSTRAINT KEY_COLUMN_USAGE_UNIQUE_POSITION_IN_UNIQUE_CONSTRAINT
      UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
               CONSTRAINT_NAME, POSITION_IN_UNIQUE_CONSTRAINT ),

    CONSTRAINT KEY_COLUMN_USAGE_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    COLUMN_NAME ),

    CONSTRAINT KEY_COLUMN_USAGE_UNIQUE
      UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
               CONSTRAINT_NAME, ORDINAL_POSITION ),

    CONSTRAINT KEY_COLUMN_USAGE_FOREIGN_KEY_COLUMNS
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
```

```
    REFERENCES COLUMNS,

CONSTRAINT KEY_COLUMN_CONSTRAINT_TYPE_CHECK
CHECK (
    ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
    ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
      FROM TABLE_CONSTRAINTS
      WHERE CONSTRAINT_TYPE IN
            ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) ),

CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_CHECK
  CHECK ( ( POSITION_IN_UNIQUE_CONSTRAINT IS NULL
          AND
            NOT ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
                 ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                   FROM REFERENTIAL_CONSTRAINTS ) )
        OR
          ( POSITION_IN_UNIQUE_CONSTRAINT IS NOT NULL
          AND
            ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
              ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                FROM REFERENTIAL_CONSTRAINTS )
        AND
            NOT EXISTS ( SELECT 1
                         FROM KEY_COLUMN_USAGE
                         GROUP BY CONSTRAINT_CATALOG,
                                  CONSTRAINT_SCHEMA,
                                  CONSTRAINT_NAME
                         HAVING COUNT ( POSITION_IN_UNIQUE_CONSTRAINT )
                                <> MAX ( POSITION_IN_UNIQUE_CONSTRAINT ) ) ) )
)
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier of the table name, and the column name of the column that participates in the unique, primary key, or foreign key constraint being described.

3) The value of ORDINAL_POSITION is the ordinal position of the specific column in the constraint being described. If the constraint described is a key of cardinality 1 (one), then the value of ORDINAL_POSITION is always 1 (one).

4) Case:

    a) If the constraint being described is a foreign key constraint, then the value of POSI-TION_IN_UNIQUE_CONSTRAINT is the ordinal position of the referenced column corresponding to the referencing column being described, in the corresponding unique key constraint.

    b) Otherwise, the value of POSITION_IN_UNIQUE_CONSTRAINT is the null value.

6.30 KEY_PERIOD_USAGE base table

Function

The KEY_PERIOD_USAGE table has one or more rows for each row in the TABLE_CONSTRAINTS table that has a CONSTRAINT_TYPE of "UNIQUE", "PRIMARY KEY", or "FOREIGN KEY". The rows list the period that is referenced in <without overlap specification> in each unique constraint, and the <referencing period specification> in each foreign key constraint.

Definition

```
CREATE TABLE KEY_PERIOD_USAGE (
    CONSTRAINT_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_PERIOD_TABLE_CATALOG_NOT_NULL
        NOT NULL,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_PERIOD_TABLE_SCHEMA_NOT_NULL
        NOT NULL,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT KEY_PERIOD_TABLE_NAME_NOT_NULL
        NOT NULL,
    PERIOD_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT KEY_PERIOD_USAGE_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    PERIOD_NAME ),

    CONSTRAINT KEY_PERIOD_USAGE_FOREIGN_KEY_CHECK_TABLE_USAGE
      FOREIGN KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES CHECK_TABLE_USAGE,

    CONSTRAINT KEY_PERIOD_USAGE_CHECK_REFERENCES_PERIODS
      CHECK ( TABLE_CATALOG NOT IN
                  ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
              OR
                ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME) IN
                  ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME
                    FROM PERIODS ) ),

    CONSTRAINT KEY_PERIOD_CONSTRAINT_TYPE_CHECK
    CHECK (
      ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
      ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM TABLE_CONSTRAINTS
        WHERE CONSTRAINT_TYPE IN
            ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) )
```

)

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and PERIOD_NAME are the catalog name, unqualified schema name, qualified identifier of the table name, and the period name of the period that participates in the unique, primary key, or foreign key constraint being described.

6.31   METHOD_SPECIFICATION_PARAMETERS base table

Function

The METHOD_SPECIFICATION_PARAMETERS base table has one row for each SQL parameter of each method specification described in the METHOD_SPECIFICATIONS base table.

Definition

```
CREATE TABLE METHOD_SPECIFICATION_PARAMETERS (
    SPECIFIC_CATALOG                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION                        INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT METHOD_SPECIFICATION_PARAMETER_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM METHOD_SPECIFICATION_PARAMETERS
                          GROUP BY SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) ),
    DTD_IDENTIFIER                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_MODE                          INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT METHOD_SPECIFICATION_PARAMETER_MODE_CHECK
        CHECK ( PARAMETER_MODE IN
                ( 'IN' ) ),
    IS_RESULT                               INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_IS_RESULT_NOT_NULL
        NOT NULL,
    AS_LOCATOR                              INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_AS_LOCATOR_NOT_NULL
        NOT NULL,
    PARAMETER_NAME                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_CATALOG               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_SCHEMA                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                ORDINAL_POSITION ),

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES METHOD_SPECIFICATIONS,

    CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_CHECK_DATA_TYPE
      CHECK (
        ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,SPECIFIC_NAME,
          'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                 OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) )
```

)

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method whose parameters are being described.

2) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the fully qualified name of the user-defined type with which the SQL-invoked method is associated.

3) The value of ORDINAL_POSITION is the ordinal position of the SQL parameter in the SQL-invoked method.

4) The values of PARAMETER_MODE have the following meanings:

IN	The SQL parameter being described is an input parameter.

5) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the parameter being described.

6) The values of IS_RESULT have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT parameter of a type-preserving function.

7) The values of AS_LOCATOR have the following meanings:

YES	The parameter is passed AS LOCATOR.
NO	The parameter is not passed AS LOCATOR.

8) Case:

   a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of PARAMETER_NAME is that <SQL parameter name>.

   b) Otherwise, the value of PARAMETER_NAME is the null value.

9) FROM_SQL_SPECIFIC_CATALOG, FROM_SQL_SPECIFIC_SCHEMA, and FROM_SQL_SPE-CIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the parameter being described.

6.32 METHOD_SPECIFICATIONS base table

This Subclause is modified by Subclause 15.3, "METHOD_SPECIFICATIONS base table", in ISO/IEC 9075-13.

Function

The METHOD_SPECIFICATIONS base table has one row for each method specification.

Definition

```
CREATE TABLE METHOD_SPECIFICATIONS (
    SPECIFIC_CATALOG                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    METHOD_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_STATIC                           INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATIONS_IS_STATIC_NOT_NULL
        NOT NULL,
    IS_OVERRIDING                       INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATIONS_IS_OVERRIDING_NOT_NULL
        NOT NULL,
    IS_CONSTRUCTOR                      INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_NOT_NULL
        NOT NULL,
    METHOD_LANGUAGE                     INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT METHOD_SPECIFICATIONS_LANGUAGE_CHECK
        CHECK ( METHOD_LANGUAGE IN
              ( 'SQL', 'ADA', 'C',
                'COBOL', 'FORTRAN', 'MUMPS', 'PASCAL','PLI' ) ),
    PARAMETER_STYLE                     INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT METHOD_SPECIFICATIONS_PARAMETER_STYLE_CHECK
        CHECK ( PARAMETER_STYLE IN
              ( 'SQL', 'GENERAL' ) ),
    DTD_IDENTIFIER                      INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_DETERMINISTIC                    INFORMATION_SCHEMA.YES_OR_NO,
    SQL_DATA_ACCESS                     INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_NOT_NULL
        NOT NULL
      CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_CHECK
        CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
    IS_NULL_CALL                        INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATIONS_IS_NULL_CALL_NOT_NULL
        NOT NULL,
    TO_SQL_SPECIFIC_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    AS_LOCATOR                          INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT METHOD_SPECIFICATIONS_AS_LOCATOR_NOT_NULL
        NOT NULL,
```

```
CREATED                             INFORMATION_SCHEMA.TIME_STAMP,
RESULT_CAST_FROM_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR              INFORMATION_SCHEMA.YES_OR_NO,

CONSTRAINT METHOD_SPECIFICATIONS_PRIMARY_KEY
  PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
    REFERENCES SCHEMATA,

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_USER_DEFINED_TYPES
  FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME )
    REFERENCES USER_DEFINED_TYPES MATCH FULL,

CONSTRAINT METHOD_SPECIFICATIONS_CHECK_DATA_TYPE
  CHECK (
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                 OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) ),

CONSTRAINT METHOD_SPECIFICATIONS_COMBINATIONS
  CHECK (
    ( ( METHOD_LANGUAGE = 'SQL'
      AND
        IS_DETERMINISTIC IS NULL )
    OR
      ( METHOD_LANGUAGE <> 'SQL'
      AND
        IS_DETERMINISTIC IS NOT NULL ) ) ),

CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_COMBINATION_CHECK
  CHECK ( IS_CONSTRUCTOR = 'NO' OR IS_OVERRIDING = 'NO' ),

CONSTRAINT METHOD_SPECIFICATIONS_SAME_SCHEMA
  CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
          ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),

CONSTRAINT METHOD_SPECIFICATIONS_CHECK_RESULT_CAST
  CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NULL )
      OR
          ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NOT NULL
        AND
          ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                     OBJECT_TYPE, RESULT_CAST_FROM_DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR )
          )
        )
```

)

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method being described.

2) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of the user-defined type with which the SQL-invoked method is associated.

3) The values of METHOD_NAME is the identifier of the method name of the SQL-invoked method being described.

4) The values of IS_STATIC have the following meanings:

YES	The SQL-invoked routine is a static method.
NO	The SQL-invoked routine is not a static method.

5) The values of IS_OVERRIDING have the following meanings:

YES	The SQL-invoked method is an overriding method.
NO	The SQL-invoked method is an original method.

6) The values of IS_CONSTRUCTOR have the following meanings:

YES	The SQL-invoked method is an SQL-invoked constructor method.
NO	The SQL-invoked method is not an SQL-invoked constructor method.

7) The values of USER_DEFINED_TYPE_CATALOG>, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the result type of the method.

8) The values of IS_NULL_CALL have the following meanings:

YES	The SQL-invoked routine is a null-call function.
NO	The SQL-invoked routine is not a null-call function.

9) The value of METHOD_LANGUAGE is the explicit or implicit <language name> contained in the method specification being described.

