INTERNATIONAL STANDARD

ISO/IEC 10728

First edition
1993-04-15

Information technology — Information Resource Dictionary System (IRDS) Services Interface

Technologies de l'information — Interface de services du gestionnaire de ressources du système d'informations (IRDS)

Reference number
ISO/IEC 10728:1993(E)

Table of Contents

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Annexes

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. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 10728 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology,* Sub-Committee SC 21, *Information retrieval, transfer and management for open systems interconnection (OSI).*

Annex A forms an integral part of this International Standard. Annex B is for information only.

Introduction

This International Standard is one of a series of International Standards on Information Resource Dictionary Systems. ISO/IEC 10027 defines the context within which this International Standard is to be applied.

Information technology – Information Resource Dictionary System (IRDS) Services Interface

1 Scope

The IRDS series of International Standards specifies a software tool that can be used to describe and potentially control an enterprise's information resources. It defines the structure and part of the content of the data to be maintained at the IRD Definition Level, and the structure of the data to be maintained at the IRD Level. It also defines the services to be provided for maintaining and retrieving data at both levels. Further details of the IRDS series of standards are to be found in ISO/IEC 10027.

This International Standard specifies a Services Interface that gives any program full access to all IRDS services, through whatever external call interface is provided by the language in which the program is written. The body of this International Standard defines the semantics of this interface, and also specifies the language bindings for ISO Pascal (ISO 7185). Language bindings for other ISO standard programming languages are provided as separate standards.

This International Standard makes no assumptions about an implementation environment, and assumes no specific run-time or compile-time interfaces.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 3166: 1988; *Codes for the representation of names of countries*

ISO 7185: 1990; *Information Technology - Programming languages - Pascal.*

ISO/IEC 9075: 1992; *Information Technology - Database Languages - SQL.*

ISO/IEC 10027: 1990; *Information Technology - Information Resource Dictionary System (IRDS) - Framework.*

ISO/IEC 10032: 1993; *Information Technology - Reference Model of Data Management*

3 Definitions and abbreviations

NOTE - SQL terms are not defined here. When used in this International Standard, they have the meanings ascribed to them in ISO/IEC 9075. All IRDS terms used in this International Standard are fully defined here or in ISO/IEC 10027.

3.1 Terms defined or referenced in the IRDS Framework (ISO/IEC 10027) and used in this International Standard

The following terms are defined (or referenced) and used in the IRDS Framework. They are used in the same way in this International Standard.

3.1.1 client

3.1.2 Information Resource Dictionary (IRD)

3.1.3 Information Resource Dictionary System (IRDS)

3.1.4 IRD definition

3.1.5 IRD definition level

3.1.6 IRD definition schema

3.1.7 IRD level

3.1.8 IRD schema

3.1.9 level pair

3.1.10 real system

3.1.11 service

3.2 Terms defined in this International Standard

Where each term listed in this clause is introduced in a later clause of this International Standard, it is printed in bold type.

3.2.1 active: A state in which a dictionary is accessible to all relevant IRDS services. When an IRD is not active, only the Reactivate IRD service is applicable.

3.2.2 archived: A content status class that indicates data that is no longer in active use.

3.2.3 attribute: A characteristic of an object.

3.2.4 common table: A table that exists in each IRD Definition and each IRD.

3.2.5 content module: A collection of objects introduced into an IRD Definition or IRD at the same time and from the source, identified by a module name that indicates the source of the module.

3.2.6 context: A working set established by default or by user request within which IRDS services are performed.

3.2.7 controlled: A content status class that indicates data that is stable and not subject to change.

3.2.8 definition object: An object recorded at the IRD definition level that controls the data which may be present at the IRD level.

3.2.9 dictionary: An IRD Definition or IRD

3.2.10 environment table: A table that exists once in each IRD Definition, controlling the services provided on that IRD Definition and any associated IRDs.

3.2.11 implementation-defined: Behaviour not defined by this International Standard, but which shall be precisely defined by any conforming implementation

3.2.12 implementation-dependent: Behaviour not defined by this International Standard, and which an implementation is not required to define. Further, there is no requirement that such behaviour be consistent from case to case.

3.2.13 internal table: A table that exists once in each IRD Definition and each IRD, rows in which cannot be accessed by the object-related services in clause 9.

3.2.14 IRD-specific table: A table that exists only in the IRD Definition or a specific IRD, as part of the representation of the data structuring rules of a defined data modelling facility.

3.2.15 IRD content status: A user-defined attribute of a working set. Every value of IRD content status belongs to one of the three predefined content status classes. Each object version takes its IRD content status from the working set that contains it.

3.2.16 IRD content status class: One of three predefined sets of IRD content statuses: uncontrolled, controlled and archived.

3.2.17 IRD object: An object recorded at the IRD level.

3.2.18 IRD Schema Group: A collection of one or more IRD Schemas that completely defines what may exist at any time in an IRD.

3.2.19 IRDS database: An IRD definition and zero or more IRDs.

3.2.20 IRDS environment: An operational instance of an implementation of the IRDS Services Interface managing an IRDS database.

3.2.21 IRDS name: A name optionally assigned when an object is added to an IRD or a definition object is added to an IRD definition. If specified, the combination of IRDS name, variation name, working set name and working set version name shall be unique.

3.2.22 IRDS session: A temporary association between an IRDS user and an IRDS environment, during which the former requests services and the latter performs them.

3.2.23 IRDS user: An individual or group authorized to use the IRDS.

3.2.24 level independent service: A service that is equally applicable to the IRD Definition level and the IRD level.

3.2.25 level specific service: A service which only applies either to the IRD Definition level or to the IRD level, but not both.

3.2.26 materialization (of a working set): That collection of object versions that can be in the working table of a cursor opened on that working set.

3.2.27 name: A string of characters used to distinguish objects, either alone or in combination with other names.

3.2.28 non-versionable (of an object type): Indicates that only one version of an object of that type may exist at any time, in a non-versionable working set; of a working set, indicates that the working set may not be based on another working set or be used as the basis for another working set.

3.2.29 object: Any concept or thing of interest to an enterprise.

3.2.30 object type: A class of objects all of whose attributes belong to a common set of attribute types.

3.2.31 object version: A record of the information about an object that is current for some period of time in some information processing context.

3.2.32 reference path: A directed association from one working set to another which allows an object version in the first working set to reference object versions in the second working set. A reference path allows references only in the direction in which it is specified.

3.2.33 referenced table: The table to which the reference is made in a referential constraint.

3.2.34 referencing table: The table from which the reference is made in a referential constraint.

3.2.35 subtable: Let SUBT and SUPERT be tables. SUBT is defined to be a subtable of SUPERT if and only if every row of SUBT corresponds to one and only one row of SUPERT, and each row of SUPERT corresponds to at most one row of SUBT. SUPERT is defined to be a supertable of SUBT.

3.2.36 supertable: A table that has at least one subtable (see definition of subtable).

3.2.37 uncontrolled: A content status class that indicates data that is not stable.

3.2.38 variation name: A name used to identify significantly different variants of objects with the same IRDS name.

3.2.39 versionable (of an object type): Indicates that more than one version of an object of that type may exist at the same time, in different working sets; of a working set, indicates that the working set may not contain objects of non-versionable object types, may be based on another working set and may be used as the basis for another working set.

3.2.40 working set: A collection of versions of either definition objects or IRD objects, defined by an IRDS user as a unit for the purposes of change management, content status specification and access control.

3.3 Data Item Name abbreviations

The following list describes all of the abbreviations that are used in naming the columns of the SQL tables and the corresponding Pascal constants, types and variables:

Add	= added
Arch	= archived
Atr	= attribute
Cls	= class
Cntl	= controlled
Col	= column
Cur	= cursor
Curr	= current
Dcs	= IRD content status
Def	= definition
Dflt	= default
Dflts	= defaults
Dic	= dictionary
Dom	= domain
Id	= identifier
Imp	= implementation
Ind	= indicator
Inst	= installation
Int	= integer
Ird	= Information Resource Dictionary
Len	= length
Lim	= limit
Maint	= maintained
Max	= maximum
Min	= minimum
Mod	= modified
Nat	= national
Num	= number
Obj	= object
Ref	= reference
Ret	= return
Scm	= schema
Sep	= separator
Sess	= session
Spec	= specification
Srv	= service
Str	= string
Tran	= transaction
Txt	= text
Ucntl	= uncontrolled
Val	= value
Var	= variation
Ver	= version
Wkg	= working
Ws	= working set

4 Conventions

This clause describes the conventions used within this International Standard to describe the facilities of an IRDS. These conventions are not themselves a subject of this standard. Several different specification conventions are used, for different purposes.

4.1 Specification of concepts and facilities

Within clause 5, diagrams are used, in conjunction with English description, to introduce some of the concepts and facilities that are formally defined later in this International Standard. Two types of diagram are used:

a) Data Structure Diagrams, described in 4.6;

b) Working Set Diagrams, described in 4.8.

4.2 Specification of data structures

In clause 6, the data structures to be managed by an IRDS Services Processor are specified using Database Language SQL (ISO/IEC 9075).

In general, this involves the use of tables to represent object types and columns to represent attribute types; certain data is also required for control purposes.

The use of SQL as a definition formalism in this International Standard does not imply any specific implementation approach. The user of the services provided at the IRDS Services Interface sees the data only in the terms defined in clause 8.

4.3 Specification of constraints - overview

Constraints on the possible values, or combinations of values, which may appear in the IRD are specified once only wherever possible. In order of preference, each constraint is specified:

a) within the formal data specification, in clause 6;

b) in the description of the relevant table, in clause 6;

c) in the description of the services that can be used to change the data, in clause 9.

The specification of constraints is described in detail in 4.7.

4.4 Specification of service data structures

In clause 8, the formats of the data structures associated with each service are described using Pascal (ISO 7185).

4.5 Specification of services

In clause 9, the services to be supported by an IRDS are specified using a combination of Pascal (ISO 7185) and English.

4.6 Data Structure Diagrams

The Data Structure Diagrams used in clause 5 illustrate tables and the constraints between the tables. A table is represented by a rectangular box.

Constraints between the tables are represented by lines between the boxes. Each line is considered to have two halves, each half associated with the box to which it is directly connected.

4.7 Specification of constraints - detail

4.7.1 Types of constraint

The following types of constraint are used within the formal data specification in clause 6:

a) Primary key constraints - which identify the primary key of each table.

b) Uniqueness constraints - which identify combinations of columns within a table whose values must be unique, when not wholly or partially null.

c) Referential constraints - which identify the dependencies of one table on another.

d) Check constraints - which allow the specification of many additional general constraints on the values or combinations of values which may appear in one or more rows in one or more tables.

In addition to their specification in clause 6, referential constraints are also illustrated diagrammatically in clause 5. The rest of this clause describes the diagramming conventions used for different types of referential constraint and the SQL syntax used to represent each type.

4.7.2 Overview of referential constraints

It is necessary to explain the type of referential constraints used before describing their representation.

A referential constraint exists between two tables A and B if at all times the values of some specified column or combination of columns, when not null, in each row in table B shall be equal to the values of a corresponding combination (in terms of number and data type) of columns in one and only one row in table A. Such constraints are further qualified as follows:

a) One-to-one - if only a single row in table B may reference any one row in table A.

b) One-to-many - if more than one row in table B can reference the same row in table A.

Orthogonal to the above classification of constraints, the following classification may also be made:

c) Optional at referencing table - if any of the columns in table B, which are used to reference table A, may be null; a row in which any of these columns is null is considered not to reference table A;

d) Optional at referenced table - if a row may exist in table A without being referenced from any row in table B;

e) Required at referencing table - if the referencing columns in table B may not be null. Every row in table B must reference a row in table A.

f) Required at referenced table - if a row in table A may exist only if there is a corresponding row in table B that references it.

The following basic constructs are used within the Data Structure Diagram to illustrate these constraints. Each half of the line that represents the constraint is treated separately in illustrating the characteristics of the constraint at each table.

a) Solid line - the constraint is required at that table;

b) Dashed line - the constraint is optional at that table;

c) "Crow's feet" at end of line - represents a one-to-many constraint. (Crow's feet are not allowed at both ends of a line.)

d) No "crow's feet" at either end - represents a one-to-one constraint.

The diagrammatic representations of some of the various constraint types listed above are now illustrated. In each case, an example is also given of how the relevant constraint would be expressed in clause 6, using SQL with or without accompanying English description.

4.7.3 Optional one-to-many referential constraint

Figure 1 illustrates the simplest (in terms of its SQL specification) constraint - an optional one-to-many referential constraint.

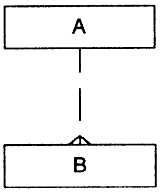


Figure 1 - Optional referential constraint

Table 1 shows the corresponding SQL syntax.

Table 1: SQL corresponding to Figure 1

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY )

CREATE TABLE B
( B1 IRDS_KEY PRIMARY KEY,

  B2 IRDS_KEY

  CONSTRAINT constraint-name
    REFERENCES A
    ON DELETE referential-action )
```

In table 1 and similar tables, IRDS_KEY represents the name of a domain (see ISO/IEC 9075) which is used throughout the tables defined in clause 6. It is used here solely to make these examples look as similar as possible to the SQL used in clause 6.

The SQL syntax used in clause 6 will normally contain an ON DELETE clause, similar to that shown in Table 1. This clause specifies additional information that is not shown in the diagram - namely the action to be taken when an attempt is made to delete a row in table A that is referenced by a row in table B.

For an optional referential constraint, as shown in table 1, the referential action can be one of the following:

a) RESTRICT - (cannot be specified explicitly, but is implied if ON DELETE clause is not specified) which means the deletion will not be allowed;

b) CASCADE - which means the referencing rows will also be deleted.

c) SET NULL - which means the referencing columns will be nullified, thus removing the reference, but the row itself will not be deleted.

4.7.4 Required uni-directional one-to-many referential constraint

Figure 2 illustrates a required uni-directional one-to-many referential constraint. The solid half of the line connected to box B illustrates that this constraint is required from B to A.

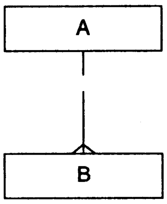


Figure 2 - Required referential constraint (one to many)

Table 2 shows an example of the SQL syntax to specify the constraint illustrated in figure 2. The addition of the NOT NULL clause on the referencing column B2 makes the constraint required from table B to table A.

Table 2 - SQL corresponding to figure 2

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY )

CREATE TABLE B
( B1 IRDS_KEY PRIMARY KEY,

  B2 IRDS_KEY NOT NULL

  CONSTRAINT constraint-name
   REFERENCES A
   ON DELETE referential-action )
```

Because of the NOT NULL clause, SET NULL is not a valid referential action for a required referential constraint.

4.7.5 Required uni-directional one-to-one referential constraint

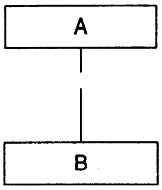


Figure 3 - Required referential constraint (one to one)

Figure 3 illustrates a required uni-directional one-to-one referential constraint. The removal of the crows feet from the line indicates in the diagram that this constraint is one-to-one.

Table 3 shows an example of the SQL syntax to specify the constraint illustrated in figure 3. The addition of a uniqueness constraint on column B2 enforces the singularity of the constraint.

Table 3 - SQL corresponding to figure 3

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY )

CREATE TABLE B
( B1 IRDS_KEY PRIMARY KEY,

  B2 IRDS_KEY NOT NULL
  CONSTRAINT constraint-name
   REFERENCES A
   ON DELETE referential-action,

  UNIQUE (B2) )
```

4.7.6 Self-referencing tables

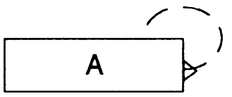


Figure 4 - Self-referencing constraint

Note that in any of the above cases A and B may be the same table, in which case the line representing the constraint is drawn as a circular arc between two points on the same box, as in figure 4, and the SQL referential constraint is between two columns, or groups of columns, in the same table, as in table 4.

Table 4: SQL corresponding to Figure 4

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY

  A2 IRDS_KEY

  CONSTRAINT constraint-name
   REFERENCES A
   ON DELETE referential-action )
```

4.7.7 Required bi-directional referential constraint

Figure 5 illustrates the diagramming convention for a required bi-directional one-to-many referential constraint.