10) Case:

    a) If the method being defined specifies LANGUAGE SQL, then the values of IS_DETERMINISTIC, PARAMETER_STYLE, TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME are the null value.

    b) Otherwise:

        i) The values of IS_DETERMINISTIC have the following meanings:

YES	The method is deterministic.
NO	The method is possibly not deterministic.

        ii) The values of PARAMETER_STYLE have the following meanings:

SQL	The method specification specified PARAMETER STYLE SQL.
GENERAL	The method specification specified PARAMETER STYLE GENERAL.

        iii) TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPE-CIFIC_NAME are catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked method being described.

11) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

12) The values of AS_LOCATOR have the following meanings:

YES	The return value is passed AS LOCATOR.
NO	The return value is not passed AS LOCATOR.

13) The value of CREATED is

    Case:

a) If Feature T011, "Timestamp in Information Schema", is supported and the SQL-implementation knows the value of CURRENT_TIMESTAMP at the time when the SQL-invoked method specification being described was created, then that value.

b) Otherwise, the NULL value.

14) Case:

a) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the SQL-invoked method specifies a <result cast>, then the values of RESULT_CAST_FROM_DTD_IDENTIFIER and RESULT_CAST_AS_LOCATOR are the null value.

b) Otherwise, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and RESULT_CAST_FROM_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the <data type> specified in the <result cast> of the SQL-invoked method being described.

Case:

i) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of RESULT_CAST_AS_LOCATOR is 'NO'.

ii) Otherwise, the value of RESULT_CAST_AS_LOCATOR is 'YES'.

6.33   PARAMETERS base table

This Subclause is modified by Subclause 22.2, "PARAMETERS base table", in ISO/IEC 9075-14.

Function

The PARAMETERS table has one row for each SQL parameter of each SQL-invoked routine described in the ROUTINES base table.

Definition

```
CREATE TABLE PARAMETERS (
    SPECIFIC_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION           INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT PARAMETERS_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM PARAMETERS
                          GROUP BY SPECIFIC_CATALOG,
                                   SPECIFIC_SCHEMA,
                                   SPECIFIC_NAME ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_MODE             INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT PARAMETER_MODE_NOT_NULL
        NOT NULL
      CONSTRAINT PARAMETER_MODE_CHECK
        CHECK (
          PARAMETER_MODE IN
          ( 'IN', 'OUT', 'INOUT' ) ),
    IS_RESULT                  INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT PARAMETERS_IS_RESULT_NOT_NULL
        NOT NULL,
    AS_LOCATOR                 INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT PARAMETERS_AS_LOCATOR_NOT_NULL
        NOT NULL,
    PARAMETER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_DEFAULT          INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT PARAMETERS_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
                    SPECIFIC_NAME, ORDINAL_POSITION ),
    CONSTRAINT PARAMETERS_UNIQUE_CHECK
```

```
    UNIQUE (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, PARAMETER_NAME),
CONSTRAINT PARAMETERS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
    REFERENCES SCHEMATA,

CONSTRAINT PARAMETERS_CHECK_DATA_TYPE
  CHECK (
      ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
        SPECIFIC_NAME, 'ROUTINE', DTD_IDENTIFIER ) IN
      ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
        FROM DATA_TYPE_DESCRIPTOR ) )
)
```

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine that contains the SQL parameter being described.

2) The value of ORDINAL_POSITION is the ordinal position of the SQL parameter in the SQL-invoked routine.

3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, and DTD_IDEN-TIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the parameter.

4) The values of PARAMETER_MODE have the following meanings:

IN	The SQL parameter being described is an input parameter.
OUT	The SQL parameter being described is an output parameter.
INOUT	The SQL parameter being described is an input parameter and an output parameter.

5) The values of IS_RESULT have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT paramaeter of a type-preserving function.

6) The values of AS_LOCATOR have the following meanings:

YES	The parameter is passed AS LOCATOR.
NO	The parameter is not passed AS LOCATOR.

7) Case:

    a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of PARAMETER_NAME is that <SQL parameter name>.

    b) Otherwise, the value of PARAMETER_NAME is the null value.

8) Case:

    a) If the parameter that is being described is an input parameter that has a from-sql routine, then FROM_SQL_SPECIFIC_CATALOG, FROM_SQL_SPECIFIC_SCHEMA, and FROM_SQL_SPE-CIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the input parameter being described.

    b) Otherwise, FROM_SQL_SPECIFIC_CATALOG, FROM_SQL_SPECIFIC_SCHEMA, and FROM_SQL_SPECIFIC_NAME are the null value.

9) Case:

    a) If the parameter that is being described is an output parameter that has a to-sql routine, then TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the output parameter being described.

    b) Otherwise, TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME are the null value.

10) Case:

    a) If <parameter default> was specified when the SQL-invoked routine was created, then the value of PARAMETER_DEFAULT is that <parameter default>.

    b) Otherwise, the value of PARAMETER_DEFAULT is the null value.

6.34   PERIODS base table

Function

The PERIODS base table has one row for each period defined for a table. It effectively contains a representation of the period descriptors.

Definition

```
CREATE TABLE PERIODS (
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PERIOD_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    START_COLUMN_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT START_COLUMN_NAME_NOT_NULL
                NOT NULL,
    END_COLUMN_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT END_COLUMN_NAME_NOT_NULL
                NOT NULL,

    CONSTRAINT PERIODS_PRIMARY_KEY
      PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

    CONSTRAINT PERIODS_FOREIGN_KEY_TABLES
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TABLES,

    CONSTRAINT PERIODS_FOREIGN_KEY_COLUMNS_1
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, START_COLUMN_NAME )
        REFERENCES COLUMNS,

    CONSTRAINT PERIODS_FOREIGN_KEY_COLUMNS_2
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, END_COLUMN_NAME )
        REFERENCES COLUMNS
)
```

Description

1)   The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the period being described.

2)   The value of PERIOD_NAME is the name of the period being described.

3)   The value of START_COLUMN_NAME is the name of the column in the table containing the period being described that acts as the start column of the period.

4)   The value of END_COLUMN_NAME is the name of the column in the table containing the period being described that acts as the end column of the period.

6.35 PRIVATE_PARAMETERS base table

Function

The PRIVATE_PARAMETERS table has one row for each private parameter of each polymorphic table function described in the ROUTINES base table.

Definition

```
CREATE TABLE PRIVATE_PARAMETERS (
    SPECIFIC_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION           INFORMATION_SCHEMA.CARDINAL_NUMBER
      CONSTRAINT PRIVATE_PARAMETERS_POSITION_NOT_NULL
        NOT NULL
      CONSTRAINT PRIVATE_PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
      CONSTRAINT PRIVATE_PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM PRIVATE_PARAMETERS
                          GROUP BY SPECIFIC_CATALOG,
                                   SPECIFIC_SCHEMA,
                                   SPECIFIC_NAME ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_DEFAULT          INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT PRIVATE_PARAMETERS_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
                    SPECIFIC_NAME, ORDINAL_POSITION ),
    CONSTRAINT PRIVATE_PARAMETERS_UNIQUE_CHECK
      UNIQUE (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, PARAMETER_NAME),
    CONSTRAINT PRIVATE_PARAMETERS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT PRIVATE_PARAMETERS_CHECK_DATA_TYPE
      CHECK (
        ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
          SPECIFIC_NAME, 'ROUTINE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                 OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) )
    )
```

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine that is the polymorphic table function that contains the private parameter being described.

2) The value of ORDINAL_POSITION is the ordinal position of the private parameter in the SQL-invoked routine.

3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, and DTD_IDEN-TIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the private parameter.

4) Case:

   a) If <SQL parameter name> was specified when the polymorphic table function was created, then the value of PARAMETER_NAME is that <SQL parameter name>.

   b) Otherwise, the value of PARAMETER_NAME is the null value.

5) Case:

   a) If <parameter default> was specified when the polymorphic table function was created, then the value of PARAMETER_DEFAULT is that <parameter default>.

   b) Otherwise, the value of PARAMETER_DEFAULT is the null value.

6.36   REFERENCED_TYPES base table

Function

The REFERENCED_TYPES table has one row for each reference type. It effectively contains a representation of the referenced type descriptors.

Definition

```
CREATE TABLE REFERENCED_TYPES (
    OBJECT_CATALOG                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_TYPE_IDENTIFIER           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER                 INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT REFERENCED_TYPES_PRIMARY_KEY
      PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ),

    CONSTRAINT REFERENCED_TYPES_CHECK_REFERENCE_TYPE
      CHECK ( ( OBJECT_CATALOG, OBJECT_SCHEMA,
                OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR
                WHERE DATA_TYPE = 'REF' ) ),

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
      FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
    )
```

Description

1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and REFERENCE_TYPE_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the reference type whose referenced type is being described.

2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,

OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the referenced type of the reference type.

3) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ROOT_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the root data type of the reference type.

NOTE 16 — The root data type indicates the row of DATA_TYPE_DESCRIPTOR that directly belongs to some column, attribute, field, *etc.*. The root data type is used to identify the access rights a user has on a specific row of the base table REFERENCED_TYPES.

6.37 REFERENTIAL_CONSTRAINTS base table

Function

The REFERENTIAL_CONSTRAINTS table has one row for each row in the TABLE_CONSTRAINTS table that has a CONSTRAINT_TYPE value of "FOREIGN KEY".

Definition

```
CREATE TABLE REFERENTIAL_CONSTRAINTS (
    CONSTRAINT_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    UNIQUE_CONSTRAINT_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT UNIQUE_CONSTRAINT_CATALOG_NOT_NULL
        NOT NULL,
    UNIQUE_CONSTRAINT_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT UNIQUE_CONSTRAINT_SCHEMA_NOT_NULL
        NOT NULL,
    UNIQUE_CONSTRAINT_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT UNIQUE_CONSTRAINT_NAME_NOT_NULL
        NOT NULL,
    MATCH_OPTION                 INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT REFERENTIAL_MATCH_OPTION_NOT_NULL
        NOT NULL
      CONSTRAINT REFERENTIAL_MATCH_OPTION_CHECK
        CHECK ( MATCH_OPTION IN
            ( 'NONE', 'PARTIAL', 'FULL' ) ),
    UPDATE_RULE                  INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT REFERENTIAL_UPDATE_RULE_NOT_NULL
        NOT NULL
      CONSTRAINT REFERENTIAL_UPDATE_RULE_CHECK
        CHECK ( UPDATE_RULE IN
              ('CASCADE',
               'SET NULL',
               'SET DEFAULT',
               'RESTRICT',
               'NO ACTION' ) ),
    DELETE_RULE                  INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT REFERENTIAL_DELETE_RULE_NOT_NULL
        NOT NULL
      CONSTRAINT REFERENTIAL_DELETE_RULE_CHECK
        CHECK ( DELETE_RULE IN
              ('CASCADE',
               'SET NULL',
               'SET DEFAULT',
               'RESTRICT',
               'NO ACTION' ) ),

    CONSTRAINT REFERENTIAL_CONSTRAINTS_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

    CONSTRAINT REFERENTIAL_CONSTRAINTS_CONSTRAINT_TYPE_CHECK
```

```
      CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
              ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                FROM TABLE_CONSTRAINTS
                WHERE CONSTRAINT_TYPE = 'FOREIGN KEY' ) ),

  CONSTRAINT UNIQUE_CONSTRAINT_CHECK_REFERENCES_UNIQUE_CONSTRAINT
    CHECK ( UNIQUE_CONSTRAINT_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
          OR
              ( ( UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA,
                  UNIQUE_CONSTRAINT_NAME ) IN
                  ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                    FROM TABLE_CONSTRAINTS
                    WHERE CONSTRAINT_TYPE IN
                          ( 'UNIQUE', 'PRIMARY KEY' ) ) ) )
  )
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.

2) The values of UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA, and UNIQUE_CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the unique or primary key constraint applied to the referenced column list being described.

3) The values of MATCH_OPTION have the following meanings:

NONE	No <match type> was specified.
PARTIAL	A <match type> of PARTIAL was specified.
FULL	A <match type> of FULL was specified.

4) The values of UPDATE_RULE have the following meanings for a referential constraint that has an <update rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.
SET DEFAULT	A <referential action> of SET DEFAULT was specified.

5) The values of DELETE_RULE have the following meanings for a referential constraint that has a <delete rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.
SET DEFAULT	A <referential action> of SET DEFAULT was specified.

6.38   ROLE_AUTHORIZATION_DESCRIPTORS base table

Function

Contains a representation of the role authorization descriptors.

Definition

```
CREATE TABLE ROLE_AUTHORIZATION_DESCRIPTORS (
    ROLE_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTOR                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_GRANTABLE                      INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORS_IS_GRANTABLE_NOT_NULL
        NOT NULL,

    CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORS_PRIMARY_KEY
      PRIMARY KEY ( ROLE_NAME, GRANTEE, GRANTOR ),

    CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORS_CHECK_ROLE_NAME
      CHECK ( ROLE_NAME IN
              ( SELECT AUTHORIZATION_NAME
                FROM AUTHORIZATIONS
                WHERE AUTHORIZATION_TYPE = 'ROLE' ) ),

    CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORS_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
            OR
              GRANTOR IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) ),

    CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORS_GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
            OR
              GRANTEE IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) )
    )
```

Description

1) The value of ROLE_NAME is the <role name> of some <role granted> by the <grant role statement> or the <role name> of a <role definition>.

2) The value of GRANTEE is an <authorization identifier>, possibly PUBLIC, or <role name> specified as a <grantee> contained in a <grant role statement>, or the <authorization identifier> of the current SQL-session when the <role definition> is executed.

3) The value of GRANTOR is the <authorization identifier> of the user or role who granted the role identified by ROLE_NAME to the user or role identified by the value of GRANTEE, or "_SYSTEM" to indicate

that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

4)   The values of IS_GRANTABLE have the following meanings:

YES	The described role is grantable.
NO	The described role is not grantable.

A role is grantable if the WITH ADMIN OPTION is specified in the <grant role statement> or a <role definition> is executed.

6.39 ROUTINE_COLUMN_USAGE base table

Function

The ROUTINE_COLUMN_USAGE table has one row for each column identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINE_COLUMN_USAGE (
    SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_COLUMN_USAGE_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
      CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
            OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
                FROM COLUMNS ) ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_FOREIGN_KEY_ROUTINE_TABLE_USAGE
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME)
      REFERENCES ROUTINE_TABLE_USAGE
    )
```

Description

1) The ROUTINE_COLUMN_USAGE table has one row for each column *COL* of a table *TAB* identified by a column reference or column name contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *SIR* being described.

2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SIR*.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.

4) The value of COLUMN_NAME is the name of the column *COL*.

6.40 ROUTINE_PERIOD_USAGE base table

Function

The ROUTINE_PERIOD_USAGE table has one row for each period identified in the <SQL routine body> of an SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINE_PERIOD_USAGE (
    SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PERIOD_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_PERIOD_USAGE_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

    CONSTRAINT ROUTINE_PERIOD_USAGE_CHECK_REFERENCES_PERIODS
      CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
           OR
             ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ) IN
             ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME
               FROM PERIODS ) ),

    CONSTRAINT ROUTINE_PERIOD_USAGE_FOREIGN_KEY_ROUTINE_TABLE_USAGE
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES ROUTINE_TABLE_USAGE
)
```

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine *R* being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and PERIOD_NAME are the catalog name, unqualified schema name, and qualified identifier of the table name, and the period name, respectively, of the period that is identified in the <SQL routine body> of *R*.

6.41   ROUTINE_PRIVILEGES base table

Function

The ROUTINE_PRIVILEGES table has one row for each execute privilege descriptor for an SQL-invoked routine. It effectively contains a representation of the execute privilege descriptors.

Definition

```
CREATE TABLE ROUTINE_PRIVILEGES (
     GRANTOR               INFORMATION_SCHEMA.SQL_IDENTIFIER,
     GRANTEE               INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
     PRIVILEGE_TYPE        INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT ROUTINE_PRIVILEGES_TYPE_CHECK
         CHECK ( PRIVILEGE_TYPE IN
                 ( 'EXECUTE' ) ),
     IS_GRANTABLE          INFORMATION_SCHEMA.YES_OR_NO
        CONSTRAINT ROUTINE_PRIVILEGES_GRANTABLE_NOT_NULL
           NOT NULL,

     CONSTRAINT ROUTINE_PRIVILEGES_PRIMARY_KEY
       PRIMARY KEY ( GRANTOR, GRANTEE, SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
                     SPECIFIC_NAME, PRIVILEGE_TYPE ),

     CONSTRAINT ROUTINE_PRIVILEGES_FOREIGN_KEY_TABLES
       FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
       REFERENCES ROUTINES,

     CONSTRAINT ROUTINE_PRIVILEGES_GRANTOR_CHECK
       CHECK ( GRANTOR = '_SYSTEM'
           OR
             GRANTOR IN
           ( SELECT AUTHORIZATION_NAME
             FROM AUTHORIZATIONS ) ),

     CONSTRAINT ROUTINE_PRIVILEGES_GRANTEE_CHECK
       CHECK ( GRANTEE = 'PUBLIC'
           OR
             GRANTEE IN
           ( SELECT AUTHORIZATION_NAME
             FROM AUTHORIZATIONS ) )
   )
```

Description

1)   The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPE-CIFIC_NAME, to the user or role identified by the value of GRANTEE for the privilege being described,

or "_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the privilege being described is granted.

3) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine on which the privilege being described has been granted.

4) The values of PRIVILEGE_TYPE have the following meanings:

EXECUTE	The user has EXECUTE privilege on the SQL-invoked routine identified by SPE-CIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME.

5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.42   ROUTINE_ROUTINE_USAGE base table

Function

The ROUTINE_ROUTINE_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINE_ROUTINE_USAGE (
    SPECIFIC_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_ROUTINE_USAGE_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
      CHECK ( ROUTINE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
           OR
              ( ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME ) IN
              ( SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME
                FROM ROUTINES ) ),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_FOREIGN_KEY_ROUTINES
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

Description

1) The ROUTINE_ROUTINE_USAGE table has one row for each SQL-invoked routine *R1* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL routine *R2*.

2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R2*.

3) The values of ROUTINE_CATALOG, ROUTINE_SCHEMA, and ROUTINE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R1*.

6.43   ROUTINE_SEQUENCE_USAGE base table

Function

The ROUTINE_SEQUENCE_USAGE table has one row for each external sequence generator identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINE_SEQUENCE_USAGE (
    SPECIFIC_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_CHECK_REFERENCES_SEQUENCES
      CHECK ( SEQUENCE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
          OR
            ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) IN
              ( SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
                FROM SEQUENCES ) ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_FOREIGN_KEY_ROUTINES
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
    )
```

Description

1) The ROUTINE_SEQUENCE_USAGE table has one row for each sequence generator *SEQ* identified by a <sequence generator name> contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *R* being described.

2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

3) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, and SEQUENCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SEQ*.

6.44   ROUTINE_TABLE_USAGE base table

Function

The ROUTINE_TABLE_USAGE table has one row for each table identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINE_TABLE_USAGE (
    SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_TABLE_USAGE_PRIMARY_KEY
      PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT ROUTINE_TABLE_USAGE_CHECK_REFERENCES_TABLES
      CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
            OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM TABLES ) ),

    CONSTRAINT ROUTINE_TABLE_USAGE_FOREIGN_KEY_ROUTINES
      FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
  )
```

Description

1) The ROUTINE_TABLE_USAGE table has one row for each table *TAB* identified by a <table reference> contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *SIR* being described.

2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SIR*.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.

6.45   ROUTINES base table

This Subclause is modified by Subclause 20.5, "ROUTINES base table", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 15.5, "ROUTINES base table", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 22.3, "ROUTINES base table", in ISO/IEC 9075-14.

Function

The ROUTINES base table has one row for each SQL-invoked routine.

Definition

```
CREATE TABLE ROUTINES (
     SPECIFIC_CATALOG                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SPECIFIC_SCHEMA                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
     SPECIFIC_NAME                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
     ROUTINE_CATALOG                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
     ROUTINE_SCHEMA                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
     ROUTINE_NAME                           INFORMATION_SCHEMA.SQL_IDENTIFIER,
     MODULE_CATALOG                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
     MODULE_SCHEMA                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
     MODULE_NAME                            INFORMATION_SCHEMA.SQL_IDENTIFIER,
     USER_DEFINED_TYPE_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
     USER_DEFINED_TYPE_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
     USER_DEFINED_TYPE_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
     ROUTINE_TYPE                           INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT ROUTINE_TYPE_NOT_NULL
         NOT NULL
       CONSTRAINT ROUTINE_TYPE_CHECK
         CHECK ( ROUTINE_TYPE IN
                 ( 'PROCEDURE', 'FUNCTION', 'PTF',
                   'INSTANCE METHOD', 'STATIC METHOD', 'CONSTRUCTOR METHOD' ) ),
     DTD_IDENTIFIER                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
     ROUTINE_BODY                           INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT ROUTINE_BODY_NOT_NULL
         NOT NULL
       CONSTRAINT ROUTINE_BODY_CHECK
         CHECK ( ROUTINE_BODY IN
                 ( 'SQL', 'EXTERNAL', 'PTF' ) ),
     ROUTINE_DEFINITION                     INFORMATION_SCHEMA.CHARACTER_DATA,
     EXTERNAL_NAME                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
     EXTERNAL_LANGUAGE                      INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT EXTERNAL_LANGUAGE_CHECK
         CHECK ( EXTERNAL_LANGUAGE IN
                 ( 'ADA', 'C', 'COBOL',
                   'FORTRAN', 'MUMPS', 'PASCAL', 'PLI' ) ),
     PARAMETER_STYLE                        INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT PARAMETER_STYLE_CHECK
         CHECK ( PARAMETER_STYLE IN
                 ( 'SQL', 'GENERAL' ) ),
     IS_DETERMINISTIC                       INFORMATION_SCHEMA.YES_OR_NO
       CONSTRAINT ROUTINES_IS_DETERMINISTIC_NOT_NULL
         NOT NULL,
```