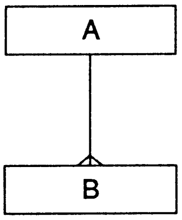


Figure 5 - Required bi-directional referential constraint

Table 5 - SQL corresponding to figure 5

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY

  CONSTRAINT constraint-name
    CHECK (A1 IN (SELECT B2 FROM B) )

CREATE TABLE B
( B1 IRDS_KEY PRIMARY KEY,

  B2 IRDS_KEY NOT NULL
    CONSTRAINT constraint-name
      REFERENCES A
      ON DELETE referential-action )
```

The SQL above is identical to that in table 2, except for the addition of a check clause, to make the constraint be required from the referenced table to the referencing table. Note that, since the constraints are mutual, rows cannot be inserted in A without setting the check constraint off until at least one corresponding row has been inserted in B.

A required bi-directional one-to-one referential constraint is represented in a similar fashion. The only difference would be the removal of the "crow's feet" in the figure, and the addition of a uniqueness constraint in the SQL for table B:

```
UNIQUE (B2) ,
```

4.7.8 Mutually-exclusive referential constraints

In some circumstances, two or more of the constraints described above may be mutually exclusive: one or other of the two constraints shall be satisfied, but not both. This is represented by an arc drawn across the relevant lines, as shown in figure 6. Here each row in table B shall at all times contain either a valid reference to a row in table A, or a valid reference to a row in table C. This example, designated Type 1, shows the mutual exclusivity constraint applying at the referring table.

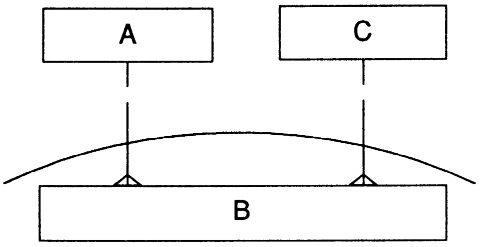


Figure 6 - Type 1 mutually-exclusive constraints

The following syntax may be used to enforce these constraints at the IRD Definition Level only. At the IRD Level, such constraints are identified declaratively in tables.

Table 6 - SQL corresponding to figure 6

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY )

CREATE TABLE C
( C1 IRDS_KEY PRIMARY KEY )

CREATE TABLE B
( BK IRDS_KEY PRIMARY KEY,

  B1 IRDS_KEY
    CONSTRAINT constraint-name
      REFERENCES A
      ON DELETE referential action,

  B2 IRDS_KEY
    CONSTRAINT constraint-name
      REFERENCES C
      ON DELETE referential action ,

  CONSTRAINT constraint-name
    CHECK
    (
      (B1 IS NULL AND B2 IS NOT NULL)
      OR
      (B2 IS NULL AND B1 IS NOT NULL)
    )
)
```

Figure 6 illustrates required uni-directional referential constraints. As discussed previously, other kinds of constraint exist. The syntax for required bi-directional constraints would require the same CHECK clause to be added to the referencing table. To support optional referential constraints, the CHECK statement shown should be replaced by the following:

```
CHECK (B1 IS NULL OR B2 IS NULL)
```

If one or both of the referential constraints is one-to-one, it will be necessary to add corresponding uniqueness constraints in table B:

```
UNIQUE (B1) ,
UNIQUE (B2) ,
```

To support more than two referential constraints, corresponding attributes B3, B4, etc. would be introduced, with the appropriate modifications to the CHECK clause.

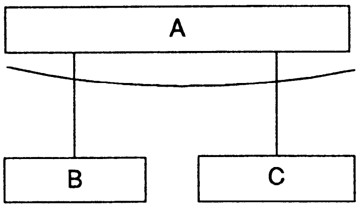


Figure 7 - Type 2 mutually-exclusive constraints

The reverse situation may also occur, as shown in figure 7. Here each row in table A may be referenced either by rows in table B or by rows in table C, but not both. In this case, designated Type 2, the mutual exclusivity occurs at the referenced table. The constraints illustrated in figure 7 are required bi-directional referential constraints.

As with Type 1 constraints, the following syntax may be used to implement these constraints at the IRD Definition Level.

Table 7 - SQL corresponding to Figure 7

```
CREATE TABLE A
( A1 IRDS_KEY PRIMARY KEY ,
 A2 CHAR NOT NULL
   CHECK (A2 IN ('B','C') ),
 UNIQUE (A1, A2) ,

CONSTRAINT constraint-name
 CHECK ( ( SELECT COUNT (*) FROM B
       WHERE A1=B1 )
    + ( SELECT COUNT (*) FROM C
       WHERE A1=C1 )
     = 1 )
)

CREATE TABLE B
( BK IRDS_KEY PRIMARY KEY ,
 B1 IRDS_KEY NOT NULL ,
 B2 CHAR NOT NULL
   CHECK (B2='B'),
 CONSTRAINT constraint-name
 FOREIGN KEY (B1, B2)
   REFERENCES A (A1, A2)
)

CREATE TABLE C
( CK IRDS_KEY PRIMARY KEY ,
 C1 IRDS_KEY NOT NULL ,
 C2 CHAR NOT NULL
   CHECK (C2='C'),
 CONSTRAINT constraint-name
 FOREIGN KEY (C1, C2)
   REFERENCES A (A1, A2)
)
```

Note that the format of the CHECK clause actually used to express the exclusivity of the constraints may differ from that shown here, depending on the circumstances.

Similar diagrams and syntax could be developed for required uni-directional constraints, or optional constraints, but neither is actually used in clause 6.

Combinations of Type 1 and Type 2 constraints are also possible, in which case the SQL will be a combination of the two approaches.

4.7.9 Subtables

Let SUPERT and SUBT be tables. SUBT is defined to be a subtable of SUPERT if and only if every row of SUBT corresponds to one and only one row of SUPERT, and each row of SUPERT corresponds to at most one row of SUBT. SUPERT is defined to be a supertable of SUBT.

A supertable may have more than one subtable. To clarify this case, let the rows of table A represent a set of objects and let the rows of tables B and C represent further sets of objects. If the sets of objects represented by the rows of tables B and C are both subsets of the set of objects represented by the rows of table A, then tables B and C are subtables of table A.

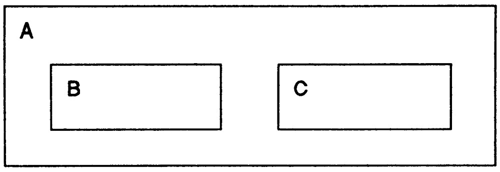


Figure 8 - Subtables

Figure 8 illustrates this, in the case where the sets of objects represented by tables B and C are disjoint. Note that use of this diagramming convention does not necessarily imply that every row in A corresponds to a row in table B or table C. If this additional requirement does hold, the situation is identical to that represented by figure 7, and the SQL representation will be identical to that given there; otherwise the SQL given in table 7 applies, but with the NOT NULL constraint removed from column A2 of table A, and the last line of the CHECK constraint changed to '<= 1'.

4.7.10 Principles for expressing constraints

The following general principles have been applied when specifying constraints in this International Standard:

a) CHECK and referential constraints are named;

b) When specifying referential constraints between tables, the following rules are applied:

i) When a 1:M relationship exists between two tables, the referential constraint is specified from the many to the one;

ii) When a 1:1 relationship is optional at one table, but mandatory at the other, the table at which the relationship is mandatory is considered dependent on the other (independent) table, and the constraint is specified from the dependent table to the independent table;

iii) If a 1:M relationship is mandatory at both tables, in addition to the referential constraint from the many to the one, a CHECK constraint is specified in the reverse direction.

iv) If a 1:1 relationship is mandatory at both tables, a CHECK constraint is used in one direction, not only for consistency with the 1:M case, but also because a second referential constraint would require additional columns for the foreign key. This is because the manner in which tables have been specified as distinct subtypes of IRD_OBJECT_VERSION does not allow primary key values in different tables to match.

NOTE - Such 1:1 relationships in the IRDS occur only in conjunction with other relationships with which they are mutually exclusive. If this were not the case, there would be no need for separate tables.

When deciding in which direction to specify the referential constraint, the function of the two tables in question is considered, also the impact of referential actions. Where possible, the constraint is specified so that all mutually exclusive constraints act in the same direction with relation to the table at which the exclusivity exists.

c) There are often several ways to express a logical constraint as a CHECK constraint, perhaps involving different columns and/or tables. When deciding how to specify a constraint, the following is the order of preference:

i) A constraint with no sub-queries is preferred to a constraint with a sub-query;

ii) When choosing between two constraints with sub-queries, the more concise expression is used;

d) Consistent techniques are used wherever possible to express particular types of constraint:

e) Where the approach dictated by the above principles conflicts with that taken in the SQL Definition Schema (ISO 9075), the SQL approach is taken wherever possible.

4.8 Working Set Diagrams

The Working Set Diagrams used in clause 5 illustrate working sets and the two sorts of path from one working set to another. A working set is represented by an ellipse, and a path from one working set to another by a line. The arrow along a line indicates the direction of the reference, and the label on or near the line indicates the type of reference.

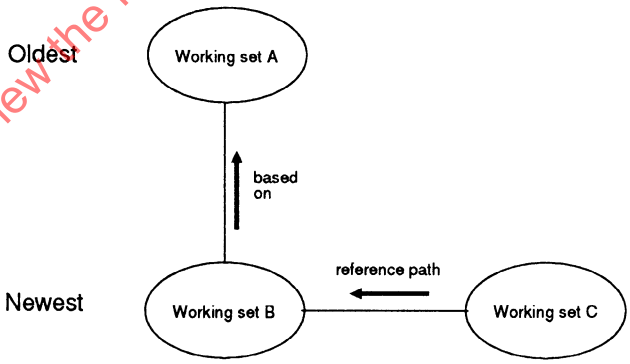


Figure 9 - Working set diagram conventions

In figure 9, working set B was created based on working set A. There is a reference path from working set C to working set B.

A reference path allows references only in the direction of the arrow. In figure 9, object versions in Working Set C can refer to object versions in Working Set B, but object versions in Working Set B can NOT refer to object versions in Working Set C.

5 IRDS concepts and facilities

This International Standard supports a set of facilities several of which are outlined in ISO/IEC 10032 and in ISO/IEC 10027. The facilities are categorized as follows:

a) data modelling facilities;

b) version control facilities;

c) naming facilities;

d) facilities for establishing limits and defaults;

e) other added value facilities.

These facilities are supported by data which is recorded in the tables for the IRD Definition Level (see ISO/IEC 10027).

The term object is used in this International Standard to refer to that which is represented by a row in a table or tables. Every object is either a definition object or an IRD object, depending on whether it is represented by rows in tables at the definition level or IRD level.

Services are provided by an IRDS Services Interface Processor (see ISO/IEC 10027) to retrieve and maintain the data in these IRD Definition tables. The IRD Definition tables are defined in clause 6. The concepts for the services are defined in clause 7, the service data structures in clause 8 and the services themselves in clause 9.

It is important to distinguish between the defining mechanism and that which is defined. The IRD Definition tables are defined in clause 6 of this International Standard using SQL Schema Definition statements (see ISO/IEC 9075) as the defining mechanism.

The defined mechanism in the IRD Definition tables is used in turn for the IRD Level pair as a defining mechanism. In this International Standard, the defined mechanism is the data modelling facility supported by the SQL Schema Definition statements. A data modelling facility is a set of rules for structuring a collection of persistent data and an associated set of data manipulation rules (see ISO/IEC 10032).

5.1 IRDS Environment concepts

Multiple IRD Definitions may exist in one real system; each IRD Definition is independent of any other IRD Definitions. Each IRD Definition may contain the definitions of zero, one or more IRD Schema Groups. For each IRD Schema Group, there exists zero or one IRD. Hence multiple IRDs may exist in one real system. Each IRD is structured according to the data modelling facility defined in a single IRD Schema Group.

One IRD Definition and its associated IRD Schema Groups and IRDs are part of an IRDS environment. An IRDS Services Interface Processor (see ISO/IEC 10027) may interact with any number of IRD Definitions. However, an IRDS user may interact with only one IRD Definition and its associated IRD Schema Groups and IRDs in one IRDS session.

An IRDS Services Interface Processor in an IRDS environment may use the services of a Database Services Processor. Each Database Services Processor conforms to a specific data modelling facility.

5.2 Categories of table

One of the aims of this International Standard is to have as much parallelism between the two level pairs as possible. For each level pair, the content of the tables is categorized as follows:

a) Internal tables: those associated with object management and version control, that exist once in each IRD Definition and IRD;

b) Environment tables: those that exist once in each IRD Definition, controlling the services provided on that IRD Definition and any associated IRDs;

c) IRD-specific tables: those that exist only in the IRD Definition or a specific IRD, including those that embody the data structuring rules of a defined data modelling facility (see ISO 10032); for the IRD Definition this data modelling facility is that of SQL (ISO/IEC 9075).

d) Common tables: those that exist once in each IRD Definition and IRD, associated with naming, version control and other common facilities;

Table 8 shows in tabular form some relevant characteristics of these categories.

Table 8 - Characteristics of table categories

Category	Row represents object	Versionable	Shared across IRD's
Internal	No	No	No
Environment	Yes	No	Yes
IRD-specific	Yes	Optional	No
Common	Yes	No	No

In this International Standard tables are sometimes referred to using terms other than the categories described above: for example, Referencing and Referenced tables (see 4.7.2,), and Supertable and Subtable (see 4.7.9). Such concepts are independent of the categorization above.

5.3 Overview of IRD Definition tables

The data on the IRD Definition Level defines and controls the data on the IRD Level.

The IRD Definition consists of 28 IRD Definition tables. These tables are listed below.

Internal tables:

1	IRD_OBJECT
2	IRD_WORKING_SET
3	IRD_OBJECT_VERSION
4	IRD_REFERENCE_PATH

Environment tables:

5	IRDS_USER
6	IMP_LIMITS
7	IRDS_DICTIONARY

IRD-specific tables:

8	IRD_SCHEMA_GROUP
9	IRD_SCHEMA
10	IRD_SCHEMA_REFERENCE
11	IRD_DATA_TYPE_DESCRIPTOR
12	IRD_DOMAIN
13	IRD_TABLE
14	IRD_VIEW
15	IRD_COLUMN
16	IRD_VIEW_TABLE_USAGE
17	IRD_VIEW_COLUMN_USAGE
18	IRD_TABLE_CONSTRAINT
19	IRD_KEY_COLUMN_USAGE
20	IRD_REF_CONSTRAINT
21	IRD_CHECK_CONSTRAINT
22	IRD_CHECK_TABLE_USAGE
23	IRD_CHECK_COLUMN_USAGE
24	IRD_ASSERTION

Common tables:

25	IRD_MODULE
26	IRD_CONTENT_STATUS
27	INSTALLATION_DEFAULT
28	IRD_WORKING_SET_PRIVILEGE

It is important to distinguish between the IRD Definition table called 'IRD_TABLE' and the members of the set of IRD tables. Each member of the set is referred to as an IRD table. The data about each IRD table is recorded in a row in the IRD Definition table called 'IRD_TABLE'.

The sequence of the IRD Definition tables in this list is based on the categories defined in 5.2 above, and then on the referential constraints between pairs of tables. In general, if table B references table A then table A precedes table B in the sequence. The sequence in the above list is used in clause 6.

Within the SQL text in clause 6, and in direct references to it elsewhere, names of tables and columns are in upper case, with consecutive words connected by an underscore character. In informal references to tables and columns, as in the text and figures in this clause, each word in the name is in lower case with an initial capital, and words are separated by spaces.

Table 3 called IRD Object Version contains a row for each row in the other IRD Definition tables, and contains audit columns that would otherwise be repeated in several IRD Definition tables. Table 1 called IRD Object contains a row for each set of rows in table 3 which represent versions of the same object; it contains audit columns and other columns that support naming and added value facilities. Table 2 contains a row for each working set defined within the IRD Definition; each row in table 3 contains a reference to a row in table 2. Table 4 contains the definitions supporting references between working sets.

Tables 5, 6 and 7 are the Environment tables. There is a row in table 5 for each IRDS User having any privileges with respect to the associated IRD Definition, and a single row in table 6 specifying any implementor-defined limits. There is a row in table 7 for the IRD Definition itself, and for each IRD created within that IRD Definition.

Tables 8 to 24 in this list are the IRD-specific tables, and are based on part of the SQL Definition Schema. The name of each table is modified to the singular form and includes the prefix 'IRD'. The sequence of presentation is the same as in the SQL standard. However, some of these tables have extra columns in addition to those in the SQL standard, to support IRDS facilities such as naming and added value facilities.