```
SQL_DATA_ACCESS                          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_NOT_NULL
    NOT NULL
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN
            ( 'NO SQL', 'CONTAINS SQL',
              'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
IS_NULL_CALL                             INFORMATION_SCHEMA.YES_OR_NO,
SQL_PATH                                 INFORMATION_SCHEMA.CHARACTER_DATA,
SCHEMA_LEVEL_ROUTINE                     INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT ROUTINES_SCHEMA_LEVEL_ROUTINE_NOT_NULL
    NOT NULL,
MAX_DYNAMIC_RESULT_SETS                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
IS_USER_DEFINED_CAST                     INFORMATION_SCHEMA.YES_OR_NO,
IS_IMPLICITLY_INVOCABLE                  INFORMATION_SCHEMA.YES_OR_NO,
SECURITY_TYPE                            INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT ROUTINES_SECURITY_TYPE_CHECK
    CHECK ( SECURITY_TYPE IN
            ( 'DEFINER', 'INVOKER', 'IMPLEMENTATION DEFINED' ) ),
TO_SQL_SPECIFIC_CATALOG                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_SCHEMA                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_NAME                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
AS_LOCATOR                               INFORMATION_SCHEMA.YES_OR_NO,
CREATED                                  INFORMATION_SCHEMA.TIME_STAMP,
LAST_ALTERED                             INFORMATION_SCHEMA.TIME_STAMP,
NEW_SAVEPOINT_LEVEL                      INFORMATION_SCHEMA.YES_OR_NO,
IS_UDT_DEPENDENT                         INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT ROUTINES_IS_UDT_DEPENDENT_NOT_NULL
    NOT NULL,
RESULT_CAST_FROM_DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR                   INFORMATION_SCHEMA.YES_OR_NO,


RETURNS_ONLY_PASS_THROUGH                INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT ROUTINES_RETURNS_ONLY_PASS_THROUGH_CHECK
    CHECK ( ( ROUTINE_TYPE = 'PTF'
          AND RETURNS_ONLY_PASS-THROUGH IS NOT NULL ),
        OR ( ROUTINE_TYPE <> 'PTF'
          AND RETURNS_ONLY_PASS-THROUGH IS NULL ) ),
DESCRIBE_PROCEDURE_SPECIFIC_CATALOG   INFORMATION_SCHEMA.IDENTIFIER,
DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA    INFORMATION_SCHEMA.IDENTIFIER,
DESCRIBE_PROCEDURE_SPECIFIC_NAME      INFORMATION_SCHEMA.IDENTIFIER,
START_PROCEDURE_SPECIFIC_CATALOG      INFORMATION_SCHEMA.IDENTIFIER,
START_PROCEDURE_SPECIFIC_SCHEMA       INFORMATION_SCHEMA.IDENTIFIER,
START_PROCEDURE_SPECIFIC_NAME         INFORMATION_SCHEMA.IDENTIFIER,
FULFILL_PROCEDURE_SPECIFIC_CATALOG    INFORMATION_SCHEMA.IDENTIFIER,
FULFILL_PROCEDURE_SPECIFIC_SCHEMA     INFORMATION_SCHEMA.IDENTIFIER,
FULFILL_PROCEDURE_SPECIFIC_NAME       INFORMATION_SCHEMA.IDENTIFIER,
FINISH_PROCEDURE_SPECIFIC_CATALOG     INFORMATION_SCHEMA.IDENTIFIER,
FINISH_PROCEDURE_SPECIFIC_SCHEMA      INFORMATION_SCHEMA.IDENTIFIER,
FINISH_PROCEDURE_SPECIFIC_NAME        INFORMATION_SCHEMA.IDENTIFIER,

CONSTRAINT ROUTINES_PRIMARY_KEY
  PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT ROUTINES_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( ROUTINE_CATALOG, ROUTINE_SCHEMA )
```

```
      REFERENCES SCHEMATA,

  CONSTRAINT ROUTINES_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME )
    REFERENCES USER_DEFINED_TYPES
      MATCH FULL,

  CONSTRAINT ROUTINES_COMBINATIONS
    CHECK ( ( ROUTINE_BODY IN ( 'SQL', 'PTF' )
            AND
              ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NULL )
          OR
              ( ROUTINE_BODY = 'EXTERNAL'
            AND
              ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NOT NULL ) ),

  CONSTRAINT ROUTINES_SAME_SCHEMA
    CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( ROUTINE_CATALOG, ROUTINE_SCHEMA )
          OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( MODULE_CATALOG, MODULE_SCHEMA )
          OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),

  CONSTRAINT ROUTINES_CHECK_RESULT_TYPE
    CHECK ( ( ROUTINE_TYPE ='PROCEDURE'
            AND
              DTD_IDENTIFIER IS NULL )
          OR
              ( ROUTINE_TYPE <>'PROCEDURE'
            AND
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', DTD_IDENTIFIER ) IN
                ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                         OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) ) ),

  CONSTRAINT ROUTINES_CHECK_RESULT_CAST
    CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
            AND
              RESULT_CAST_AS_LOCATOR IS NULL )
          OR
              ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
            AND
              RESULT_CAST_AS_LOCATOR IS NOT NULL
            AND
              ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                'ROUTINE', RESULT_CAST_FROM_DTD_IDENTIFIER ) IN
                ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                         OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) ) )

  CONSTRAINT ROUTINES_PTF_DESCRIBE_FOREIGN_KEY
    FOREIGN KEY ( DESCRIBE_PROCEDURE_SPECIFIC_CATALOG,
                  DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA,
                  DESCRIBE_PROCEDURE_SPECIFIC_NAME )
```

```
        REFERENCES ROUTINES,

  CONSTRAINT ROUTINES_PTF_DESCRIBE_CHECK
    CHECK ( ROUTINE_TYPE = 'PTF'
        OR
              ( DESCRIBE_PROCEDURE_SPECIFIC_CATALOG,
                DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA,
                DESCRIBE_PROCEDURE_SPECIFIC_NAME ) IS NULL ),

  CONSTRAINT ROUTINES_PTF_START_FOREIGN_KEY
    FOREIGN KEY ( START_PROCEDURE_SPECIFIC_CATALOG,
                  START_PROCEDURE_SPECIFIC_SCHEMA,
                  START_PROCEDURE_SPECIFIC_NAME )
    REFERENCES ROUTINES,

  CONSTRAINT ROUTINES_PTF_START_CHECK
    CHECK ( ROUTINE_TYPE = 'PTF'
        OR
              ( START_PROCEDURE_SPECIFIC_CATALOG,
                START_PROCEDURE_SPECIFIC_SCHEMA,
                START_PROCEDURE_SPECIFIC_NAME ) IS NULL ),

  CONSTRAINT ROUTINES_PTF_FULFILL_FOREIGN_KEY
    FOREIGN KEY ( FULFILL_PROCEDURE_SPECIFIC_CATALOG,
                  FULFILL_PROCEDURE_SPECIFIC_SCHEMA,
                  FULFILL_PROCEDURE_SPECIFIC_NAME )
    REFERENCES ROUTINES,

  CONSTRAINT ROUTINES_PTF_FULFILL_CHECK
    CHECK ( ( ROUTINE_TYPE = 'PTF'
        AND
              ( FULFILL_PROCEDURE_SPECIFIC_CATALOG,
                FULFILL_PROCEDURE_SPECIFIC_SCHEMA,
                FULFILL_PROCEDURE_SPECIFIC_NAME ) IS NOT NULL ),
        OR
              ( ROUTINE_TYPE <> 'PTF'
        AND
              ( FULFILL_PROCEDURE_SPECIFIC_CATALOG,
                FULFILL_PROCEDURE_SPECIFIC_SCHEMA,
                FULFILL_PROCEDURE_SPECIFIC_NAME ) IS NULL ) ),

  CONSTRAINT ROUTINES_PTF_FINISH_FOREIGN_KEY
    FOREIGN KEY ( FINISH_PROCEDURE_SPECIFIC_CATALOG,
                  FINISH_PROCEDURE_SPECIFIC_SCHEMA,
                  FINISH_PROCEDURE_SPECIFIC_NAME )
    REFERENCES ROUTINES,