Tables 25 to 27 are included to support defaults and other added value facilities. Some of these tables contain references to the data modelling facility tables (8 to 24).

Table 28 contains the definitions supporting access control.

The structure diagrams shown on the following pages (figures 10, 11, 12 and 13) show how the tables on the IRD Definition Level are inter-related, using the conventions defined in clause 4.

Figure 10 shows the basic structure at this level; figure 11 shows how data type descriptors are related to columns and domains, and figure 12 shows how viewed tables are defined in terms of tables and their columns. Figure 13 shows the relationships between the tables introduced for version control.

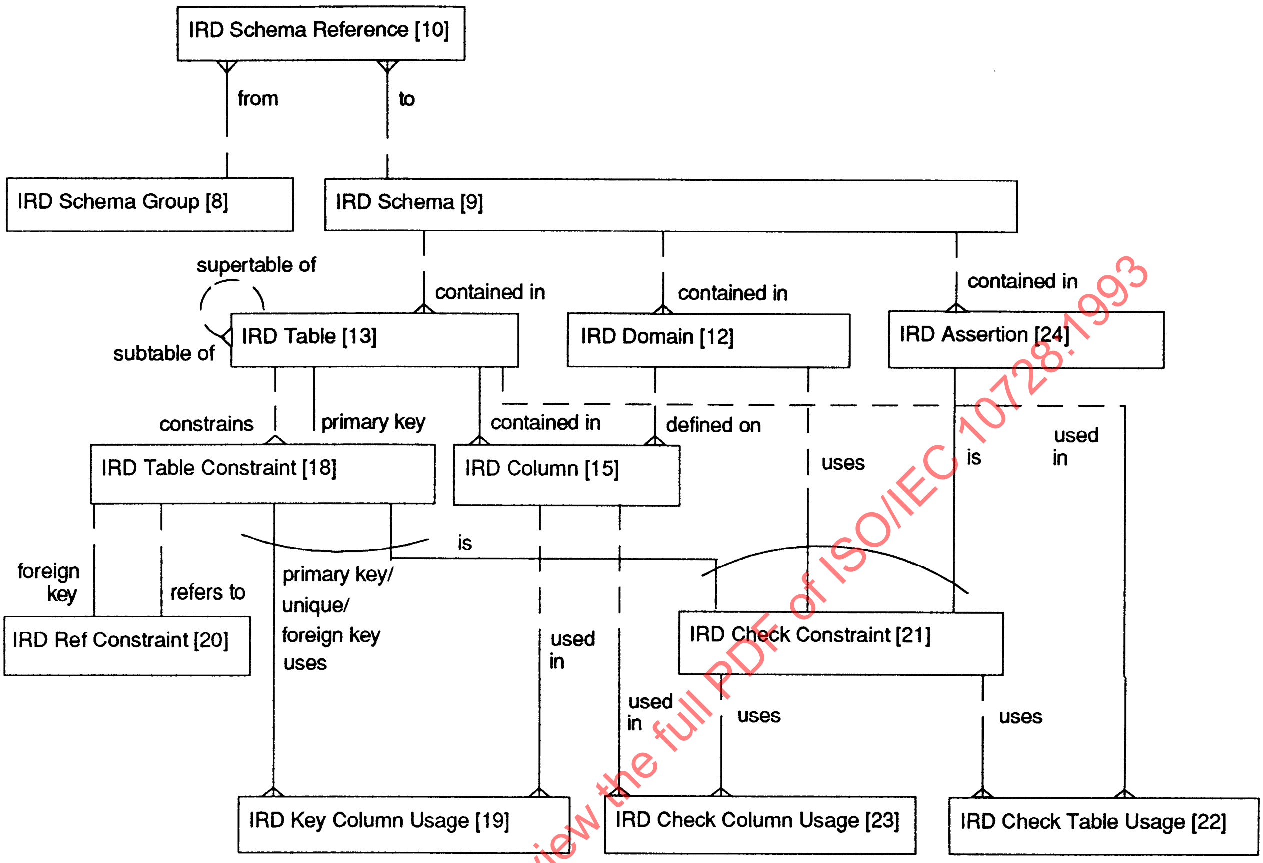


Figure 10 - Structure diagram for the IRD Definition Level

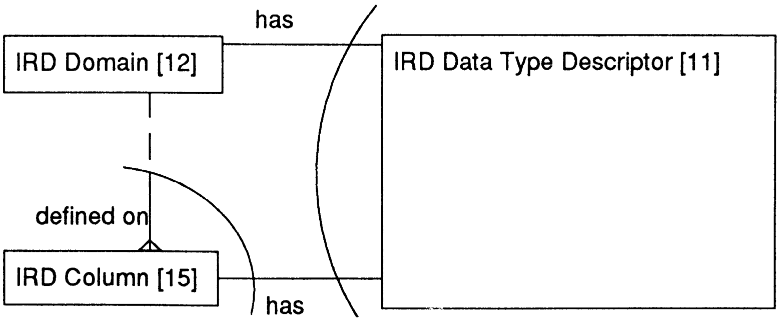


Figure 11 - Use of Data Type Descriptors

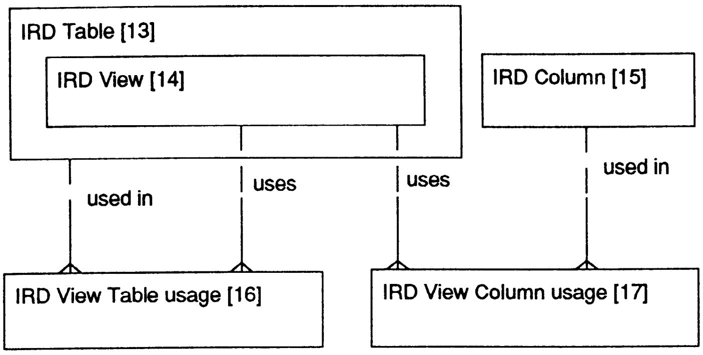


Figure 12 - IRD View usages

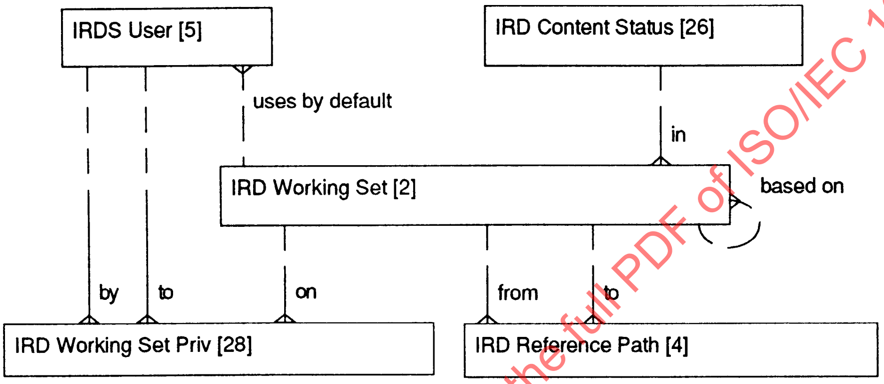


Figure 13 - IRD Definition Level Version Control tables

Tables 1 and 3 do not appear in any figure because each of the other tables is a subtable of table 3. Table 25 does not appear in any figure because a Module can be referenced by any definition object type. Tables 6, 7 and 27 do not appear in any figure because they are used for control purposes only, and are not directly related to other tables.

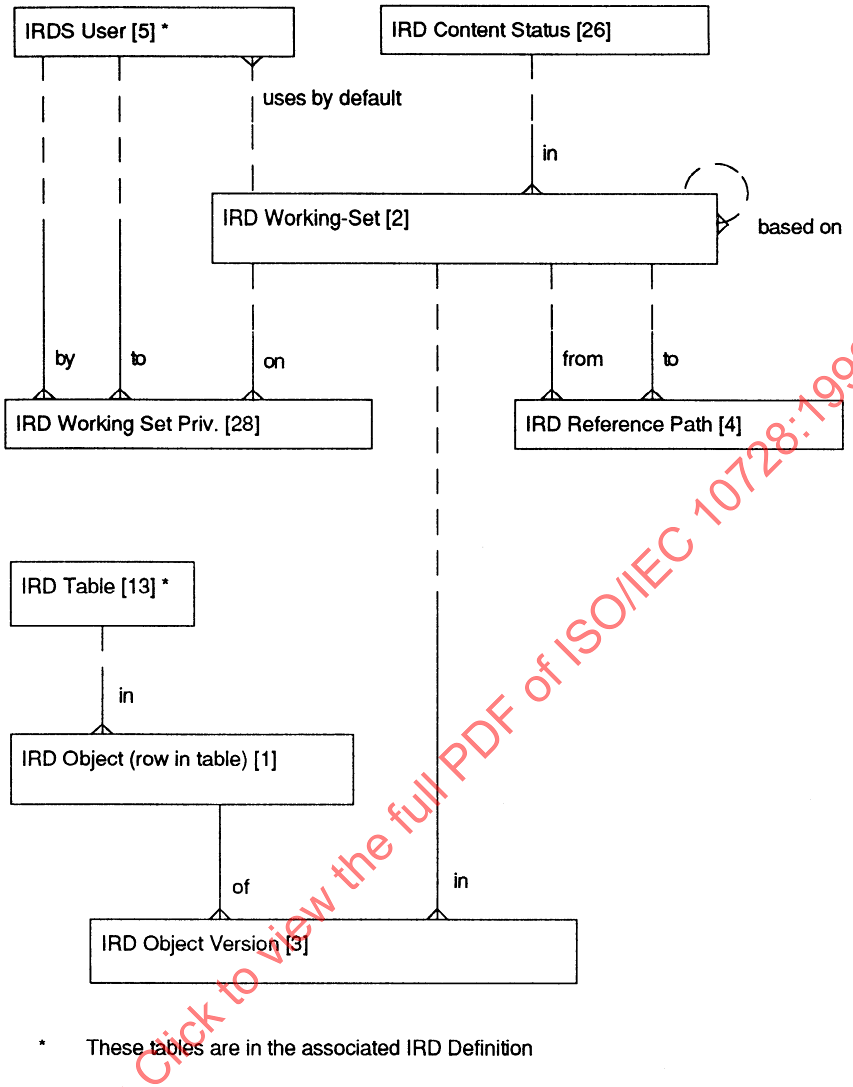


Figure 14 - Tables on the IRD Level

5.4 Overview of IRD tables

5.4.1 Overview

Each IRD may contain several IRD-specific tables. These tables may be defined in another standard, by an implementor or by a user. In order to be able to define the IRD level services in this International Standard, a number of internal and common tables are introduced. Two of these are named IRD Object and IRD Object Version, or, within the SQL statements in clause 6, IRD_OBJECT and IRD_OBJECT_VERSION.

The following holds for each IRD:

a) Each IRD is described by an IRD Schema Group. Through rows in the IRD Schema Reference table, that IRD Schema Group references one or more IRD Schemas that provide the detailed definition of the IRD.

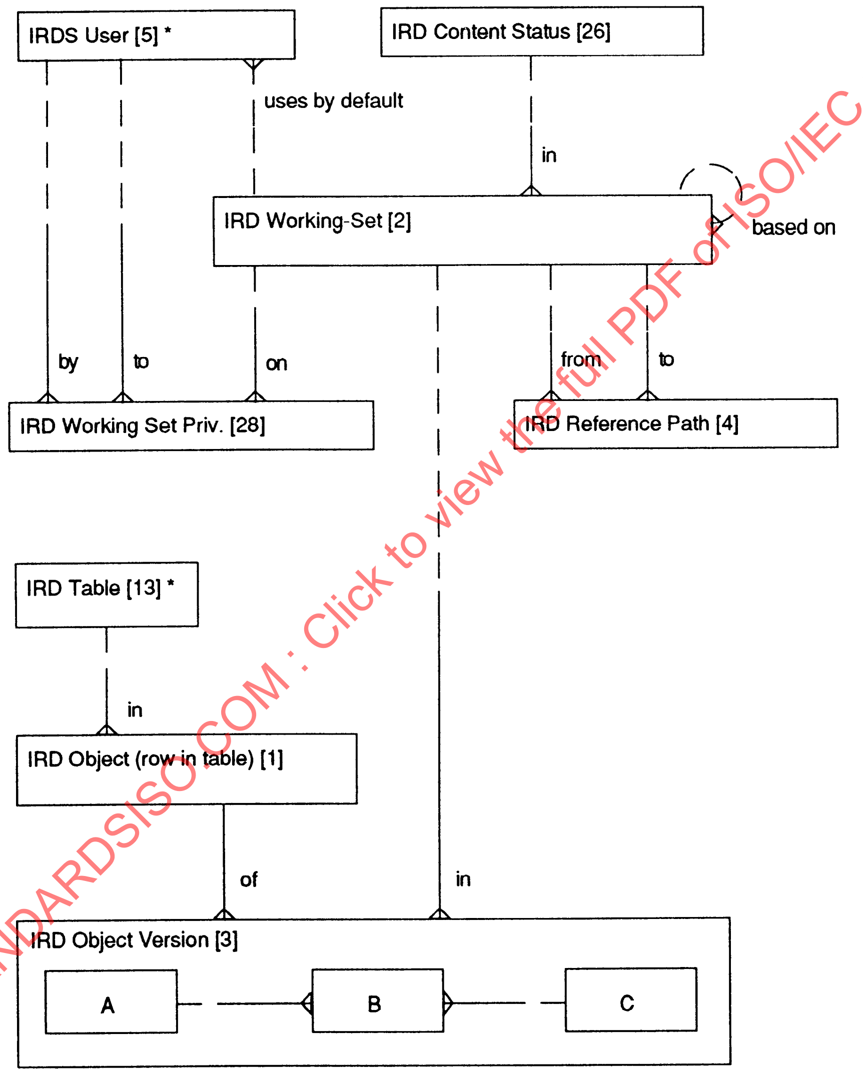
Each such IRD Schema may include the definitions of one or more IRD Tables, through the inclusion of rows in the IRD Definition table called IRD Table which reference that IRD Schema. For each table so defined, the IRD contains an IRD table conforming to that definition.

b) For each row in each such IRD table, there is a corresponding row in the IRD Object Version table, in which audit attributes and version control information are recorded;

c) For each row or rows in the IRD Object Version table which represent versions of the same object, there is a corresponding row in the IRD Object table.

IRD_TABLE				
IRD_ KEY	IRD_ TABLE_ NAME	IRD_ SCHEMA_ KEY	ETC.	
~	A	~		
~	B	~		
~	C	~		

Figure 15 - Example rows in the IRD_TABLE table



* These tables are in the associated IRD Definition

Figure 16 - Example of tables at the IRD Level

5.4.2 Internal and common tables

Figure 14 shows how the tables used in the definition of the IRD level are related to one another and to some of the tables defined for the IRD Definition Level (tables 5 and 13), using the conventions defined in 4.7. The structure represented in figure 14 and defined in clause 6 is defined solely for the purpose of specifying rules and defining the IRDS services for the IRD level pair in clause 9. There is no requirement that any implementation actually uses this tabular structure.

5.4.3 IRD-specific tables

Each row in the IRD_TABLE table in the IRD Definition (see 6.1.4.13) which is part of an IRD Schema Group defines a table at the IRD level in any associated IRD. Each such table is a subtable of the IRD object version table at the IRD level. In SQL terms this means that the tables defined by the rows in the IRD_TABLE table must contain columns referencing the columns which form the key of the IRD_OBJECT_VERSION table.

If the IRD Definition contained, for example, the rows shown in figure 15, then the tables that would appear at the IRD level are those represented in figure 16 as the subtables A, B and C. For the sake of illustration, two referential constraints are assumed to be defined in table B, one to table A and the other to table C. A more complex example is provided, for tutorial purposes, in annex B.

The columns of the IRD-specific tables at the IRD level are defined by the rows in the IRD_COLUMN table at the IRD Definition level. Similarly the referential integrity and other table constraints are defined by the IRD_TABLE_CONSTRAINT table. The check constraints are defined by the IRD_CHECK_CONSTRAINT table. The Domains and Assertions are defined by the IRD_DOMAIN and IRD_ASSERTION tables respectively.

5.5 Data and the objects to which the data refers

It is important in the context of an IRDS to distinguish clearly between the data about an information systems concept and the manifestation of that concept.

For example, it is important to distinguish between a table in the SQL sense and data about that same table. The data about the table, for example who added its definition to a schema and when its definition was last modified, should not be confused with data contained in the table. In an IRDS context the data contained in a table at one level may at the same time be data about some other table at a lower level.

5.5.1 Definition objects comprising data modelling facility

In terms specific to the data levels used in this standard, an IRD Definition contains, among other things, data about IRD Schema Groups and the IRD schemas, tables, columns and constraints associated with each. Any object associated with an IRD Schema is said to be a component of that IRD Schema. Each type of component is supported at the IRD Definition Level by a table. These tables are used to represent the SQL data modelling facility.

Each IRD Schema comprises one row in the IRD Definition table called IRD Schema and those rows in the other tables shown in figure 10 which reference (directly or indirectly) this row.

Each IRD Schema Group comprises one row in the IRD Definition table called IRD Schema Group, one or more rows in the IRD Definition table called IRD Schema Reference, and the IRD Schemas referenced by those rows.

This International Standard places no restriction on the number of IRDs that may refer to a single IRD Schema Group.

In addition to the data modelling facilities provided by the SQL standard, the IRDS Services Interface supports the definition of supertables and subtables as defined in 4.7.9.

Let SUPERT and SUBT be tables. If SUBT is specified to be a subtable of SUPERT then for each row in SUBT there is always a corresponding row in SUPERT, with the same primary key. Thus SUBT effectively inherits all the columns and constraints of SUPERT. If a row is inserted into SUBT, a corresponding row is inserted into SUPERT by the IRDS Services Interface processor. If a row of SUBT is modified such that any of the inherited columns are modified then those columns of the row in SUPERT are modified. If a row of SUBT is deleted then the corresponding row of SUPERT is deleted. When a row of SUBT is inserted, deleted or modified, all the integrity constraints specified for both SUBT and SUPERT are enforced.

An inheritance hierarchy may be multi-levelled, i.e. SUPERT may itself be a subtable of some other table. In this case the above process is applied recursively.

This version of this International Standard defines facilities for a table to inherit directly from only a single supertable.

5.5.2 Definition objects dependent on an IRD Schema Group

Before any data may be placed in IRD tables, an IRD must be created (using the Create IRD service). An IRD Schema Group may be referenced by a Create IRD service only after it has been validated (using the Validate IRD Schema Group service) to verify that it satisfies its internal constraints. Any schema group that has not been validated, or of which any components has been modified since validation, is said to be unvalidated.

5.5.3 Content of IRD tables

A subset of the content of the IRD tables may comprise one or more Application Schemas, but the IRD tables may also contain data that is not, and may never be, part of an Application Schema.

For example, an IRD table called Application Table may contain rows indicating the intention to create tables for data about EMPLOYEEs and SUPPLIERs. Such rows may be inserted before any application data concerning employees or suppliers is captured or stored. Similarly, rows may be included in a table entitled Application Column indicating the intended content of these application tables.

The concepts of validation and activation may apply to application schemas but the meaning of these concepts in relation to application schemas is not defined by this International Standard.

5.5.4 Accessibility of tables to users

Most of the IRD and IRD Definition tables are accessible to IRDS users for retrieval and update. This access is provided by the services described in clauses 7, 8 and 9.

The following tables are provided to support the internal working of an IRDS Services Interface Processor:

IRD_OBJECT
IRD_OBJECT_VERSION
IRD_WORKING_SET

The IRD_WORKING_SET table is accessible to users only through the services provided for creating, modifying and deleting working sets (Create Working Set, Drop Working Set, Modify Content Status). The other two tables are maintained and used by the IRDS Services Interface Processor, and cannot be directly accessed by a user.

5.6 Version Control concepts

This International Standard provides for identical version control facilities at the IRD and IRD Definition levels. Though the examples given are usually at the IRD Level, this clause is generally applicable to both levels.

Some IRDS users may not wish to be conscious of the version control facilities specified in this International Standard. Such users may simply define a single working set and make that their default working set, thus using it as the context for all services.

5.6.1 Objects and Versions of Objects

Objects of certain object types may exist within an IRD in several versions. What are loosely referred to as the attributes of an object are actually, with minor exceptions, the attributes of an object version. Two exceptions are the IRDS name and variation name of the object version, which are common to all versions of the object, and if changed are applied to all object versions.

A non-versionable object type is one of which only a single version may exist for any object. Such objects may be created only in a non-versionable working set. Objects of versionable object types may exist either in versionable or non-versionable working sets.

Object types represented by rows in the environment and common tables (see above) are all non-versionable, and exist in the Environment and Common working sets respectively. Object types represented by rows in IRD-specific tables are usually likely to be versionable; all such tables defined for the IRD Definition level in 6.1 are versionable.

A user may choose to create a non-versionable working set. There is no way to change a non-versionable working set into a versionable working set, or vice versa, once it has been created.

Rows in the IRD Object, IRD Working set and IRD Reference Path tables do not represent object versions, and therefore are not considered to be in any working set.

5.6.2 Working Sets

On the IRD Definition level it is possible to identify one or more working sets, each of which is associated with the IRD Definition Schema.

At any one time an IRDS user may be developing a set of IRD objects that describes a concept such as a program, a screen layout, a database schema, or a business model. Such a set of objects needs to be thought about and managed as a whole.

For example, an IRDS user will rarely think about the individual lines of code in a program and their references to a database schema. Yet each of these is likely to be a specific version of a specific IRD object. Version control is applied to sets of object versions that the user wishes to handle as a unit for the purposes of change management; this is called an IRD working set.

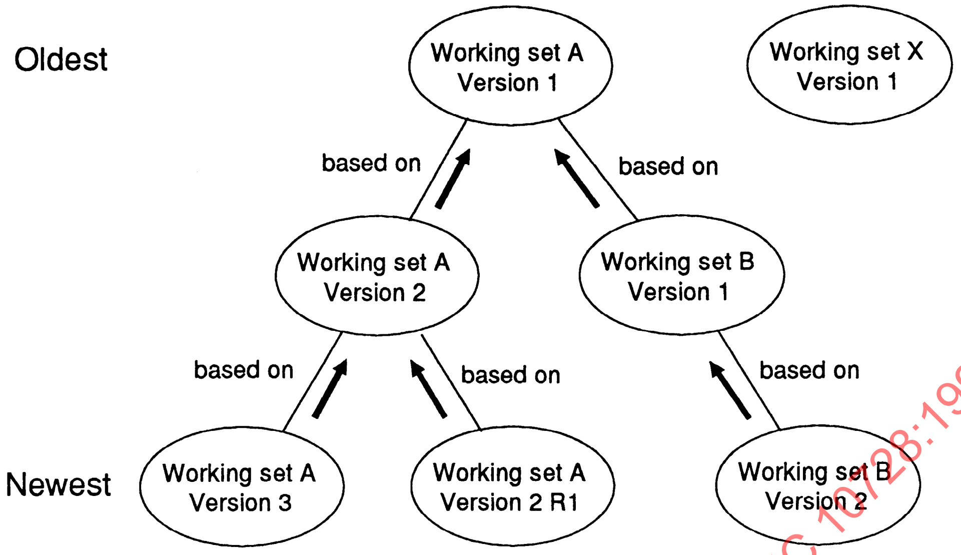


Figure 17 - Basing one working set on another

For each IRD, each of its IRD working sets is identified by an IRD working set name and an IRD working set version name. IRD working sets that are related for change management or project management purposes may share the same IRD working set name. However, this International Standard imposes no structure on the working set names or the working set version names, nor restrictions on their use. The IRD services that relate to version control operate upon the IRD working sets.

As on the IRD Definition level, there is a set of tables on the IRD level for each IRD, which support the version control mechanism. These tables are as follows:

IRD_WORKING_SET
IRD_WORKING_SET_PRIVILEGE
IRD_REFERENCE_PATH

An object version in the IRD is identified by the combination of a working set key and an object key. This implies that an IRD working set contains at most one version of any object.

5.6.3 Working sets and users

An IRD working set is created by the invocation of the Create Working Set service described in clause 9. Its contents are initially empty. Objects are subsequently added to an IRD working set by invocation of the Add Object service, with a specified or default IRD working set as the current working set.

5.6.4 Basing one working set on another

When a new working set is created, it can be based upon another working set. The result of this is to make the new working set effectively a copy of the working set being used as a basis. A non-versionable working set (see 5.6.1) may neither be based on another working set, nor be the basis for any other working set.

Figure 17 illustrates the way in which one working set may be used as the basis for another.

More than one IRD working set may be based on a single earlier IRD working set. Thus it is possible to define both stand-alone IRD working sets and trees of IRD working sets. The root of the tree is the oldest IRD working set. This is illustrated in figure 17. Note that figure 17 is an instance diagram, not a type diagram.

The "version path" of a working set WS_n is a sequence of working sets (WS_n, ..., WS_1) where each working set WS_j ($n>=j>1$) is based on WS_{j-1}.

Whenever a reference is made to an object version in the current version path, the working set identifier of the referenced object version is taken to be that of the working set in the current version path containing the latest version of the referenced object.

In figure 17 instances of working sets are shown by ellipses. Instances of the 'is the basis for' constraint are shown by lines.

In this example, there is a working set with working set name 'X' and working set version name '1'. This working set is not the basis of any other working sets, nor is it based on any other working set.

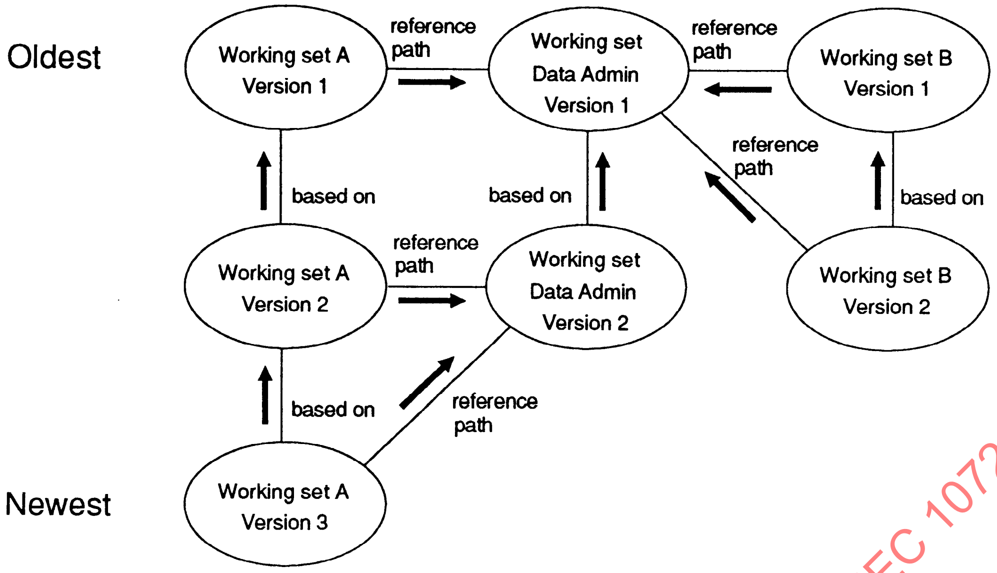


Figure 18 - Example references between working sets

There are four working sets with the working set name of 'A'. They have four different working set version names: '1', '2', '3' and '2 R1' respectively. Working set 'A' version '1' is not based on any working set but is itself the basis for working set 'A' version '2' and working set 'B' version '1'.

Working set 'A' version '2 R1' might be a correction working set, based on working set 'A' version '2'.

Working set 'B' version '1' is based on working set 'A' version '1', and is itself the basis for working set 'B' version '2'.

5.6.5 Materialization of a working set

The materialization of a working set is a set of object versions, defined recursively as follows:

Let WSA and WSB be working sets:

a) If WSA is not based on any working set, then the materialization of WSA is the set of all object versions included in WSA;

b) If WSA is based on WSB, then the materialization of WSA is the set of all object versions in WSA plus all object versions in the materialization of WSB except:

   1) versions of objects of which a version is in WSA;

   2) versions of objects that have been deleted in WSA.

The set of object versions in WSA is said to be superimposed on the materialization of WSB. An object version is visible in WSA if and only if that object version is in the materialization of WSA.

Referential constraints, such as those illustrated in figure 10, are to be interpreted AFTER materialization.

A working set WSA that is based upon another working set WSB can subsequently be dropped from the IRD, provided that it is not itself the basis for any further working set. The effect of dropping WSA is to delete from the IRD all object versions that were either created or modified within the context of WSA. An IRDS user can make use of this operation to start again after encountering errors in an uncontrolled working set.

A working set WSB that is the basis for another working set WSA, but is not itself based upon any working set, may also be dropped. The effect of doing so is to create versions in WSA of any objects in WSB that remain in the materialization of WSA, and to set the basis of WSA to 'none', represented by null. An IRDS user can make use of this operation to remove an archived working set from the IRD.

5.6.6 References from one working set to another

Sometimes an organization may have several teams or subteams working independently. The hierarchical inheritance mechanism described in clause 5.6.4 works well for a single team. In the example shown in figure 18, team A might own all the working sets with the working set name 'A'.

However, it often happens that a data administration team or a team developing common shared software needs to provide definitions to several teams all working at different speeds. In real life it is not possible to guarantee that teams will always deliver a new system on time. Thus a degree of independence and separate working must be allowed, together with the sharing of definitions where necessary.

Therefore this International Standard prescribes facilities for reference paths to be defined which permit object versions in one working set to reference object versions in one or more other working sets. In other words, while an object version may exist in only one working set, it may, subject to the existence of reference paths, be referenced from object versions in any number of other working sets, not necessarily having the same working set name or being based on the working set containing it. No reference path shall exist from a working set A to any working set which is within the version path of A.

In figure 18 the data administration team, owners of the working sets whose working set name is 'Data Admin', might provide data definitions to teams A and B.

The effect of a reference path from a working set A version 1 to working set Data Admin version 1 is to allow any user whose context is working set A version 1 (subject to the necessary privileges) to create references from object versions created or modified in that context to object versions in working set Data Admin version 1. The existence of such a reference path does not mean that all object versions in the Data Admin version 1 working set are visible in working set A version 1.

This allows project team A, working with working set A version 1, to see and reference, but not alter, the shared definitions. Note that any kind of dictionary content could be shared in this way.

Similar reference paths are shown for later working sets of project A and for the working sets of project B. When Working Set A Version 2 was created, it would initially have inherited the reference paths from Working Set A Version 1. The Modify Reference Path service would then have been used to redirect the reference path from Data Admin Version 1 to Data Admin Version 2, and to adjust any references using the refernce path. The references now refer to the object versions in the materialization of Data Admin Version 2.

Note that when project A moved on to a new working set referencing a new version of the Data Admin definitions, project B did not have to, even though they themselves chose to go to a new working set for their own definitions.

Modifications, deletions and insertions can only be made to object versions in the materialization of the working set that is the IRDS User's current context. A user wishing to update object versions in a working set accessed through a reference path must have update privileges for that working set and must make that working set their current context (see 5.6.8).

Let X and Y be object versions in working sets A and B respectively. If the row of the Object Version table that represents X contains within its columns the primary keys of the row representing object version Y, then X is said to "reference" Y. For such a reference to be valid, either X and Y must be contained in the same working set, or a reference path must first have been defined from A to B.

Where the columns of an IRD table contain references to object versions contained in other working sets (to which reference paths exist), the table effectively defines an aggregate object, i.e. it provides a means for the IRDS user to reference, as a single unit, a collection of object versions, of various object types, whose content may be controlled by some other IRDS user. The referenced objects may themselves in turn be aggregate objects, and in this way an IRDS user may define the specific hierarchical structures of objects that are required.