  CONSTRAINT ROUTINES_PTF_FINISH_CHECK
    CHECK ( ROUTINE_TYPE = 'PTF'
        OR
              ( FINISH_PROCEDURE_SPECIFIC_CATALOG,
                FINISH_PROCEDURE_SPECIFIC_SCHEMA,
                FINISH_PROCEDURE_SPECIFIC_NAME ) IS NULL )
```

)

Description

1) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine being described.

2) The values of ROUTINE_CATALOG, ROUTINE_SCHEMA, and ROUTINE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the routine name of the SQL-invoked routine being described.

3) 04 The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the null value.

4) Case:

   a) If the SQL-invoked routine being described was defined as a method of a user-defined type, then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of this user-defined type.

   b) Otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

5) The values of ROUTINE_TYPE have the following meanings:

PROCE-DURE	The SQL-invoked routine being described is an SQL-invoked procedure.
FUNCTION	The SQL-invoked routine being described is an SQL-invoked function that is not an SQL-invoked method or a polymorphic table function.
INSTANCE METHOD	The SQL-invoked routine being described is an SQL-invoked method that is neither a static SQL-invoked method nor an SQL-invoked constructor method.
STATIC METHOD	The SQL-invoked routine being described is a static SQL-invoked method.
CONSTRUC-TOR METHOD	The SQL-invoked routine being described is an SQL-invoked constructor method.
PTF	The SQL-invoked routine being described is a polymorphic table function.

6) If the SQL-invoked routine being described is an SQL-invoked procedure, then DTD_IDENTIFIER is the null value; otherwise, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the result type of the SQL-invoked routine being described.

7) The values of ROUTINE_BODY have the following meanings:

SQL	The SQL-invoked routine being described is an SQL routine.
EXTERNAL	The SQL-invoked routine being described is an external routine.
PTF	The SQL routine being described is a polymorphic table function.

8) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

9) The values of IS_DETERMINISTIC have the following meanings:

YES	DETERMINISTIC was specified when the SQL-invoked routine was defined.
NO	DETERMINISTIC was not specified when the SQL-invoked routine was defined.

10) The values of IS_NULL_CALL have the following meanings:

YES	The SQL-invoked routine is a function and returns null if any of its parameters are null.
NO	The SQL-invoked routine is a function and its return value is determined by invoking the routine.
null	The routine being described is a procedure.

11) Case:

    a) If the SQL-invoked routine being described is an SQL routine, and the SQL-invoked routine is not contained in an SQL-server module definition, and the character representation of the <routine body> that defined the SQL-invoked routine can be represented without truncation, then the value of ROUTINE_DEFINITION is that character representation.

    b) Otherwise, the value of ROUTINE_DEFINITION is the null value.

12) Case:

    a) If the SQL-invoked routine being described is an external routine, then:

        i) The value of EXTERNAL_NAME is the external name of the external routine.

ii)   The value of EXTERNAL_LANGUAGE is the language of the external routine.

iii) The value of PARAMETER_STYLE is the SQL parameter passing style of the external routine.

b)   Otherwise, the values of EXTERNAL_NAME, EXTERNAL_LANGUAGE and PARAMETER_STYLE are the null value.

13) Case:

a)   If the routine being described is an SQL routine, then the value of SQL_PATH is the SQL-path of the routine being described.

b)   Otherwise, the value of SQL_PATH is the null value.

14) Case:

a)   If the SQL-invoked routine is a schema-level routine, then the value of SCHEMA_LEVEL_ROUTINE is 'YES'.

b)   Otherwise, the value of SCHEMA_LEVEL_ROUTINE is 'NO'.

15) The value of MAX_DYNAMIC_RESULT_SETS is

Case:

a)   If the routine being described is an SQL-invoked procedure defined by an <SQL-invoked routine> for which <maximum returned result sets> was specified, then the value of <maximum returned result sets>.

b)   Otherwise, 0 (zero).

16) The values of IS_USER_DEFINED_CAST have the following meanings:

YES	The SQL-invoked routine is a function that is a user-defined cast function.
NO	The function SQL-invoked routine is a function that is not a user-defined cast function.
null	The SQL-invoked routine is a procedure.

17) The values of IS_IMPLICITLY_INVOCABLE have the following meanings:

YES	The user-defined cast function is implicitly invocable.
NO	The user-defined cast function is not implicitly invocable.
null	The routine is not a user-defined cast function.

18) The values of SECURITY_TYPE have the following meanings:

DEFINER	The routine has the security characteristic DEFINER.
INVOKER	The routine has the security characteristic INVOKER.

IMPLEMEN-TATION DEFINED	The external routine has the security characteristic IMPLEMENTATION DEFINED.
null	Either the SQL-invoked routine is not a polymorphic table function, or it is an external routine and Feature T324, "Explicit security for SQL routines" is not implemented.

19) If the SQL-invoked routine being described is an external routine and has a to-sql routine for its result type, then TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA and TO_SQL_SPE-CIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked routine being described.

20) The values of AS_LOCATOR have the following meanings:

YES	The return value of the SQL-invoked routine being described is passed AS LOCATOR.
NO	The return value of the SQL-invoked routine being described is not passed AS LOCATOR.
null	The SQL-invoked routine is a procedure.

21) If Feature T272, "Enhanced savepoint management", is not implemented, then the value of NEW_SAVEPOINT_LEVEL is null; otherwise, the values of NEW_SAVEPOINT_LEVEL have the following meanings:

YES	The SQL-invoked routine is an SQL-invoked function or is an SQL-invoked procedure that specifies NEW SAVEPOINT LEVEL.
NO	The SQL-invoked routine is an SQL-invoked procedure that does not specify NEW SAVEPOINT LEVEL or specifies OLD SAVEPOINT LEVEL.

22) The values of IS_UDT_DEPENDENT have the following meanings:

YES	The SQL-invoked routine being described is dependent on a user-defined type.
NO	The SQL-invoked routine being described is not dependent on a user-defined type.

23) Case:

   a) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the SQL-invoked routine specifies a <result cast>, then the values of RESULT_CAST_FROM_DTD_IDENTIFIER and RESULT_CAST_AS_LOCATOR are the null value.

   b) Otherwise, SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and RESULT_CAST_FROM_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the <data type> specified in the <result cast> of the SQL-invoked routine being described.

Case:

i) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of RESULT_CAST_AS_LOCATOR is 'NO'.

ii) Otherwise, the value of RESULT_CAST_AS_LOCATOR is 'YES'.

24) The value of CREATED is

Case:

a) If Feature T011, "Timestamp in Information Schema", is supported and the SQL-implementation knows the value of CURRENT_TIMESTAMP at the time when the SQL-invoked routine being described was created, then that value.

b) Otherwise, the NULL value.

25) The value of LAST_ALTERED is

Case:

a) If Feature T011, "Timestamp in Information Schema", is supported and the SQL-implementation knows the value of CURRENT_TIMESTAMP at the time when the SQL-invoked routine being described was last altered, then that value.

b) Otherwise, the NULL value.

This value is identical to the value of CREATED for SQL-invoked routines that have never been altered.

26) The values of RETURNS_ONLY_PASS_THROUGH have the following meanings:

YES	The routine is a polymorphic table function that specifies RETURNS ONLY PASS THROUGH.
NO	The routine is a polymorphic table function that does not specify RETURNS ONLY PASS THROUGH.
null	The routine is not a polymorphic table function.

27) If the routine being described is a polymorphic table function that has a PTF describe component procedure *P*, then DESCRIBE_PROCEDURE_SPECIFIC_CATALOG, DESCRIBE_PROCEDURE_SPE-CIFIC_SCHEMA, and DESCRIBE_PROCEDURE_SPECIFIC_NAME are the catalog name, schema name, and specific name of *P*; otherwise, they are the null value.

28) If the routine being described is a polymorphic table function that has a PTF start component procedure *P*, then START_PROCEDURE_SPECIFIC_CATALOG, START_PROCEDURE_SPECIFIC_SCHEMA, and START_PROCEDURE_SPECIFIC_NAME are the catalog name, schema name, and specific name of *P*; otherwise, they are the null value.

29) If the routine being described is a polymorphic table function that has a PTF fulfill component procedure *P*, then FULFILL_PROCEDURE_SPECIFIC_CATALOG, FULFILL_PROCEDURE_SPE-CIFIC_SCHEMA, and FULFILL_PROCEDURE_SPECIFIC_NAME are the catalog name, schema name, and specific name of *P*; otherwise, they are the null value.

30) If the routine being described is a polymorphic table function that has a PTF finish component procedure *P*, then FINISH_PROCEDURE_SPECIFIC_CATALOG, FINISH_PROCEDURE_SPECIFIC_SCHEMA, and FINISH_PROCEDURE_SPECIFIC_NAME are the catalog name, schema name, and specific name of *P*; otherwise, they are the null value.

6.46 SCHEMATA base table

Function

The SCHEMATA table has one row for each schema.

Definition

```
CREATE TABLE SCHEMATA (
    CATALOG_NAME                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCHEMA_NAME                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCHEMA_OWNER                     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT SCHEMA_OWNER_NOT_NULL
        NOT NULL,
    DEFAULT_CHARACTER_SET_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_CHARACTER_SET_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL
        NOT NULL,
    DEFAULT_CHARACTER_SET_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL
        NOT NULL,
    SQL_PATH                         INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SCHEMATA_PRIMARY_KEY
      PRIMARY KEY ( CATALOG_NAME, SCHEMA_NAME ),

    CONSTRAINT SCHEMATA_FOREIGN_KEY_AUTHORIZATIONS
      FOREIGN KEY ( SCHEMA_OWNER )
      REFERENCES AUTHORIZATIONS,

    CONSTRAINT SCHEMATA_FOREIGN_KEY_CATALOG_NAME
      FOREIGN KEY ( CATALOG_NAME )
      REFERENCES CATALOG_NAME,

    CONSTRAINT SCHEMATA_CHECK_REFERENCES_CHARACTER_SETS
      CHECK ( DEFAULT_CHARACTER_SET_CATALOG NOT IN
          ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
          ( DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
          DEFAULT_CHARACTER_SET_NAME ) IN
          ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                   CHARACTER_SET_NAME
            FROM CHARACTER_SETS ) )
    )
```

Description

1)   The value of CATALOG_NAME is the name of the catalog of the schema described by this row.

2)   The value of SCHEMA_NAME is the unqualified schema name of the schema described by this row.

3)   The values of SCHEMA_OWNER are the authorization identifiers that own the schemata.

4)   The values of DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, and DEFAULT_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default character set for columns and domains in the schemata.

5)   Case:

   a)   If <schema path specification> was specified in the <schema definition> that defined the schema described by this row and the character representation of the <schema path specification> can be represented without truncation, then the value of SQL_PATH is that character representation.

   b)   Otherwise, the value of SQL_PATH is the null value.

6.47 SEQUENCES base table

Function

The SEQUENCES base table has one row for each external sequence generator.

Definition

```
CREATE TABLE SEQUENCES (
    SEQUENCE_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    START_VALUE                INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT SEQUENCES_START_VALUE_NOT_NULL
        NOT NULL,
    MINIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT SEQUENCES_MINIMUM_VALUE_NOT_NULL
        NOT NULL,
    MAXIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT SEQUENCES_MAXIMUM_VALUE_NOT_NULL
        NOT NULL,
    INCREMENT                  INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT SEQUENCES_INCREMENT_NOT_NULL
        NOT NULL,
    CYCLE_OPTION               INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT SEQUENCES_CYCLE_OPTION_NOT_NULL
        NOT NULL,

    CONSTRAINT SEQUENCES_PRIMARY_KEY
      PRIMARY KEY (SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME),

    CONSTRAINT SEQUENCES_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA )
      REFERENCES SCHEMATA,

    CONSTRAINT SEQUENCES_CHECK_DATA_TYPE
      CHECK ( ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA,
              SEQUENCE_NAME, 'SEQUENCE', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR ) )
)
```

Description

1) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, and SEQUENCE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the sequence generator being described.

2) The values of SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the data type of the sequence generator.

3) The values of START_VALUE, MINIMUM_VALUE, MAXIMUM_VALUE, and INCREMENT are the character representations of start value, minimum value, maximum value, and increment, respectively, of the sequence generator being described.

4) The values of CYCLE_OPTION have the following meanings:

YES	The cycle option of the sequence generator is CYCLE.
NO	The cycle option of the sequence generator is NO CYCLE.

6.48   SQL_CONFORMANCE base table

This Subclause is modified by Subclause 9.1, "SQL_CONFORMANCE base table", in ISO/IEC 9075-3.
This Subclause is modified by Subclause 20.6, "SQL_CONFORMANCE base table", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 25.11, "SQL_CONFORMANCE base table", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 15.1, "SQL_CONFORMANCE base table", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 15.6, "SQL_CONFORMANCE base table", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 22.4, "SQL_CONFORMANCE base table", in ISO/IEC 9075-14.

Function

The SQL_CONFORMANCE base table has one row for each conformance element (part, feature, and subfeature) identified by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_CONFORMANCE
    (
      TYPE                            INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT SQL_CONFORMANCE_TYPE_CHECK
          CHECK ( TYPE IN ( 'PART', 'FEATURE', 'SUBFEATURE' ) ),
      ID                              INFORMATION_SCHEMA.CHARACTER_DATA,
      SUB_ID                          INFORMATION_SCHEMA.CHARACTER_DATA,
      NAME                            INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT SQL_CONFORMANCE_NAME_NOT_NULL
          NOT NULL,
      SUB_NAME                        INFORMATION_SCHEMA.CHARACTER_DATA,
      IS_SUPPORTED                    INFORMATION_SCHEMA.YES_OR_NO
        CONSTRAINT SQL_CONFORMANCE_IS_SUPPORTED_NOT_NULL
          NOT NULL,
      IS_VERIFIED_BY                  INFORMATION_SCHEMA.CHARACTER_DATA,
      COMMENTS                        INFORMATION_SCHEMA.CHARACTER_DATA,

      CONSTRAINT SQL_CONFORMANCE_PRIMARY_KEY
        PRIMARY KEY ( TYPE, ID, SUB_ID ),

      CONSTRAINT SQL_CONFORMANCE_CHECK_SUPPORTED_VERIFIED
        CHECK ( IS_SUPPORTED = 'YES'
              OR
                IS_VERIFIED_BY IS NULL )
    )
```

Description

1)   The values of TYPE have the following meanings:

PART	the conformance element described is a Part of ISO/IEC 9075.
FEATURE	the conformance element described is an optional feature of ISO/IEC 9075.

SUBFEA-TURE	the conformance element described is a subfeature of an optional feature of ISO/IEC 9075.

2) The ID and NAME columns identify the conformance element described.

3) If the conformance element is a subfeature, then the SUB_ID and SUB_NAME columns identify the sub-feature by the subfeature identifier and name assigned to it. Otherwise, the values of SUB_ID and SUB_NAME are each a character string consisting of a single space.

4) ⬚03 ⬚04 ⬚09 ⬚10 ⬚13 ⬚14 The IS_SUPPORTED column is 'YES' if an SQL-implementation fully supports that confor-mance element described when SQL-data in the identified catalog is accessed through that implementation and is 'NO' if the SQL-implementation does not fully support that conformance element described when accessing SQL-data in that catalog.

5) ⬚03 ⬚04 ⬚09 ⬚10 ⬚13 ⬚14 If full support for the conformance element described has been verified by testing, then the IS_VERIFIED_BY column shall contain information identifying the conformance test used to verify the conformance claim; otherwise, IS_VERIFIED_BY shall be the null value.

6) If the value of the IS_SUPPORTED column for a feature is 'YES' and if that feature has subfeatures,then the value of the IS_SUPPORTED column in every row identifying subfeatures of the feature shall also be 'YES'.

7) ⬚03 ⬚04 ⬚09 ⬚10 ⬚13 ⬚14 The COMMENTS column contains any implementer comments pertinent to the identified SQL part, feature, or subfeature.

Table Population

1) There is one row in this table for every Part defined in ISO/IEC 9075. In each such row:

a) TYPE is 'PART'.

b) ID is the number of the Part of ISO/IEC 9075.

c) NAME is the name of the Part as given on the title page of that Part.

d) SUB_ID and SUB_NAME are each a character string consisting of a single space.

e) IS_SUPPORTED, IS_VERIFIED_BY, and COMMENTS are as described by Description 4), Description 5), and Description 7).

2) There is one row in this table for every row in Table 2, "Feature taxonomy and definition for mandatory features". In each such row:

a) If "Feature ID" in Table 2, "Feature taxonomy and definition for mandatory features", does not contains a <minus sign> (–), then TYPE is 'FEATURE'; otherwise TYPE is 'SUBFEATURE'.

b) ID is the value of "Feature ID", discarding any <minus sign> (–) and characters that follow it, from Table 2, "Feature taxonomy and definition for mandatory features".

c) NAME is the value of "Feature Name" from the row in Table 2, "Feature taxonomy and definition for mandatory features" whose "Feature ID" has the same value as ID.

  d) If TYPE is 'SUBFEATURE', then SUB_ID and SUB_NAME are the values of "Feature ID" and "Feature Name", respectively, from Table 2, "Feature taxonomy and definition for mandatory features"; otherwise, each is a character string consisting of a single space.

  e) IS_SUPPORTED, IS_VERIFIED_BY, and COMMENTS are as described by Description 4), Description 5), and Description 7).

3) There is one row in this table for every row in Table 3, "Feature taxonomy for optional features". In each such row:

  a) TYPE is 'FEATURE'.

  b) ID is the value of "Feature ID" from Table 3, "Feature taxonomy for optional features".

  c) NAME is the value of "Feature Name" from Table 3, "Feature taxonomy for optional features".

  d) SUB_ID and SUB_NAME are each a character string consisting of a single space.

  e) IS_SUPPORTED, IS_VERIFIED_BY, and COMMENTS are as described by Description 4), Description 5), and Description 7).

4) There is one row in this table for every row in Table 43, "Feature taxonomy and definition for mandatory features", in [ISO9075-2]. In each such row:

  a) If "Feature ID" in Table 43, "Feature taxonomy and definition for mandatory features", in [ISO9075-2], does not contains a <minus sign> (–), then TYPE is 'FEATURE'; otherwise TYPE is 'SUBFEATURE'.

  b) ID is the value of "Feature ID", discarding any <minus sign> (–) and characters that follow it, from Table 43, "Feature taxonomy and definition for mandatory features", in [ISO9075-2].

  c) NAME is the value of "Feature Name" from the row in Table 43, "Feature taxonomy and definition for mandatory features", in [ISO9075-2] whose "Feature ID" has the same value as ID.

  d) If TYPE is 'SUBFEATURE', then SUB_ID and SUB_NAME are the values of "Feature ID" and "Feature Name", respectively, from Table 43, "Feature taxonomy and definition for mandatory features", in [ISO9075-2]; otherwise, each is a character string consisting of a single space.

  e) IS_SUPPORTED, IS_VERIFIED_BY, and COMMENTS are as described by Description 4), Description 5), and Description 7).

5) There is one row in this table for every row in Table 44, "Feature taxonomy for optional features", in [ISO9075-2]. In each such row:

  a) TYPE is 'FEATURE'.

  b) ID is the value of "Feature ID" from Table 44, "Feature taxonomy for optional features", in [ISO9075-2].

  c) NAME is the value of "Feature Name" from Table 44, "Feature taxonomy for optional features", in [ISO9075-2].

  d) SUB_ID and SUB_NAME are each a character string consisting of a single space.

  e) IS_SUPPORTED, IS_VERIFIED_BY, and COMMENTS are as described by Description 4), Description 5), and Description 7).

6.49 SQL_IMPLEMENTATION_INFO base table

This Subclause is modified by Subclause 9.2, "SQL_IMPLEMENTATION_INFO base table", in ISO/IEC 9075-3.

Function

The SQL_IMPLEMENTATION_INFO base table has one row for each SQL-implementation information item defined by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_IMPLEMENTATION_INFO (
      IMPLEMENTATION_INFO_ID                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
      IMPLEMENTATION_INFO_NAME                INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT SQL_IMPLEMENTATION_INFO_NAME_NOT_NULL
          NOT NULL,
      INTEGER_VALUE                           INFORMATION_SCHEMA.CARDINAL_NUMBER,
      CHARACTER_VALUE                         INFORMATION_SCHEMA.CHARACTER_DATA,
      COMMENTS                                INFORMATION_SCHEMA.CHARACTER_DATA,

      CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
        PRIMARY KEY ( IMPLEMENTATION_INFO_ID ),

      CONSTRAINT SQL_IMPLEMENTATION_INFO_CHECK_INTEGER_EXCLUDES_CHARACTER
        CHECK ( INTEGER_VALUE IS NULL
            OR
              CHARACTER_VALUE IS NULL )
  )
```

Description

1) The SQL_IMPLEMENTATION_INFO table consists of exactly one row for each SQL-implementation information item defined in ISO/IEC 9075.

2) The IMPLEMENTATION_INFO_ID and IMPLEMENTATION_INFO_NAME columns identify the SQL-implementation information item by the integer and name assigned to it.

3) Depending on the type of information, the value is present in either INTEGER_VALUE or CHARAC-TER_VALUE; the other column is the null value.

4) The COMMENTS column is intended for any implementer comments pertinent to the identified item.

Table Population

None.

6.50 SQL_SIZING base table

This Subclause is modified by Subclause 9.3, "SQL_SIZING base table", in ISO/IEC 9075-3.

Function

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_SIZING (
     SIZING_ID                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
     SIZING_NAME                INFORMATION_SCHEMA.CHARACTER_DATA
       CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL
         NOT NULL,
     SUPPORTED_VALUE            INFORMATION_SCHEMA.CARDINAL_NUMBER,
     COMMENTS                   INFORMATION_SCHEMA.CHARACTER_DATA,

     CONSTRAINT SQL_SIZING_PRIMARY_KEY
       PRIMARY KEY (SIZING_ID ),

     CONSTRAINT SQL_SIZING_SIZING_NAME_UNIQUE
       UNIQUE ( SIZING_NAME )
   )
```

Description

1) The SQL_SIZING table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075.

2) The SIZING_ID and SIZING_NAME columns identify the sizing item by the integer and description assigned to it.

3) The values of the SUPPORTED_VALUE column are:

0	The SQL-implementation either places no limit on this sizing item or the SQL-implementation cannot determine the limit.
null	The SQL-implementation does not support any features for which this sizing item is applicable.
Any other value	The maximum size supported by the SQL-implementation for this sizing item.

4) The COMMENTS column is intended for any implementor comments pertinent to the identified SQL sizing item.