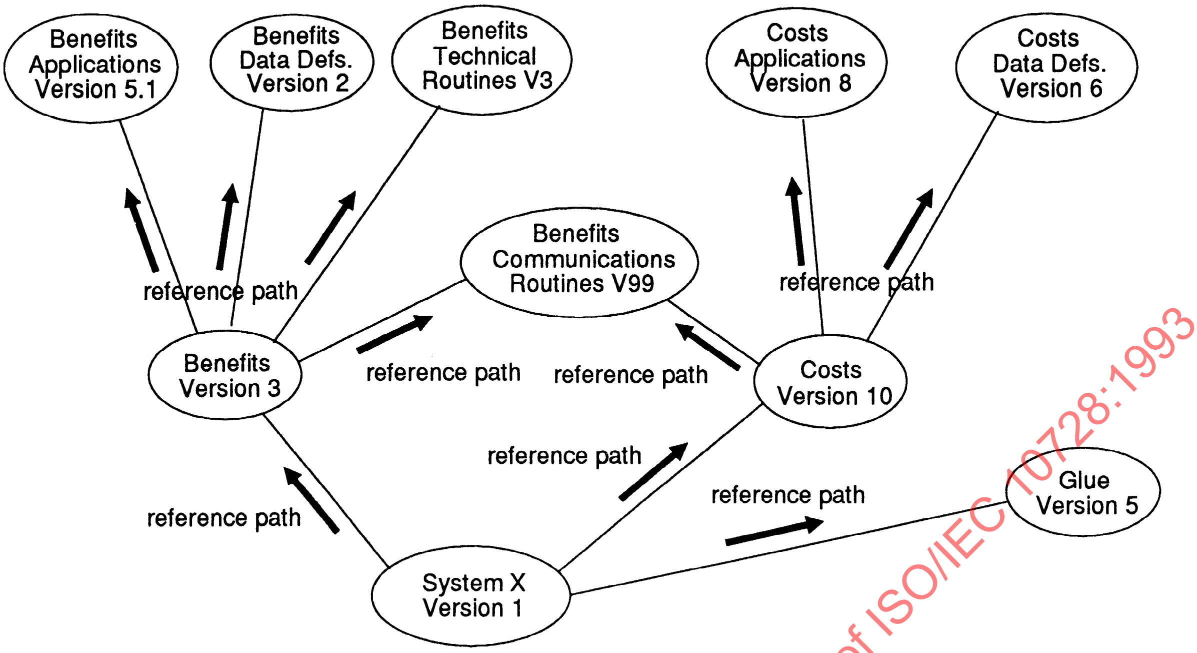

Benefits Applications Version 5.1
Benefits Data Defs. Version 2
Benefits Technical Routines V3
Costs Applications Version 8
Costs Data Defs. Version 6
Benefits Communications Routines V99
reference path
reference path
Benefits Version 3
reference path
reference path
Costs Version 10
reference path
Glue Version 5
reference path
reference path
System X Version 1

Figure 19 - Building an aggregate using reference paths

An example of such a structure is outlined in figure 19. The working set "BENEFITS VERSION 3" is the source of reference paths to four other working sets, and contains a table representing the aggregate object "BENEFITS VERSION 3", whose rows contain the required references to the subordinate objects (in one or other of the related working sets) of which it is composed.

In this way, the reference path mechanism enables hierarchical structures of collections of object versions to be defined.

It should be noted that if a single context is used (see below), it is only the references to the related objects, not the objects themselves, that are included in the materialization of the working set. However, by the use of a full context (see below) the materialization can be made to include all working sets in which referenced object versions could exist.

5.6.7 References to multiple versions of an object

It may occasionally be required to reference multiple versions of the same object. It is not possible to see multiple versions of an object in a single version path. However, this is possible using reference paths from the current working set to the specific working sets containing the required object versions.

5.6.8 Context

SQL query expressions are evaluated on a subset of the IRDS database defined by the materialization of a specified working set, possibly augmented by object versions from certain other working sets. The actual physical database is implementation-dependent; it is materialization that makes it appear to be an SQL database of tables, columns and rows. The single working set on which the materialization is based is said to provide the context for the operation.

If a reference path is defined from working set B to working set C, B is said to have a direct reference path to C. If working set A also has a direct reference path to working set B, A is said to have an indirect reference path to C. In some circumstances (see below), all working sets that are directly or indirectly referenced by the specified working set also contribute those object versions contained in them to the context. If object versions from the referenced working sets are included the context is called a "full context", otherwise it is called a "single context". The working set that defines the single context or the starting point for tracing references is called the "context specifier". It should be noted that in the case of a full context, the materialization process applies not only to the working set that is the context specifier, but also to any referenced working sets.

Every SQL clause used in this International Standard, or stored as part of an IRD Definition or IRD, must be evaluated within some context.

a) For services, either the session default context specifier is used or the context specifier is provided as a parameter to the service. For retrievals, the choice between a single context and a full context is controllable by parameter; for updates, only a single context is allowed.

b) SQL conditions in constraints have a context specifier that is the working set directly containing the object version. A full context is used, thereby allowing integrity constraints between object versions in different working sets but having access across reference paths.

c) SQL views are evaluated according to the context of the user of the view.

There are some services, such as Create Working Set, that do not require a context to be set. These services update one or more of the Internal tables, or of the tables in the Common or Environment working sets. For example, the Create Working Set service also creates a set of initial privileges for the working set. The precise operations are specified explicitly in clause 9, as operations on the abstract data structure specified in clause 6.

5.6.9 IRD content status

Each IRD working set shall have an attribute called IRD_CONTENT_STATUS. The IRDS user may define values of IRD content status, each of which shall have an IRD content status class of Uncontrolled, Controlled or Archived.

An object version may be added, updated or deleted only if it is contained in, or to be added to, an IRD working set whose IRD content status has an IRD content status class of Uncontrolled, or which is non-versionable.

To prevent unexpected side effects, any working set on which one or more other working sets are based (see 5.6.4) must always be in a Controlled IRD content status.

Only a user with suitable privileges is empowered to invoke the services that modify the IRD Content Status of an IRD working set.

By modifying the IRD Content Status, a working set may be reclassified from any Content Status Class to any other, provided that the constraints specified in 6.3.2 and 9.2.12 are not violated. Whether or not constraints shall be enforced is also specified through the choice of IRD Content Status associated with the working set.

5.6.10 References in the IRD

An IRD object version may refer to a different IRD object version by means of a reference conforming to a referential integrity constraint. The reference is to the object version, not to the object. Each such reference is made by the values of a pair of columns, one identifying the object of which a version is referenced, and the other identifying the working set in which the desired version of that object exists.

Where a reference is created or amended so as to refer to an object version outside the current working set materialization, such a reference shall only be allowed if there is a reference path defined from the referencing working set to the referenced working set.

5.6.11 Granularity of Version Control

The granularity of version control is under the control of the IRDS user. At one extreme, the user could create a new IRD working set prior to every single object-level change that he required to make. At the other, he could accumulate in one working set any number of changes from the working set upon which it is based. In practice, it is expected that each IRDS installation will set its own guide-lines for the application of version control: there could, for example, be user-supplied checks that any IRD working set used as the basis for another, or accessible by a reference path from another, is either of Controlled status, or else no longer the 'latest' with a particular working set name. Such checks set an effective upper bound on the granularity of version control.

5.6.12 Access control

All access to an IRD definition or IRD is controlled through the working set privilege mechanism specified in the Working Set Privilege table in clause 6, in conjunction with the working set facilities described above.

5.7 Naming facilities

A standard naming structure is provided for definition objects and dictionary objects. This makes it possible to distinguish all dictionary objects in an IRD by name. Other naming and naming-related facilities are provided in this standard.

5.7.1 Names

Each definition object version or dictionary object version about which data is recorded in an IRD or IRD Definition may have a unique name assigned by the user. The name is unique only within a single IRD or IRD Definition as appropriate.

This name has four components:

- IRDS name

- variation name
- working set name
- working set version name

5.7.2 IRDS names

An IRDS name may be assigned to a definition object or IRD object by the user when a row is added to an IRD Definition table or IRD table.

5.7.3 Variation name

A variation name may be used to qualify an IRDS name, differentiating between objects which are essentially similar but differ in some respect. Variation names have the same format as IRDS names, and have an implementor-defined maximum length.

5.7.4 Working set name and working set version name

An object version can only be accessed in the context of a working set. For uniqueness, the identifier of the working set is part of the identifier of each object version.

A working set is identified by a working set name and a working set version name.

These two components identify a specific version of an IRD object, which may have a given IRDS name and optionally a variation name. When an IRD object version is created it is given the IRDS name and variation name assigned to it by the IRDS user, if any, and the IRD working set name and IRD working set version name of the working set context the IRDS user has established using the Set Context service described in clause 9.

The purpose of the IRD working set name and version name is to identify a version of an object that is going through several versions during its development.

5.8 Definable limits and installation defaults

Limits may be defined by an implementor, by an installer or by an IRDS user. Installation defaults may be defined by an installer.

5.8.1 Implementation-defined limits

An implementation shall define maximum lengths for IRDS names and variation names. These apply to IRD Definition tables and to IRD tables. The check clauses on the Implementation Limits table in clause 6 specify minimum values for these implementation-defined maxima.

5.8.2 Installation defaults

The maximum and minimum permitted lengths of IRDS names may vary from one IRD table to another, although the maximum length of any name is constrained by that previously defined by the implementor. These maxima and minima for each IRD table are default values and may be over-ridden by user-defined values for each table. The defaults are specified in the Installation Defaults table for the IRD Definition and for each IRD.

5.9 Creating and dropping IRDs

For convenience each IRD is given an internal name (the IRDS_NAME of the associated IRDS Dictionary object) conforming to the normal format for IRDS names, as well as an external name conforming to an implementation-defined format. Thus Create IRD creates an IRD with user-specified internal and external names; all other services that reference an IRD by name use the internal name.

5.10 IRD schema modification

In responding to IRD level service requests, the IRDS must access the definition objects (such as IRD tables, IRD domains and IRD assertions) belonging to the IRD Schema Group referenced by an active IRD in order to validate and process each service request. However, from time to time it may be necessary to change the IRD Schema Group referenced by an IRD, either to correct errors or to extend the IRD Schema Group.

If IRD accesses and/or updates were to continue while these modifications were taking place, the integrity of the IRD might be compromised, and services might not operate correctly. Therefore all accesses and updates to the IRD must be disallowed while the referenced IRD Schema Group is subject to modification. The Deactivate IRD service does this. It must be invoked prior to using the Reactivate IRD service to make the IRD refer to a different IRD Schema Group.

For IRD activity to be resumed, it is necessary that the IRD Schema Group be self-consistent and consistent with the contents of the IRD. The Validate IRD Schema Group service may be used to carry out the first part of this checking; the Reactivate IRD service repeats this consistency check, then performs the checking that the contents of the IRD are consistent with the referenced IRD Schema Group, and if successful allows access to the IRD to be resumed.

5.11 Other added value functionality

5.11.1 Audit attributes

Audit attributes are associated with each row in each IRD Definition table or IRD table.

The audit attributes are as follows:

a) name of user adding the object version;

b) date/time when the object version was added;

c) name of user modifying the object version;

d) date/time when the object version was last modified;

e) number of times the object version has been modified since initial creation.

Each audit attribute is represented as a column in the IRD Object Version table. In addition, attributes a and b in the above list are represented as columns in the IRD Object table.

The audit attributes are automatically maintained by the IRDS Services Interface processor each time a change is made. Where a new object version is effectively created by basing one working set on another, the values of a and b above will be updated only when the object version is first modified, and will then remain unaltered throughout its life. The values of c, d and e, on the other hand, will be updated each time an object version is modified.

5.11.2 IRDS content modules

An IRDS content module may be defined in one of four ways:

a) in an International Standard;

b) in a national standard;

c) by an implementor of a standard IRDS;

d) by an installer or user of a copy of a standard IRDS.

The content modules that are present in an IRD Definition or IRD are listed in the IRD Definition table called IRD Module, where the name of each module indicates its source. Reference to this table enables a user of the IRDS Services Interface to discover which sets of definitions, defined for example in other international or national standards, are available in a particular IRD Definition or IRD.

5.11.3 System-maintained values

An IRD column in an IRD table may be designated as system-maintained. This means that the IRDS is responsible for maintaining the values in this column in an IRD.

6 Abstract data structures

6.1 IRD Definition Level

6.1.1 IRD Definition Level data structure

The definition of the IRD Definition Level consists of the definitions of each of its components (i.e. domains, tables, views and assertions), and the services to maintain them, that are required to support the IRDS functionality defined in this International Standard. This IRD Definition Level functionality then allows the content of the IRD Level to be defined. The IRD Definition Level data structure is described informally in 5.3, and defined formally below.

A description of each of the IRD Definition Level components follows. For each component, the following parts are included:

a) function

b) definition in SQL

c) a description in narrative form of the purpose of each element of the component. In the case that the component is a table, this includes both the columns of the table and the constraints referring to other tables.

It should be noted that much of the structure of the content of each IRD is identical to the structure of the content of an IRD Definition, and that clause 6.2 defines these common features by reference to 6.1. Unless otherwise stated, therefore, what follows in 6.1, with the exception of those tables identified in clause 5 as IRD-specific, is applicable to both an IRD Definition and an IRD.

The SQL definition takes precedence over the natural language description.

6.1.2 IRD Definition Level Schema

6.1.2.1 Schema IRD Definition

Function

Create the SQL schema that specifies the definition level of the IRD.

Definition

```
CREATE SCHEMA IRD_DEF
        AUTHORIZATION IRD_DEF
```

Description

IRD_DEF is the SQL schema that contains the following definitions of SQL domains, SQL tables and views, and SQL assertions, that together comprise the IRD Definition.

The choice of authorization identifier is arbitrary.

6.1.3 IRD Definition Level Domains

6.1.3.1 Domain SQL Name

Function

Define a domain that contains all valid SQL Names.

Definition

CREATE DOMAIN SQL_NAME AS CHARACTER VARYING (128)

Description

This domain specifies any varying character value that conforms to the rules for an SQL_NAME (see ISO/IEC 9075).

6.1.3.2 Domain IRDS Key

Function

Define a domain that contains all valid IRDS Keys

Definition

CREATE DOMAIN IRDS_KEY AS CHARACTER VARYING (KL)

Description

1 This domain specifies any IRDS_KEY. Since an IRDS_KEY is a unique key for all objects and definition objects within the scope of one IRD Definition, it is internally generated, and is hence of little concern to the user.

2 Some IRD tables have a first column whose name is the name of the table suffixed with KEY, whose domain is IRDS_KEY, and which is the (unary) primary key. Others (those which are subtables of IRD_OBJECT_VERSION) have a pair of columns whose names are the name of the table suffixed with _OBJ_KEY and _WS_KEY respectively, whose domains are IRDS_KEY and which together form the primary key.

3 KL is the implementation-defined maximum length of an IRDS_KEY.

6.1.3.3 Domain Char Data

Function

Define a domain that contains any character data.

Definition

CREATE DOMAIN CHAR_DATA AS CHARACTER VARYING (ML)

Description

1 This domain specifies any character data, and is used for columns that contain words such as 'PRIMARY KEY' and so forth.

2 ML is the (SQL) implementation-defined maximum length of a varying character string.

6.1.3.4 Domain Cardinal

Function

Define a domain that contains a non-negative number.

Definition

CREATE DOMAIN CARDINAL AS INTEGER CHECK (VALUE >= 0)

Description

1 The domain CARDINAL contains every non-negative number that is less than the implementation-defined maximum for INTEGER (i.e., the implementation-defined value of NUMERIC_PREC_RADIX raised to the power of implementation-defined NUMERIC_PRECISION). For a fuller explanation of the terms used here see the description of table 11 (IRD Data type descriptor).

6.1.3.5 Domain Boolean

Function

Define a domain that contains the two values 'TRUE' and 'FALSE'.

Definition

```
CREATE DOMAIN BOOLEAN AS CHARACTER VARYING (5)
    CHECK ( VALUE IN ('TRUE', 'FALSE') )
```

Description

1 The domain BOOLEAN contains the values 'TRUE' and 'FALSE'.

6.1.4 IRD Definition Level Tables

6.1.4.1 Table IRD Object

Function

This Internal table contains, for each row in each IRD definition level (sub) table (representing an IRD definition object), the names of the object, and audit data about who first created a version of it, plus the date and time when this creation was performed.

The purpose of this table is to simplify the representation of common properties, and of naming and other rules. The table is system-maintained, and cannot be accessed directly by the user. Access to the table is provided through IRD_OBJECT_VERSION_VIEW.

Definition

```
CREATE TABLE IRD_OBJECT
    (
    IRD_OBJECT_KEY IRDS_KEY PRIMARY KEY ,

    IRDS_NAME CHAR (imp_name_lim) ,

    IRD_VAR_NAME CHAR (imp_var_lim) ,

    IRD_MODULE_OBJ_KEY IRDS_KEY ,
    IRD_MODULE_WS_KEY IRDS_KEY ,
    CONSTRAINT IRD_OBJECT_INTRODUCED_BY_IRD_MODULE
        FOREIGN KEY (IRD_MODULE_OBJ_KEY, IRD_MODULE_WS_KEY)
            REFERENCES IRD_MODULE
            ON DELETE CASCADE ,

    ADD_BY_OBJ_KEY IRDS_KEY ,
    ADD_BY_WS_KEY IRDS_KEY ,
    CONSTRAINT  IRD_OBJECT_ADDED_BY_IRDS_USER
        FOREIGN KEY (ADD_BY_OBJ_KEY, ADD_BY_WS_KEY)
            REFERENCES IRDS_USER ,
    DATE_TIME_ADD  TIMESTAMP NOT NULL ,

    PURPOSE CHAR_DATA ,

    UNIQUE (IRDS_NAME, IRD_VAR_NAME)
    )
```

Description

1 IRD_OBJECT_KEY is the implementation-generated key that is used as part of the primary key for an IRD Table.

2 IRDS_NAME is the optional IRDS name of a row. If not null, this name shall conform to the rules for an SQL name (see ISO/IEC 9075).

3 IRD_VAR_NAME is the optional variation name applicable to this row. If not null, this name shall conform to the rules for an SQL name (see ISO/IEC 9075).

4 IRD_MODULE_OBJ_KEY and IRD_MODULE_WS_KEY together form a foreign key referencing the IRD Module table, indicating the module of IRDS definitions that introduced this Object.

5 ADD_BY_OBJ_KEY and ADD_BY_WS_KEY together form a foreign key referencing the IRDS User who added the row

6 DATE_TIME_ADD is the date and time at which the row was added. The SQL data type is as defined in ISO/IEC 9075.

7 PURPOSE may optionally contain text indicating the reason for the existence of the Object.

6.1.4.2 Table IRD Working Set

Function

Each row of this Internal table represents an IRD working set.

Definition

```
CREATE TABLE IRD_WORKING_SET
    (
    IRD_WORKING_SET_KEY IRDS_KEY PRIMARY KEY ,

    WORKING_SET_NAME CHAR (imp_name_lim),
    WORKING_SET_VERSION_NAME  CHAR (imp_name_lim),

    BASED_ON_WORKING_SET_KEY IRDS_KEY
    CONSTRAINT IRD_WORKING_SET_BASED_ON_IRD_WORKING_SET
        REFERENCES IRD_WORKING_SET
        ON DELETE SET NULL ,

    IRD_CONTENT_STATUS_OBJ_KEY IRDS_KEY ,
    IRD_CONTENT_STATUS_WS_KEY IRDS_KEY ,
    CONSTRAINT     IRD_WORKING_SET_HAS_IRD_CONTENT_STATUS
    FOREIGN KEY (IRD_CONTENT_STATUS_OBJ_KEY, IRD_CONTENT_STATUS_WS_KEY)
        REFERENCES IRD_CONTENT_STATUS ,

    VERSIONABLE BOOLEAN NOT NULL,
    ADD_BY_OBJ_KEY IRDS_KEY ,
    ADD_BY_WS_KEY IRDS_KEY ,
    CONSTRAINT  IRD_WORKING_SET_ADDED_BY_IRDS_USER
        FOREIGN KEY (ADD_BY_OBJ_KEY, ADD_BY_WS_KEY)
            REFERENCES IRDS_USER ,

    DATE_TIME_ADD  TIMESTAMP NOT NULL ,

    CONSTRAINT IRD_WORKING_SET_NAME_UNIQUE
        UNIQUE (WORKING_SET_NAME, WORKING_SET_VERSION_NAME ) )
```

Description

1 IRD_WORKING_SET_KEY is the IRDS_KEY of the working set

2 WORKING_SET_NAME is the name of the working set, which together with WORKING_SET_VERSION_NAME can be used to identify an IRD working set

3 WORKING_SET_VERSION_NAME is the name of the version, which together with WORKING_SET_NAME can be used to identify an IRD working set version.

4 BASED_ON_WORKING_SET_KEY is the IRDS_KEY of the working set on which this one was based, or null if there was no basis working set

5 IRD_CONTENT_STATUS_OBJ_KEY and IRD_CONTENT_STATUS_WS_KEY together form the primary key of the IRD content status applicable to this working set.

6 VERSIONABLE If 'TRUE', indicates that this working set can be based on another working set, and can be the basis for one or more other working sets. If 'FALSE', indicates that BASED_ON_WORKING_SET_KEY must be null, and that no other working set may have a value of BASED_ON_WORKING_SET_KEY equal to the WORKING_SET_KEY of this working set.

7 ADD_BY_OBJ_KEY and ADD_BY_WS_KEY together form a foreign key referencing the IRDS User who added the IRD working set

8 DATE_TIME_ADD is the date and time at which the IRD working set was added. The SQL data type is as defined in ISO/IEC 9075.

6.1.4.3 Table IRD Object Version

Function

This Internal table contains, for each row in each IRD table, (representing a version of an IRD object), audit data about who most recently modified that row, plus the date and time when this modification was performed.

The purpose of this table is to simplify the representation of audit attributes and the specification of services. The table is system-maintained, and cannot be accessed directly by the user. Access to the table is provided through IRD_OBJECT_VERSION_VIEW.

Definition

```
CREATE TABLE IRD_OBJECT_VERSION
    (
    IRD_OBJECT_KEY IRDS_KEY
        CONSTRAINT
        IRD_OBJECT_VERSION_IS_VERSION_OF_IRD_OBJECT
            REFERENCES IRD_OBJECT
            ON DELETE CASCADE ,
IRD_WORKING_SET_KEY IRDS_KEY NOT NULL
    CONSTRAINT
    IRD_OBJECT_VERSION_CONTAINED_IN_IRD_WORKING_SET
        REFERENCES IRD_WORKING_SET
        ON DELETE CASCADE,

PRIMARY KEY (IRD_OBJECT_KEY, IRD_WORKING_SET_KEY) ,

VERS_ADD_BY_OBJ_KEY IRDS_KEY NOT NULL ,
VERS_ADD_BY_WS_KEY IRDS_KEY NOT NULL ,
CONSTRAINT IRD_OBJECT_VERSION_ADDED_BY_IRDS_USER
    FOREIGN KEY (VERS_ADD_BY_OBJ_KEY, VERS_ADD_BY_WS_KEY)
        REFERENCES IRDS_USER ,

DATE_TIME_VERS_ADD TIMESTAMP NOT NULL ,

VERS_MOD_BY_OBJ_KEY IRDS_KEY ,
VERS_MOD_BY_WS_KEY IRDS_KEY ,
CONSTRAINT IRD_OBJECT_VERSION_MODIFIED_BY_IRDS_USER
    FOREIGN KEY (VERS_MOD_BY_OBJ_KEY, VERS_MOD_BY_WS_KEY)
        REFERENCES IRDS_USER ,

DATE_TIME_VERS_MOD TIMESTAMP ,

TIMES_MOD CARDINAL NOT NULL ,

    CONSTRAINT MOD_BY_COLUMNS_CONSISTENT
        CHECK ( ( TIMES_MOD = 0 AND
                    (VERS_MOD_BY_OBJ_KEY, VERS_MOD_BY_WS_KEY, DATE_TIME_VERS_MOD)
                        IS NULL )
                OR ( TIMES_MOD <> 0 AND
                    (VERS_MOD_BY_OBJ_KEY, VERS_MOD_BY_WS_KEY, DATE_TIME_VERS_MOD)
                        IS NOT NULL ) )
)
```

Description

1 IRD_OBJECT_KEY is the IRDS_KEY that identifies the IRD object of which this object version is a version.

2 IRD_WORKING_SET_KEY is the IRDS_KEY of the IRD Working Set in which this object version exists.

3 VERS_ADD_BY_OBJ_KEY and VERS_ADD_BY_WS_KEY together form a foreign key referencing the IRDS User who added the row.

4 DATE_TIME_VERS_ADD is the date and time at which the row was added. The SQL data type is as defined in ISO/IEC 9075.

5 VERS_MOD_BY_OBJ_KEY and VERS_MOD_BY_WS_KEY together form a foreign key referencing the IRDS User who most recently modified the row.

6 DATE_TIME_VERS_MOD is the date and time at which the row was most recently modified. The SQL data type is as defined in ISO/IEC 9075.

7 TIMES_MOD is the number of times that the row has been modified since it was first added.

6.1.4.4 Table IRD Reference Path

Function

Each row of this Internal table indicates a reference path from an IRD working set F to another IRD working set T. The existence of the reference path allows object versions in the materialization of F to reference object versions in the materialization of T.

Definition

```
CREATE TABLE IRD_REFERENCE_PATH
    (
    REFTO_WORKING_SET_KEY IRDS_KEY NOT NULL,
    REFFROM_WORKING_SET_KEY IRDS_KEY NOT NULL,
    PRIMARY KEY (REFTO_WORKING_SET_KEY, REFFROM_WORKING_SET_KEY) ,
        CONSTRAINT IRD_REFERENCE_PATH_REFERS_TO_IRD_WORKING_SET
            REFERENCES IRD_WORKING_SET ,

        CONSTRAINT IRD_REFERENCE_PATH_REFERRED_TO_FROM_IRD_WORKING_SET
            REFERENCES IRD_WORKING_SET
            ON DELETE CASCADE ,

    CONSTRAINT REFTO_WS_DISTINCT_FROM_REFFROM_WS
        CHECK(REFTO_WORKING_SET_KEY <> REFFROM_WORKING_SET_KEY)
    )
```

Description

1 REFTO_WORKING_SET_KEY and REFFROM_WORKING_SET_KEY together form the primary key of the IRD Reference Path table.

2 REFFROM_WORKING_SET_KEY is the IRDS_KEY of the IRD working set from which references are permitted along the specified reference path.

3 REFTO_WORKING_SET_KEY is the IRDS_KEY of the IRD working set to which references are permitted along the specified reference path.

6.1.4.5 Table IRDS User

Function

This Environment table has one row for each IRDS User in the IRDS environment. These are all those IRDS Users that may grant or be granted privileges, those who have added rows to or modified the rows of IRD definition table or IRD tables, and those that own or may create a schema.

Definition

```
CREATE TABLE IRDS_USER
    (
    IRDS_USER_OBJ_KEY IRDS_KEY NOT NULL ,
    IRDS_USER_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRDS_USER_OBJ_KEY, IRDS_USER_WS_KEY) ,

    IRDS_USER_NAME CHAR (128) NOT NULL UNIQUE ,

    IRDS_USER_ACTIVE BOOLEAN NOT NULL ,

    DEFAULT_WORKING_SET_KEY IRDS_KEY ,

    IRDS_USER_MAY_CREATE_IRD BOOLEAN NOT NULL ,
    IRDS_USER_MAY_CREATE_WS BOOLEAN NOT NULL ,
    IRDS_USER_MAY_CREATE_REF_PATH BOOLEAN NOT NULL ,
    IIRDS_USER_MAY_MODIFY_REF_PATH BOOLEAN NOT NULL ,
    IIRDS_USER_MAY_DROP_REF_PATH BOOLEAN NOT NULL ,
    )
```

Description

1 The means by which rows are inserted into and deleted from this table are implementation-defined.

2 IRDS_USER_OBJ_KEY and IRDS_USER_WS_KEY together form the primary key of the IRDS User table.

3 IRDS_USER_NAME is the name of the IRDS_USER. The values of IRDS_USER_NAME are all the authorization identifiers that are known in the IRDS environment. If IRDS_USER_NAME is 'PUBLIC', then the row represents all users, and privileges whose grantee is the IRDS_USER_KEY of this row are available to all.

4 If IRDS_USER_ACTIVE is 'TRUE' the row represents an IRDS User who is currently permitted to exercise all the privileges for which he is recorded as grantee in some privilege descriptor.

If IRDS_USER_ACTIVE is 'FALSE' the row represents an IRDS User who is not currently permitted to exercise any privileges for which he is recorded as grantee in some privilege descriptor.

5 DEFAULT_WORKING_SET_KEY specifies the IRD working set to be set as the working set context by the Open IRDS service. If it is null, there is no default, and a context must be specified by invoking the Set Context service.

6 If IRDS_USER_MAY_CREATE_IRD is true, the IRDS User represented by this row may invoke the Create IRD Service; if false, he may not.

7 If IRDS_USER_MAY_CREATE_WS is true, the IRDS User represented by this row may invoke the Create Working Set Service; if false, he may not.

8 If IRDS_USER_MAY_CREATE_REF_PATH is true, the IRDS User represented by this row may invoke the Create Reference Path Service; if false, he may not.

9 If IRDS_USER_MAY_MODIFY_REF_PATH is true, the IRDS User represented by this row may invoke the Modify Reference Path Service; if false, he may not.

10 If IRDS_USER_MAY_DROP_REF_PATH is true, the IRDS User represented by this row may invoke the Drop Reference Path Service; if false, he may not.

6.1.4.6 Table Implementation Limits

Function

This Environment table contains a single row that records implementation-defined limits. This row is defined by the implementation, and cannot be modified or deleted.

Definition

```
CREATE TABLE IMP_LIMITS
    (
    IMP_LIMITS_OBJ_KEY IRDS_KEY NOT NULL ,
    IMP_LIMITS_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IMP_LIMITS_OBJ_KEY, IMP_LIMITS_WS_KEY) ,

    IMP_NAME_LIM CARDINAL NOT NULL
        CONSTRAINT IMP_NAME_LIM_RANGE_CHECK
        CHECK (IMP_NAME_LIM BETWEEN 31 AND 255) ,
    IMP_INT_LIM CARDINAL NOT NULL
        CONSTRAINT IMP_INT_LIM_RANGE_CHECK
        CHECK (IMP_INT_LIM >= 32767) ,

    IMP_TEXT_LIM CARDINAL NOT NULL
        CONSTRAINT IMP_TEXT_LIM_RANGE_CHECK
        CHECK (IMP_TEXT_LIM >= 72) ,

    IMP_VAR_LIM     CARDINAL NOT NULL
        CONSTRAINT IMP_VAR_LIM_RANGE_CHECK
        CHECK (IMP_VAR_LIM >= 8) ,

    IMP_DIC_NAME_LEN CARDINAL NOT NULL
        CONSTRAINT IMP_DIC_NAME_LEN_RANGE_CHECK
        CHECK (IMP_NAME_LIM BETWEEN 31 AND 255) ,

    CONSTRAINT IMP_LIMITS_HAS_EXACTLY_ONE_ROW
    CHECK ( 1 = SELECT COUNT(*) FROM IMP_LIMITS )
    )
```

Description

1 IMP_LIMITS_OBJ_KEY and IMP_LIMITS_WS_KEY together form the primary key of the Implementation Limits table.

2 IMP_NAME_LIM is the implementation limit for the length of the IRDS name for a row in an IRD Table. No name shall be longer than the value specified here.

3 IMP_INT_LIM is the implementation limit for the largest integer value. No numeric attribute IRD Column value shall exceed the value specified here.

4 IMP_TEXT_LIM is the implementation limit for the longest text value. No character value shall be longer than the value specified here.

5 IMP_VAR_LIM implementation limit for the length of a Variation Name. No Variation Name shall be longer than the length specified here.

6 IMP_DIC_NAME_LEN is the implementation-defined length of the external dictionary name used in table 7 and in the Create IRD Definition and Create IRD services.

6.1.4.7 Table IRDS Dictionary

Function

This Environment table has one row for the IRD Definition and one row for each IRD created within the IRD Definition.