Table Population

None.

6.51   TABLE_CONSTRAINTS base table

Function

The TABLE_CONSTRAINTS table has one row for each table constraint associated with a table. It effectively contains a representation of the table constraint descriptors.

Definition

```
CREATE TABLE TABLE_CONSTRAINTS (
    CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT_TYPE         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT CONSTRAINT_TYPE_NOT_NULL
      NOT NULL
      CONSTRAINT CONSTRAINT_TYPE_CHECK
        CHECK ( CONSTRAINT_TYPE IN
                ( 'UNIQUE', 'PRIMARY KEY',
                  'FOREIGN KEY', 'CHECK' ) ),
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TABLE_CONSTRAINTS_TABLE_CATALOG_NOT_NULL
        NOT NULL,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TABLE_CONSTRAINTS_TABLE_SCHEMA_NOT_NULL
        NOT NULL,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TABLE_CONSTRAINTS_TABLE_NAME_NOT_NULL
        NOT NULL,
    IS_DEFERRABLE           INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT TABLE_CONSTRAINTS_IS_DEFERRABLE_NOT_NULL
        NOT NULL,
    INITIALLY_DEFERRED      INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT TABLE_CONSTRAINTS_INITIALLY_DEFERRED_NOT_NULL
        NOT NULL,
    ENFORCED                INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT TABLE_CONSTRAINTS_ENFORCED_NOT_NULL
        NOT NULL,

    CONSTRAINT TABLE_CONSTRAINTS_PRIMARY_KEY
      PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ),

    CONSTRAINT TABLE_CONSTRAINTS_DEFERRED_CHECK
      CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED ) IN
              ( VALUES ( 'NO',  'NO' ),
                       ( 'YES', 'NO' ),
                       ( 'YES', 'YES' ) ) ),

    CONSTRAINT TABLE_CONSTRAINTS_CHECK_VIEWS
      CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
            OR
```

```
        ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES
              WHERE TABLE_TYPE <> 'VIEW' ) ) ),

CONSTRAINT UNIQUE_TABLE_PRIMARY_KEY_CHECK
   CHECK ( UNIQUE ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                    FROM TABLE_CONSTRAINTS
                    WHERE CONSTRAINT_TYPE = 'PRIMARY KEY' ) )
)
```

Description

1) The values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described. If the <table constraint definition> or <add table constraint definition> that defined the constraint did not specify a <constraint name>, then the values of CONSTRAINT_CATALOG, CON-STRAINT_SCHEMA, and CONSTRAINT_NAME are implementation-defined.

2) The values of CONSTRAINT_TYPE have the following meanings:

FOREIGN KEY	The constraint being described is a foreign key constraint.
UNIQUE	The constraint being described is a unique constraint.
PRIMARY KEY	The constraint being described is a primary key constraint.
CHECK	The constraint being described is a check constraint.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, the unqualified schema name, and the qualified identifier of the name of the table to which the table constraint being described applies.

4) The values of IS_DEFERRABLE have the following meanings:

YES	The table constraint is deferrable.
NO	The table constraint is not deferrable.

5) The values of INITIALLY_DEFERRED have the following meanings:

YES	The table constraint is initially deferred.
NO	The table constraint is initially immediate.

6) The values of ENFORCED have the following meanings:

YES	The table constraint is enforced.
NO	The table constraint is not enforced.

6.52   TABLE_METHOD_PRIVILEGES base table

Function

The TABLE_METHOD_PRIVILEGES base table has one row for each table/method privilege descriptor. It effectively contains a representation of the table/method privilege descriptors.

Definition

```
CREATE TABLE TABLE_METHOD_PRIVILEGES (
      GRANTOR                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
      GRANTEE                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
      IS_GRANTABLE            INFORMATION_SCHEMA.YES_OR_NO
        CONSTRAINT TABLE_METHOD_PRIVILEGES_IS_GRANTABLE_NOT_NULL
            NOT NULL,

      CONSTRAINT TABLE_METHOD_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

      CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
          REFERENCES TABLES,

      CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
          REFERENCES ROUTINES,

      CONSTRAINT TABLE_METHOD_PRIVILEGES_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
              OR
                GRANTOR IN
              ( SELECT AUTHORIZATION_NAME
                FROM AUTHORIZATIONS ) ),

      CONSTRAINT TABLE_METHOD_PRIVILEGES_GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
              OR
                GRANTEE IN
              ( SELECT AUTHORIZATION_NAME
                FROM AUTHORIZATIONS ) )
```

)

Description

1) The value of GRANTOR is the <authorization identifier> of the user or role who granted a table/method privilege on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME, and the method of the identified table's structured type identified by the SPECIFIC_CATALOG, SPE-CIFIC_SCHEMA, and SPECIFIC_NAME, to the user or role identified by the value of GRANTEE for the table/method privilege being described, or "_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the table/method privilege being described is granted.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described was granted.

4) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the method on which the privilege being described was granted.

5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6.53 TABLE_PRIVILEGES base table

Function

The TABLE_PRIVILEGES table has one row for each table privilege descriptor. It effectively contains a representation of the table privilege descriptors.

Definition

```
CREATE TABLE TABLE_PRIVILEGES (
    GRANTOR                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE          INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT TABLE_PRIVILEGES_TYPE_CHECK
        CHECK ( PRIVILEGE_TYPE IN
                ( 'SELECT', 'INSERT', 'DELETE', 'UPDATE',
                  'TRIGGER', 'REFERENCES' ) ),
    IS_GRANTABLE            INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT TABLE_PRIVILEGES_GRANTABLE_NOT_NULL
        NOT NULL,
    WITH_HIERARCHY          INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT TABLE_PRIVILEGES_WITH_HIERARCHY_NOT_NULL
          NOT NULL,

    CONSTRAINT TABLE_PRIVILEGES_PRIMARY_KEY
      PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                    PRIVILEGE_TYPE ),

    CONSTRAINT TABLE_PRIVILEGES_FOREIGN_KEY_TABLES
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TABLES,

    CONSTRAINT TABLE_PRIVILEGES_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
              OR
              GRANTOR IN
              ( SELECT AUTHORIZATION_NAME
                FROM AUTHORIZATIONS ) ),

    CONSTRAINT TABLE_PRIVILEGES_GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
              OR
                GRANTEE IN
              ( SELECT AUTHORIZATION_NAME
                FROM AUTHORIZATIONS ) )
```

)

Description

1) The value of GRANTOR is the <authorization identifier> of the user or role who granted table privileges, on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME, to the user or role identified by the value of GRANTEE for the table privilege being described, or "_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the table privilege being described is granted.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described has been granted.

4) The values of PRIVILEGE_TYPE have the following meanings:

SELECT	The user has SELECT privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
DELETE	The user has DELETE privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
INSERT	The user will automatically be granted INSERT privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
UPDATE	The user will automatically be granted UPDATE privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
REFER-ENCES	The user will automatically be granted REFERENCES privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
TRIGGER	The user has TRIGGER privilege on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.

5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

6) The values of WITH_HIERARCHY have the following meanings:

YES	The privilege being described was granted WITH HIERARCHY OPTION and is thus grantable.
NO	The privilege being described was not granted WITH HIERARCHY OPTION and is thus not grantable.

6.54 TABLES base table

This Subclause is modified by Subclause 25.13, "TABLES base table", in ISO/IEC 9075-9.

Function

The TABLES table contains one row for each table including views. It effectively contains a representation of the table descriptors.

Definition

```
CREATE TABLE TABLES (
    TABLE_CATALOG                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_TYPE                       INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT TABLE_TYPE_NOT_NULL
        NOT NULL
      CONSTRAINT TABLE_TYPE_CHECK
        CHECK ( TABLE_TYPE IN
              ( 'BASE TABLE', 'VIEW', 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY',
                'SYSTEM VERSIONED' ) ),
    SELF_REFERENCING_COLUMN_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_GENERATION             INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT REFERENCE_GENERATION_CHECK
        CHECK ( REFERENCE_GENERATION IN
              ( 'SYSTEM GENERATED', 'USER GENERATED', 'DERIVED' ) ),
    USER_DEFINED_TYPE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_INSERTABLE_INTO               INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT IS_INSERTABLE_INTO_NOT_NULL
        NOT NULL,
    IS_TYPED                         INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT IS_TYPED_NOT_NULL
        NOT NULL,
    COMMIT_ACTION                    INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT COMMIT_ACTIONCHECK
        CHECK ( COMMIT_ACTION IN
              ( 'DELETE', 'PRESERVE' ) ),

    CONSTRAINT TABLES_PRIMARY_KEY
      PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT TABLES_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT TABLES_CHECK_TABLE_IN_COLUMNS
      CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM COLUMNS ) ),
```

```
      CONSTRAINT TABLES_FOREIGN_KEY_USER_DEFINED_TYPES
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME )
          REFERENCES USER_DEFINED_TYPES MATCH FULL,

      CONSTRAINT TABLES_TYPED_TABLE_CHECK
        CHECK ( ( IS_TYPED = 'YES'
                AND
                  ( ( USER_DEFINED_TYPE_CATALOG,
                      USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME,
                      SELF_REFERENCING_COLUMN_NAME,
                      REFERENCE_GENERATION ) IS NOT NULL
                  AND
                    ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      SELF_REFERENCING_COLUMN_NAME ) IN
                      ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                               COLUMN_NAME
                        FROM COLUMNS
                        WHERE IS_SELF_REFERENCING = 'YES' ) ) )
              OR
                ( IS_TYPED = 'NO'
                AND
                  ( USER_DEFINED_TYPE_CATALOG,
                    USER_DEFINED_TYPE_SCHEMA,
                    USER_DEFINED_TYPE_NAME,
                    SELF_REFERENCING_COLUMN_NAME,
                    REFERENCE_GENERATION ) IS NULL ) ),

      CONSTRAINT TABLES_SELF_REFERENCING_COLUMN_CHECK
        CHECK ( ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NULL
            OR ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NOT NULL ),

      CONSTRAINT TABLES_TEMPORARY_TABLE_CHECK
        CHECK ( ( TABLE_TYPE IN ( 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY' ) )
              = ( COMMIT_ACTION IS NOT NULL ) ),

      CONSTRAINT TABLES_CHECK_NOT_VIEW
        CHECK ( NOT EXISTS (
            SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
            FROM TABLES
            WHERE TABLE_TYPE = 'VIEW'
          EXCEPT
            SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
            FROM VIEWS ) )
      )
```

Description

1) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the fully qualified name of the table.