Definition

```
CREATE TABLE IRDS_DICTIONARY
    (
    IRDS_DICTIONARY_OBJ_KEY IRDS_KEY NOT NULL ,
    IRDS_DICTIONARY_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRDS_DICTIONARY_OBJ_KEY, IRDS_DICTIONARY_WS_KEY) ,

    DICTIONARY_NAME CHAR (NL) NOT NULL UNIQUE ,

    DICTIONARY_ACTIVE BOOLEAN NOT NULL ,

    DEFINED_BY_SCHEMA_GROUP_OBJ_KEY IRDS_KEY ,
    DEFINED_BY_SCHEMA_GROUP_WS_KEY IRDS_KEY ,
    CONSTRAINT DICTIONARY_DEFINED_BY_SCHEMA_GROUP
        FOREIGN KEY (DEFINED_BY_SCHEMA_GROUP_OBJ_KEY,
                DEFINED_BY_SCHEMA_GROUP_WS_KEY )
            REFERENCES IRD_SCHEMA_GROUP

    )
```

Description

1 Rows are inserted into this table only by the services Create IRD Definition and Create IRD, and deleted only by the services Drop IRD Definition and Drop IRD. Only three columns of this table may be modified, all implicitly: the column DICTIONARY_ACTIVE, which is initially true, is set false by Deactivate IRD, and is set true by Reactivate IRD; and the pair of columns DEFINED_BY_SCHEMA_GROUP_OBJ_KEY, DEFINED_BY_SCHEMA_GROUP_WS_KEY, which may be modified by a successful Reactivate IRD service.

2 IRDS_DICTIONARY_OBJ_KEY and IRDS_DICTIONARY_WS_KEY together form the primary key of the IRDS Dictionary table.

3 DICTIONARY_NAME is the name of the IRDS_DICTIONARY, in an implementation-defined format. NL is the implementation-defined length of a dictionary name, specified by the IMP_DIC_NAME_LEN column in the Implementation Limits table.

4 If DICTIONARY_ACTIVE is 'TRUE' the row represents an IRDS Dictionary that is currently accessible to IRDS services;

If DICTIONARY_ACTIVE is 'FALSE' the row represents an IRDS Dictionary that has been deactivated, and will not be accessible to other IRDS services until the Reactivate IRD service has been successfully invoked.

5 DEFINED_BY_SCHEMA_GROUP_OBJ_KEY and DEFINED_BY_SCHEMA_GROUP_WS_KEY together identify the Schema Group that was referenced when the IRD was created or most recently reactivated. Both are NULL if the row represents the IRD Definition.

6.1.4.8 Table IRD Schema Group

Function

Each row of this table names a set of IRD Domains, IRD Tables and IRD Assertions that together specify one IRD Database. The inclusion of IRD Tables in an IRD Schema Group is subject to the constraint that it shall not contain a reference to any IRD Table that is not contained in the same IRD Schema Group.

Definition

```
CREATE TABLE IRD_SCHEMA_GROUP
    (
    IRD_SCHEMA_GROUP_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_SCHEMA_GROUP_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_SCHEMA_GROUP_OBJ_KEY, IRD_SCHEMA_GROUP_WS_KEY) ,

    IRD_SCHEMA_GROUP_NAME SQL_NAME NOT NULL )
```

Description

1 IRD_SCHEMA_GROUP_OBJ_KEY and IRD_SCHEMA_GROUP_WS_KEY together form the primary key of the IRD Schema Group table.

2 IRD_SCHEMA_GROUP_NAME is the SQL name of the IRD Schema Group.

6.1.4.9 Table IRD Schema

Function

Each row of this table names a set of IRD Domains, IRD Tables and IRD Assertions.

Definition

```
CREATE TABLE IRD_SCHEMA
    (
    IRD_SCHEMA_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_SCHEMA_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_SCHEMA_OBJ_KEY, IRD_SCHEMA_WS_KEY) ,

    IRD_SCHEMA_NAME SQL_NAME NOT NULL UNIQUE
    )
```

Description

1 IRD_SCHEMA_OBJ_KEY and IRD_SCHEMA_WS_KEY together form the primary key of the IRD Schema table.

2 IRD_SCHEMA_NAME is the SQL name of the IRD Schema.

6.1.4.10 Table IRD Schema Reference

Function

Each row of this table represents the inclusion of the referenced IRD Schema in the referenced IRD Schema Group.

Definition

```
CREATE TABLE IRD_SCHEMA_REFERENCE
    (
    IRD_SCHEMA_REFERENCE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_SCHEMA_REFERENCE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_SCHEMA_REFERENCE_OBJ_KEY, IRD_SCHEMA_REFERENCE_WS_KEY) ,

    IRD_REF_SCHEMA_GROUP_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_REF_SCHEMA_GROUP_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_SCHEMA_REFERENCE_FROM_IRD_SCHEMA_GROUP
        FOREIGN KEY (IRD_REF_SCHEMA_GROUP_OBJ_KEY, IRD_REF_SCHEMA_GROUP_WS_KEY)
            REFERENCES IRD_SCHEMA_GROUP
            ON DELETE CASCADE ,

    IRD_REF_SCHEMA_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_REF_SCHEMA_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_SCHEMA_REFERENCE_TO_IRD_SCHEMA
        FOREIGN KEY (IRD_REF_SCHEMA_OBJ_KEY, IRD_REF_SCHEMA_WS_KEY)
            REFERENCES IRD_SCHEMA

    )
```

Description

1 IRD_SCHEMA_REFERENCE_OBJ_KEY and IRD_SCHEMA_REFERENCE_WS_KEY together form the primary key of the IRD Schema Reference table.

2 IRD_REF_SCHEMA_GROUP_OBJ_KEY and IRD_REF_SCHEMA_GROUP_WS_KEY together form the primary key of the IRD Schema Group making the reference, i.e. containing the referenced IRD Schema.

3 IRD_REF_SCHEMA_OBJ_KEY and IRD_REF_SCHEMA_WS_KEY together form the primary key of the IRD Schema referenced, i.e. forming part of the IRD Schema Group.

6.1.4.11 Table IRD Data Type Descriptor

Function

Each row in this table represents an IRD Data type descriptor that is referenced either by an IRD Column or an IRD Domain.

Definition

```
CREATE TABLE IRD_DATA_TYPE_DESCRIPTOR
    (
    IRD_DATA_TYPE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_DATA_TYPE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_DATA_TYPE_OBJ_KEY, IRD_DATA_TYPE_WS_KEY) ,

    IRD_DOMAIN_OBJ_KEY IRDS_KEY ,
    IRD_DOMAIN_WS_KEY IRDS_KEY ,
    UNIQUE (IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY) ,
    CONSTRAINT IRD_DATA_TYPE_DESCRIPTOR_REFERENCES_IRD_DOMAIN
        FOREIGN KEY (IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY)
            REFERENCES IRD_DOMAIN
            ON DELETE CASCADE ,

    IRD_COLUMN_OBJ_KEY IRDS_KEY ,
    IRD_COLUMN_WS_KEY IRDS_KEY ,
    UNIQUE (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY) ,
    CONSTRAINT IRD_DATA_TYPE_DESCRIPTOR_REFERENCES_IRD_COLUMN
        FOREIGN KEY (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY)
            REFERENCES IRD_COLUMN
            ON DELETE CASCADE ,

     CONSTRAINT IRD_DATA_TYPE_DESCRIPTOR_REFERENCES_IRD_DOMAIN_OR_IRD_COLUMN
        CHECK (
                ( IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY ) IS NULL AND
                ( IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY ) IS NOT NULL
                OR
                ( IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY) IS NOT NULL AND
                ( IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY) IS NULL
                ) ,

    DATA_TYPE CHAR_DATA NOT NULL ,
    CHAR_MAX_LENGTH CARDINAL ,
    NUMERIC_PRECISION CARDINAL ,
    NUMERIC_PREC_RADIX CARDINAL ,
    NUMERIC_SCALE CARDINAL ,
    DATETIME_PRECISION CARDINAL ,
CONSTRAINT IRD_DATA_TYPE_CONSISTENT
    CHECK
        (
            DATA_TYPE IN    ('CHARACTER',
                             'NATIONAL CHARACTER',
                             'CHARACTER VARYING',
                             'NATIONAL CHARACTER VARYING'
                             )
            AND CHAR_MAX_LENGTH IS NOT NULL
            AND (NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
                        NUMERIC_SCALE, DATETIME_PRECISION) IS NULL
        OR
            DATA_TYPE IN ('REAL', 'DOUBLE PRECISION', 'FLOAT')
            AND CHAR_MAX_LENGTH IS NULL
            AND (NUMERIC_PRECISION, NUMERIC_PREC_RADIX) IS NOT NULL
            AND NUMERIC_PREC_RADIX = 2
            AND NUMERIC_SCALE IS NULL
            AND DATETIME_PRECISION IS NULL
        OR
            DATA_TYPE IN ('INTEGER', SMALLINT', 'NUMERIC', 'DECIMAL')
            AND CHAR_MAX_LENGTH IS NULL
            AND (NUMERIC_PRECISION, NUMERIC_PREC_RADIX) IS NOT NULL
            AND NUMERIC_SCALE IS NOT NULL
            AND       (    NUMERIC_SCALE <> 0 AND NUMERIC_PREC_RADIX = 10
                           OR
                           NUMERIC_SCALE = 0 AND NUMERIC_PREC_RADIX IN (2, 10)
                      )
            AND DATETIME_PRECISION IS NULL
        OR
            DATA_TYPE IN ('DATE', 'TIME', 'TIMESTAMP', 'INTERVAL')
            AND CHAR_MAX_LENGTH IS NULL
            AND (NUMERIC_PRECISION, NUMERIC_PREC_RADIX) IS NULL
            AND NUMERIC_SCALE IS NULL
            AND DATETIME_PRECISION IS NOT NULL
        )
)
```

Description

1 IRD_DATA_TYPE_OBJ_KEY and IRD_DATA_TYPE_WS_KEY together form the primary key of the Data Type Descriptor table.

2 If the IRD Data_type_descriptor belongs to an IRD Domain, then the combination of IRD_DOMAIN_OBJ_KEY and IRD_DOMAIN_WS_KEY is the primary key of that IRD Domain, otherwise both are null.

3 If the IRD Data_type_descriptor belongs to an IRD Column, then the combination of IRD_COLUMN_OBJ_KEY and IRD_COLUMN_WS_KEY is the primary key of that IRD Column, otherwise both are null.

4 The value of DATA_TYPE, CHAR_MAX_LENGTH, NUMERIC_PRECISION, NUMERIC_PREC_RADIX, NUMERIC_SCALE, and DATETIME_PRECISION contain the data type of the IRD domain or column to which the IRD Data type descriptor belongs, the maximum length of the IRD domain or column if it is a character or national character type, the precision and radix of the precision if it is a numeric type, and the datetime precision if it is a datetime or interval type. These terms are defined in ISO/IEC 9075.

6.1.4.12 Table IRD Domain

Function

Each row in this IRD Definition Table specifies an IRD_Domain, consisting of a data type, an optional default value and an optional IRD check constraint.

If a default value is required for an IRD column, and that IRD column does not have one, but it is based on an IRD domain, then the default value for the IRD domain, if any, shall be used.

If an IRD domain has an IRD domain constraint, it shall never be false for any value of any IRD Column that is defined on that IRD domain.

Definition

```
CREATE TABLE IRD_DOMAIN
    (
    IRD_DOMAIN_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_DOMAIN_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY) ,

    IRD_DOMAIN_NAME SQL_NAME NOT NULL ,

    IRD_DOMAIN_CONSTRAINT_NAME SQL_NAME NOT NULL ,

    IRD_DOMAIN_SCHEMA_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_DOMAIN_SCHEMA_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_DOMAIN_CONTAINED_IN_IRD_SCHEMA
        FOREIGN KEY (IRD_DOMAIN_SCHEMA_OBJ_KEY, IRD_DOMAIN_SCHEMA_WS_KEY)
            REFERENCES IRD_SCHEMA
            ON DELETE CASCADE ,

    IRD_DOMAIN_DEFAULT CHAR_DATA ,

    IRD_DOMAIN_CONSTRAINT_OBJ_KEY IRDS_KEY ,
    IRD_DOMAIN_CONSTRAINT_WS_KEY IRDS_KEY ,
        CONSTRAINT IRD_DOMAIN_CONSTRAINED_BY_IRD_CHECK_CONSTRAINT
        FOREIGN KEY (IRD_DOMAIN_CONSTRAINT_OBJ_KEY, IRD_DOMAIN_CONSTRAINT_WS_KEY)
            REFERENCES IRD_CHECK_CONSTRAINT
            ON DELETE SET NULL ,

    CONSTRAINT IRD_DOMAIN_HAS_DATA_TYPE_DESCRIPTOR
    CHECK
        ( ( SELECT COUNT (*) FROM IRD_DATA_TYPE_DESCRIPTOR AS X
                WHERE ( IRD_DOMAIN_OBJ_KEY = X.IRD_DOMAIN_OBJ_KEY
                    AND IRD_DOMAIN_WS_KEY = X.IRD_DOMAIN_WS_KEY ) )
            = 1
        ),

    UNIQUE (IRD_DOMAIN_SCHEMA_OBJ_KEY, IRD_DOMAIN_SCHEMA_WS_KEY, IRD_DOMAIN_NAME)
    )
```

Description

1 IRD_DOMAIN_OBJ_KEY and IRD_DOMAIN_WS_KEY together form the primary key of the IRD Domain table.

2 IRD_DOMAIN_NAME is the SQL name of the IRD domain

3 IRD_DOMAIN_CONSTRAINT_NAME if the SQL name of the constraint applicable to the IRD domain

4 IRD_DOMAIN_SCHEMA_OBJ_KEY and IRD_DOMAIN_SCHEMA_WS_KEY together form the primary key of the IRD schema that contains the domain, and within which its name is unique.

5 The value of IRD_DOMAIN_DEFAULT is null if the IRD domain being described has no default value. Otherwise, IRD_DOMAIN_DEFAULT contains a literal that obeys the rules of ISO/IEC 9075 subclause <literal>, and specifies the default value for the IRD domain.

6 IRD_DOMAIN_CONSTRAINT_OBJ_KEY and IRD_DOMAIN_CONSTRAINT_WS_KEY together form the primary key of the IRD domain constraint, if any, for this IRD domain.

6.1.4.13 Table IRD Table

Function

The IRD_TABLE table contains one row for each IRD table in the IRDS environment, including IRD views. It effectively contains a representation of the IRD table descriptors.

Definition

```
CREATE TABLE IRD_TABLE
    (
    IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY) ,

    IRD_TABLE_NAME SQL_NAME NOT NULL ,

    IRD_TABLE_SCHEMA_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_SCHEMA_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_TABLE_CONTAINED_IN_IRD_SCHEMA
        FOREIGN KEY (IRD_TABLE_SCHEMA_OBJ_KEY, IRD_TABLE_SCHEMA_WS_KEY)
            REFERENCES IRD_SCHEMA
            ON DELETE CASCADE ,

    SUPERTABLE_OBJ_KEY IRDS_KEY ,
    SUPERTABLE_WS_KEY IRDS_KEY ,
        CHECK ( ( ( SUPERTABLE_OBJ_KEY IS NULL ) AND (SUPERTABLE_WS_KEY IS NULL ) )
            OR
            ( ( SUPERTABLE_OBJ_KEY IS NOT NULL ) AND (SUPERTABLE_WS_KEY IS NOT NULL ) ) )
    CONSTRAINT SUPERTABLE_PRESENT
        FOREIGN KEY (SUPERTABLE_OBJ_KEY, SUPERTABLE_WS_KEY)
            REFERENCES IRD_TABLE
            ON DELETE CASCADE ,

    MAX_NAME_LEN CARDINAL
        CONSTRAINT MAX_NAME_LEN_RANGE_CHECK
        CHECK (MAX_NAME_LEN >= 31) ,

    MIN_NAME_LEN CARDINAL
        CONSTRAINT MIN_NAME_LEN_RANGE_CHECK
        CHECK (MIN_NAME_LEN > 0) ,
        CONSTRAINT CONSISTENT_LEN_RANGE_CHECK
        CHECK (MAX_NAME_LEN >= MIN_NAME_LEN) ,

    IRD_TABLE_TYPE CHAR_DATA NOT NULL
        CHECK (IRD_TABLE_TYPE IN 'BASE TABLE', 'VIEW') ,

    VERSIONABLE BOOLEAN NOT NULL ,

    UNIQUE (IRD_TABLE_SCHEMA_OBJ_KEY, IRD_TABLE_SCHEMA_WS_KEY, IRD_TABLE_NAME),
    CONSTRAINT IRD_VIEW_IS_A_VIEW
        CHECK
        ( NOT EXISTS
            (   SELECT (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY) FROM IRD_TABLE
                        WHERE IRD_TABLE_TYPE = 'VIEW'
                EXCEPT
                SELECT (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY) FROM IRD_VIEW
            )
        )
    )
```

Description

1 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD Table table.