2) The values of TABLE_TYPE have the following meanings:

BASE TABLE	The table being described is a persistent base table.
VIEW	The table being described is a viewed table.
GLOBAL TEMPORARY	The table being described is a global temporary table.
LOCAL TEMPORARY	The table being described is a created local temporary table.
SYSTEM VERSIONED	The table being described is a system-versioned table.

3) The value of SELF_REFERENCING_COLUMN_NAME is the name of the self-referencing column of the table, if the table is a typed table. Otherwise, the value of SELF_REFERENCING_COLUMN_NAME is the null value.

4) The values of COMMIT_ACTION have the following meanings:

DELETE	A <table commit action> of DELETE was specified.
PRESERVE	A <table commit action> of PRESERVE was specified.
null	The table being described is not a temporary table.

5) The values of REFERENCE_GENERATION have the following meanings:

SYSTEM GENERATED	The values of the self-referencing column of the table are generated by the SQL-server.
USER GENERATED	The values of the self-referencing column of the table are generated by the user.
DERIVED	The values of the self-referencing column of the table are generated from columns of the table.
null	The table being described does not have a self-referencing column.

6) If the table being described is a table of a structured type *TY*, then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the fully qualified name of *TY*; otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

7) The values of IS_INSERTABLE_INTO have the following meanings:

YES	The table being described is insertable-into.
NO	The table being described is not insertable-into.

8) The values of IS_TYPED have the following meanings:

YES	The table being described is a typed table.
NO	The table being described is not a typed table.

Definition Schema   253

6.55   TRANSFORMS base table

Function

The TRANSFORMS base table has one row for each transform.

Definition

```
CREATE TABLE TRANSFORMS (
      USER_DEFINED_TYPE_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER,
      USER_DEFINED_TYPE_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
      USER_DEFINED_TYPE_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
      SPECIFIC_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
      GROUP_NAME                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TRANSFORM_TYPE                INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT TRANSFORM_TYPE_NOT_NULL
          NOT NULL
        CONSTRAINT TRANSFORM_TYPE_CHECK
          CHECK ( TRANSFORM_TYPE IN
                  ('TO SQL', 'FROM SQL') ),

      CONSTRAINT TRANSFORMS_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME, GROUP_NAME, TRANSFORM_TYPE ),

      CONSTRAINT TRANSFORMS_TYPES_FOREIGN_KEY
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME )
          REFERENCES USER_DEFINED_TYPES,

      CONSTRAINT TRANSFORMS_ROUTINES_FOREIGN_KEY
        FOREIGN KEY (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME)
          REFERENCES ROUTINES
   )
```

Description

1) The values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and
   USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier,
   respectively, of the user-defined type for which the transform being described applies.

2) The values of SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME are the catalog
   name, unqualified schema name, and qualified identifier, respectively, of the SQL-invoked routine that
   acts as the transform function for the transform being described. The value of GROUP_NAME is the
   identifier that acts as the name of a transform group.

3) The values of TRANSFORM_TYPE have the following meanings:

TO SQL	The transform being described identifies a to-sql function
FROM SQL	The transform being described identifies a from-sql function

6.56   TRANSLATIONS base table

Function

The TRANSLATIONS table has one row for each character transliteration descriptor.

Definition

```
CREATE TABLE TRANSLATIONS (
    TRANSLATION_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRANSLATION_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRANSLATION_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SOURCE_CHARACTER_SET_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_CATALOG_NOT_NULL
        NOT NULL,
    SOURCE_CHARACTER_SET_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_SCHEMA_NOT_NULL
        NOT NULL,
    SOURCE_CHARACTER_SET_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_NAME_NOT_NULL
        NOT NULL,
    TARGET_CHARACTER_SET_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_CATALOG_NOT_NULL
        NOT NULL,
    TARGET_CHARACTER_SET_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_SCHEMA_NOT_NULL
        NOT NULL,
    TARGET_CHARACTER_SET_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_NAME_NOT_NULL
        NOT NULL,
    TRANSLATION_SOURCE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_CATALOG_NOT_NULL
        NOT NULL,
    TRANSLATION_SOURCE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_SCHEMA_NOT_NULL
        NOT NULL,
    TRANSLATION_SOURCE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_NAME_NOT_NULL
        NOT NULL,

    CONSTRAINT TRANSLATIONS_PRIMARY_KEY
      PRIMARY KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME ),

    CONSTRAINT TRANSLATIONS_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA )
        REFERENCES SCHEMATA,

    CONSTRAINT TRANSLATIONS_FOREIGN_KEY_ROUTINES
      FOREIGN KEY ( TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
                    TRANSLATION_SOURCE_NAME )
        REFERENCES ROUTINES,

    CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_SOURCE
```

```
      CHECK ( SOURCE_CHARACTER_SET_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
            OR
              ( SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
                SOURCE_CHARACTER_SET_NAME ) IN
                ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                         CHARACTER_SET_NAME
                  FROM CHARACTER_SETS ) ),

  CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_TARGET
    CHECK ( TARGET_CHARACTER_SET_CATALOG NOT IN
            ( SELECT CATALOG_NAME
              FROM SCHEMATA )
          OR
            ( TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
              TARGET_CHARACTER_SET_NAME ) IN
              ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                       CHARACTER_SET_NAME
                FROM CHARACTER_SETS ) )
  )
```

Description

1) The values of TRANSLATION_CATALOG, TRANSLATION_SCHEMA, and TRANSLATION_NAME
   are the catalog name, unqualified schema name, and qualified identifier, respectively, of the transliteration
   being described.

2) The values of SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
   and SOURCE_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified
   identifier, respectively, of the character set specified as the source for the transliteration.

3) The values of TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
   and TARGET_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified
   identifier, respectively, of the character set specified as the target for the transliteration.

4) The values of TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA, and
   TRANSLATION_SOURCE_NAME are the catalog name, unqualified schema name, and qualified iden-
   tifier, respectively, of the specific name of the SQL-invoked routine used for the transliteration.

6.57   TRIGGERED_UPDATE_COLUMNS base table

Function

The TRIGGERED_UPDATE_COLUMNS base table has one row for each column identified by a <column name> in a <trigger column list> of a <trigger definition>.

Definition

```
CREATE TABLE TRIGGERED_UPDATE_COLUMNS (
      TRIGGER_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TRIGGER_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TRIGGER_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
      EVENT_OBJECT_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_CATALOG_NOT_NULL
          NOT NULL,
      EVENT_OBJECT_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_SCHEMA_NOT_NULL
          NOT NULL,
      EVENT_OBJECT_TABLE          INFORMATION_SCHEMA.SQL_IDENTIFIER
        CONSTRAINT EVENT_OBJECT_TABLE_NOT_NULL
          NOT NULL,
      EVENT_OBJECT_COLUMN         INFORMATION_SCHEMA.SQL_IDENTIFIER,

      CONSTRAINT TRIGGERED_UPDATE_COLUMNS_PRIMARY_KEY
        PRIMARY KEY
          ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME, EVENT_OBJECT_COLUMN ),

      CONSTRAINT TRIGGERED_UPDATE_COLUMNS_EVENT_MANIPULATION_CHECK
        CHECK ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME ) IN
                  ( SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME
                    FROM TRIGGERS
                    WHERE EVENT_MANIPULATION = 'UPDATE' ) ),

      CONSTRAINT TRIGGERED_UPDATE_COLUMNS_FOREIGN_KEY_COLUMNS
        FOREIGN KEY ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
                      EVENT_OBJECT_TABLE, EVENT_OBJECT_COLUMN )
          REFERENCES COLUMNS
  )
```

Description

1) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, schema name, and trigger name of the trigger being described.

2) The values of EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, and EVENT_OBJECT_TABLE are the catalog name, schema name, and table name of the table containing the column being described. The TRIGGERED_UPDATE_COLUMNS base table has one row for each column contained in an explicitly specified <trigger column list> of a trigger whose trigger event is UPDATE.

6.58 TRIGGER_COLUMN_USAGE base table

Function

The TRIGGER_COLUMN_USAGE base table has one row for each explicitly or implicitly identified column of a table referenced in the <trigger definition> of the trigger being described.

Definition

```
CREATE TABLE TRIGGER_COLUMN_USAGE (
     TRIGGER_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
     TRIGGER_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
     TRIGGER_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
     TABLE_CATALOG               INFORMATION_SCHEMA.SQL_IDENTIFIER,
     TABLE_SCHEMA                INFORMATION_SCHEMA.SQL_IDENTIFIER,
     TABLE_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
     COLUMN_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,

     CONSTRAINT TRIGGER_COLUMN_USAGE_PRIMARY_KEY
       PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

     CONSTRAINT TRIGGER_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
       CHECK ( TABLE_CATALOG NOT IN
             ( SELECT CATALOG_NAME
               FROM SCHEMATA )
             OR
             ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
             ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
               FROM COLUMNS) ),

     CONSTRAINT TRIGGER_COLUMN_USAGE_FOREIGN_KEY_TRIGGER_TABLE_USAGE
       FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
         REFERENCES TRIGGER_TABLE_USAGE
   )
```

Description

1) The TRIGGER_COLUMN_USAGE base table has one row for each column *COL* of a table *TAB* identified by a column reference or column name contained in the <trigger definition> of a trigger *TR* being described.

2) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TR*.

3) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*

4) The value of COLUMN_NAME is the name of the column *COL*

6.59 TRIGGER_PERIOD_USAGE base table

Function

The TRIGGER_PERIOD_USAGE base table has one row for each period identified in the <trigger definition>
of the trigger being described.

Definition

```
CREATE TABLE TRIGGER_PERIOD_USAGE (
      TRIGGER_CATALOG               INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TRIGGER_SCHEMA                INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TRIGGER_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_CATALOG                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_SCHEMA                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
      TABLE_NAME                    INFORMATION_SCHEMA.SQL_IDENTIFIER,
      PERIOD_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,

      CONSTRAINT TRIGGER_PERIOD_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

      CONSTRAINT TRIGGER_PERIOD_USAGE_CHECK_REFERENCES_PERIODS
        CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
              OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME
                FROM PERIODS) ),

      CONSTRAINT TRIGGER_PERIOD_USAGE_FOREIGN_KEY_TRIGGER_TABLE_USAGE
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
          REFERENCES TRIGGER_TABLE_USAGE
   )
```

Description

1) The values of TRIGGER_CATALOG, TRIGGER_SCHEMA, and TRIGGER_NAME are the catalog
    name, unqualified schema name, and qualified identifier, respectively, of the trigger *TR* being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME and PERIOD_NAME are the
    catalog name, unqualified schema name, and qualified identifier of the table name, and the period name,
    respectively, of the period that is identified in the <trigger definition> of *TR*.