2 IRD_TABLE_NAME is the SQL name of the IRD table.

3 IRD_TABLE_SCHEMA_OBJ_KEY and IRD_TABLE_SCHEMA_WS_KEY together form the primary key of the IRD Schema that contains this table definition.

4 If not NULL, SUPERTABLE_OBJ_KEY and SUPERTABLE_WS_KEY together form the primary key of the supertable (see 5.5.1) of the table defined by this row. The table defined by the current row is known as a subtable of the supertable. This table inherits all the columns and constraints of its supertable, including any columns the supertable itself has inherited.

In any situation where the sequence of columns in a table has significance, the inherited columns appear before the columns of the table itself. This rule is applied recursively. Thus the columns of the highest supertable appear first.

5 MAX_NAME_LEN is the maximum length of an IRDS name for an IRD Table. No row in the IRD Table shall have an IRDS name longer than this value.

6 MIN_NAME_LEN is the minimum length of an IRDS name for an IRD Table. No row in the IRD Table shall have an IRDS name shorter than this value.

7 The values of IRD_TABLE_TYPE have the following meanings:

BASE TABLE: The IRD table being described is a persistent base table.

VIEW: The IRD table being described is an IRD View.

8 VERSIONABLE If true, indicates that rows in the IRD table described by this row can reference either a versionable or non-versionable working set. If false, indicates that rows in the IRD table described by this row can reference only a non-versionable working set.

6.1.4.14 Table IRD View

Function

The IRD_VIEW table contains one row for each IRD View in the IRDS environment, containing the view definition.

Definition

```
CREATE TABLE IRD_VIEW
    (
    IRD_VIEW_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_VIEW_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_VIEW_OBJ_KEY, IRD_VIEW_WS_KEY) ,
IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_TABLE_WS_KEY IRDS_KEY NOT NULL  ,

UNIQUE (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY) ,

VIEW_DEFINITION CHAR_DATA NOT NULL ,

CHECK_OPTION BOOLEAN NOT NULL ,

UPDATABLE BOOLEAN NOT NULL ,

CONSTRAINT IRD_VIEW_HAS_CORRECT_TABLE_TYPE
    CHECK
        ( ( IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY ) IN
            (SELECT ( IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY ) FROM IRD_TABLE
                        WHERE IRD_TABLE_TYPE = 'VIEW' )
        ) ,

CONSTRAINT UPDATABLE_AND_CHECK_OPTION_ARE_COMPATIBLE
    CHECK ( (UPDATABLE, CHECK_OPTION) <> ('FALSE', 'TRUE') )
)
```

Description

1 IRD_VIEW_OBJ_KEY and IRD_VIEW_WS_KEY together form the primary key of the IRD View table.

2 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD Table that is this IRD View.

3 VIEW_DEFINITION is a character representation of the <query expression> in the associated <view definition>. The character string representing the <query expression> shall be parsed according to the rules specified in ISO/IEC 9075, in the context of the schema in which the <view definition> is contained.

4 The possible values of UPDATABLE have the following meanings:

’TRUE’, when CHECK_OPTION has the value ’FALSE’, means the IRD view is updatable,

’TRUE’, when CHECK_OPTION also has the value ’TRUE’, means the IRD View was defined with the check option as defined in ISO/IEC 9075.

’FALSE’ means the view cannot be updated; CHECK_OPTION shall also have the value ’FALSE’.

Whether an IRD view is updatable is determined by parsing VIEW_DEFINITION in the context of the IRD table definition.

5 Where the <query expression> includes a join, the columns on which the join is made shall not include the object key portion of the primary key of any table unless they also include the working set key portion of the primary key of that same table.

6.1.4.15 Table IRD Column

Function

Each row in this IRD Definition Table specifies an IRD Column in an IRD Table. Each IRD Column has either a Data Type or an IRD Domain. A column may have a default value.

If a row is inserted into an IRD table, and values are not supplied for one or more IRD columns, then, if any such IRD column has a default value, such value shall be used as if it had been supplied in the insert statement.

Definition

```
CREATE TABLE IRD_COLUMN
    (
    IRD_COLUMN_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_COLUMN_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY) ,

    IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_COLUMN_CONTAINED_IN_IRD_TABLE
        FOREIGN KEY (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY)
            REFERENCES IRD_TABLE
            ON DELETE CASCADE ,

    IRD_COLUMN_NAME SQL_NAME NOT NULL ,

    IRD_COLUMN_POSITION CARDINAL NOT NULL ,

    MANDATORY BOOLEAN NOT NULL ,

    SYSTEM_MAINTAINED BOOLEAN NOT NULL ,

    IRD_DOMAIN_OBJ_KEY IRDS_KEY ,
    IRD_DOMAIN_WS_KEY IRDS_KEY ,
    CONSTRAINT IRD_COLUMN_DEFINED_ON_IRD_DOMAIN
        FOREIGN KEY (IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY)
            REFERENCES IRD_DOMAIN ,

    IRD_COLUMN_DEFAULT CHAR_DATA ,

    UNIQUE (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY, IRD_COLUMN_NAME) ,

    UNIQUE (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY, IRD_COLUMN_POSITION) ,

    CONSTRAINT IRD_COLUMN_HAS_DOMAIN_OR_DATA_TYPE
        CHECK
            ( IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY) IS NOT NULL
            AND
            ( IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY) NOT IN
                (SELECT  IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY
                    FROM IRD_DATA_TYPE_DESCRIPTOR )
            OR
            ( IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY) IS NULL
            AND
            ( IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY) IN
                (SELECT  IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY
                    FROM IRD_DATA_TYPE_DESCRIPTOR )
    )
```

Description

1 Case:

a If an IRD column belongs to an IRD base table, then the <table definition> and the <column definition> are associated with that column.

b If an IRD column belongs to an IRD view, then the <view definition> and the corresponding column description of the IRD table specified by the <query expression> are associated with that column.

2 IRD_COLUMN_OBJ_KEY and IRD_COLUMN_WS_KEY together form the primary key of the IRD Column table.

3 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD table containing the IRD column being described.

4 IRD_COLUMN_NAME is the SQL name of the IRD column being described.

5 IRD_COLUMN_POSITION is the ordinal position in the IRD Table of the column being described.

6 If MANDATORY is 'TRUE', a null value for this IRD Column is not allowed if the subject has a content status in which constraints are enforced. If MANDATORY is 'FALSE', no such constraint applies. The default value for this column, if none is specified, is 'FALSE'.

7 If SYSTEM_MAINTAINED is 'TRUE', the values of this IRD Column are maintained by the IRDS; if SYSTEM_MAINTAINED is 'FALSE', they are maintained by the IRDS user. The default value for this column, if none is specified, is 'FALSE'.

8 IRD_DOMAIN_OBJ_KEY and IRD_DOMAIN_WS_KEY are null if the IRD column being described is not defined on an IRD domain. Otherwise, the values of IRD_DOMAIN_OBJ_KEY and IRD_DOMAIN_WS_KEY together form the primary key of the IRD domain on which the IRD column being described is based.

9 The value of IRD_COLUMN_DEFAULT is null if the IRD column being described has no default value or if its default value comes only from an IRD domain. Otherwise, IRD_COLUMN_DEFAULT contains a literal that obeys the rules of ISO/IEC 9075 subclause <literal> that specifies the default value for the IRD column.

6.1.4.16 Table IRD View Table Usage

Function

The IRD_VIEW_TABLE_USAGE table has one row for each table referenced by a <view definition>.

Definition

```
CREATE TABLE IRD_VIEW_TABLE_USAGE
    (
    IRD_VIEW_TABLE_USAGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_VIEW_TABLE_USAGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_VIEW_TABLE_USAGE_OBJ_KEY, IRD_VIEW_TABLE_USAGE_WS_KEY) ,

    IRD_VIEW_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_VIEW_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_VIEW_TABLE_USAGE_USED_BY_IRD_VIEW
        FOREIGN KEY (IRD_VIEW_OBJ_KEY, IRD_VIEW_WS_KEY)
            REFERENCES IRD_VIEW
            ON DELETE CASCADE ,

    IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_VIEW_TABLE_USAGE_USES_IRD_TABLE
        FOREIGN KEY (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY)
            REFERENCES IRD_TABLE
            ON DELETE CASCADE ,
    CONSTRAINT REFERENCING_AND_REFERENCED_TABLES_ARE_DISTINCT
        CHECK
            ( ( IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY) NOT IN
                (SELECT IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY
                    FROM IRD_VIEW AS X )
                    WHERE ( IRD_VIEW_OBJ_KEY = X.IRD_VIEW_OBJ_KEY AND
                            IRD_VIEW_WS_KEY = X.IRD_VIEW_WS_KEY )
            )
    )
```

Description

1 IRD_VIEW_TABLE_USAGE_OBJ_KEY and IRD_VIEW_TABLE_USAGE_WS_KEY together form the primary key of the IRD View Table Usage table.

2 IRD_VIEW_OBJ_KEY and IRD_VIEW_WS_KEY together form the primary key of the IRD View that uses the IRD Table.

3 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD Table used by the IRD View.

6.1.4.17 Table IRD View Column Usage

Function

The IRD_VIEW_COLUMN_USAGE table has one row for each column referenced by a view.

Definition

```
CREATE TABLE IRD_VIEW_COLUMN_USAGE
    (
    IRD_VIEW_COLUMN_USAGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_VIEW_COLUMN_USAGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_VIEW_COLUMN_USAGE_OBJ_KEY, IRD_VIEW_COLUMN_USAGE_WS_KEY) ,

    IRD_VIEW_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_VIEW_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_VIEW_COLUMN_USAGE_USED_BY_IRD_VIEW
        FOREIGN KEY (IRD_VIEW_OBJ_KEY, IRD_VIEW_WS_KEY)
            REFERENCES IRD_VIEW
            ON DELETE CASCADE ,

    IRD_COLUMN_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_COLUMN_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_VIEW_COLUMN_USAGE_USES_IRD_COLUMN
        FOREIGN KEY (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY)
            REFERENCES IRD_COLUMN
            ON DELETE CASCADE
    )
```

Description

1 IRD_VIEW_COLUMN_USAGE_OBJ_KEY and IRD_VIEW_COLUMN_USAGE_WS_KEY together form the primary key of the IRD View Column Usage table.

2 IRD_VIEW_OBJ_KEY and IRD_VIEW_WS_KEY together form the primary key of the IRD View that uses the IRD Column.

3 IRD_COLUMN_OBJ_KEY and IRD_COLUMN_WS_KEY together form the primary key of the IRD Column used by the IRD View.

6.1.4.18 Table IRD Table Constraint

Function

Each row in this table specifies a constraint on an IRD table. Every IRD table constraint is a Uniqueness Constraint, a Referential (Foreign key) Constraint, or Check Constraint, depending on the value in the Definition Column IRD_CONSTRAINT_TYPE.

Definition

```
CREATE TABLE IRD_TABLE_CONSTRAINT
(
IRD_TABLE_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_TABLE_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
PRIMARY KEY (IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY) ,

IRD_CONSTRAINT_NAME SQL_NAME NOT NULL ,

IRD_CONSTRAINT_TYPE CHAR_DATA NOT NULL
    CHECK (IRD_CONSTRAINT_TYPE IN ('UNIQUE', 'FOREIGN KEY', 'CHECK', 'PRIMARY KEY') ) ,

UNIQUE (IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY,
            IRD_CONSTRAINT_TYPE) ,

IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_TABLE_WS_KEY IRDS_KEY NOT NULL ,

CONSTRAINT IRD_TABLE_CONSTRAINT_CONSTRAINS_IRD_TABLE
    FOREIGN KEY (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY)
        REFERENCES IRD_TABLE
        ON DELETE CASCADE ,

CONSTRAINT IRD_TABLE_CONSTRAINT_CONSTRAINS_IRD_BASE_TABLE
    CHECK ( ( IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY ) IN
                    (SELECT ( IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY ) FROM IRD_TABLE
                        WHERE IRD_TABLE_TYPE <> 'VIEW') ) ,

CONSTRAINT IRD_TABLE_CONSTRAINT_IS_COMPLETE
    CHECK ( ( SELECT COUNT(*)
                FROM IRD_TABLE_CONSTRAINT AS X
                WHERE (IRD_TABLE_CONSTRAINT_OBJ_KEY = X.IRD_TABLE_CONSTRAINT_OBJ_KEY
                    AND IRD_TABLE_CONSTRAINT_WS_KEY = X.IRD_TABLE_CONSTRAINT_WS_KEY
                    AND X.IRD_CONSTRAINT_TYPE IN ('UNIQUE', 'PRIMARY KEY')
                    )
            + SELECT COUNT(*) FROM IRD_REF_CONSTRAINT AS Y
                WHERE (IRD_TABLE_CONSTRAINT_OBJ_KEY = Y.IRD_TABLE_CONSTRAINT_OBJ_KEY
                    AND IRD_TABLE_CONSTRAINT_WS_KEY = Y.IRD_TABLE_CONSTRAINT_WS_KEY
                    )
            + SELECT COUNT(*) FROM IRD_CHECK_CONSTRAINT AS Z
                WHERE (IRD_TABLE_CONSTRAINT_OBJ_KEY = Z.IRD_TABLE_CONSTRAINT_OBJ_KEY
                    AND IRD_TABLE_CONSTRAINT_WS_KEY = Z.IRD_TABLE_CONSTRAINT_WS_KEY
                    )
            ) = 1
    ),
    CONSTRAINT IRD_TABLE_UNIQUE_PRIMARY_KEY
    CHECK
        (UNIQUE
            (SELECT IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY FROM IRD_TABLE_CONSTRAINT
                WHERE IRD_CONSTRAINT_TYPE = 'PRIMARY KEY' )
        )
    )
```

Description

1 IRD_TABLE_CONSTRAINT_OBJ_KEY and IRD_TABLE_CONSTRAINT_WS_KEY together form the primary key of the IRD Table Constraint table.

2 IRD_CONSTRAINT_NAME is the SQL name of the IRD Table Constraint.

3 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD table in which the IRD table constraint is contained.

4 IRD_CONSTRAINT_TYPE

Each row in this IRD table for which IRD_CONSTRAINT_TYPE is 'UNIQUE', 'PRIMARY KEY' or 'FOREIGN KEY' is referenced by one or more IRD Key Column Usages each of which references an IRD column belonging to the same IRD Table as the IRD Table Constraint.

Each row in this IRD table for which IRD_CONSTRAINT_TYPE is 'UNIQUE', or 'PRIMARY KEY' means that no two rows of that IRD Table may have duplicate (indistinguishable) values in these Columns.

Each row in this table for which IRD_CONSTRAINT_TYPE is 'FOREIGN KEY' is referenced by a row of IRD_REF_CONSTRAINT.

Each row in this table for which IRD_CONSTRAINT_TYPE is 'CHECK' is referenced by a row of IRD Check Constraint. The check clause in that row shall not be false for any row in the IRD table.

6.1.4.19 Table IRD Key Column Usage

Function

The IRD_KEY_COLUMN_USAGE table has one or more rows for each row in the IRD_TABLE_CONSTRAINTs table that has a CONSTRAINT_TYPE of 'Unique', 'Primary key' or 'Foreign key'. The rows list the columns that constitute each unique or primary key constraint, and the referencing columns in each foreign key constraint.

Definition

```
CREATE TABLE IRD_KEY_COLUMN_USAGE
    (
    IRD_KEY_COLUMN_USAGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_KEY_COLUMN_USAGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_KEY_COLUMN_USAGE_OBJ_KEY, IRD_KEY_COLUMN_USAGE_WS_KEY) ,

    IRD_TABLE_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRDS_KEY_COLUMN_USAGE_USED_BY_IRD_TABLE_CONSTRAINT
        FOREIGN KEY (IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY)
            REFERENCES IRD_TABLE_CONSTRAINT
            ON DELETE CASCADE ,
IRD_COLUMN_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_COLUMN_WS_KEY IRDS_KEY NOT NULL ,
CONSTRAINT IRDS_KEY_COLUMN_USAGE_USES_IRD_COLUMN
    FOREIGN KEY (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY)
        REFERENCES IRD_COLUMN
        ON DELETE CASCADE ,

IRD_COLUMN_POSITION CARDINAL NOT NULL ,

UNIQUE ( IRD_OBJECT_KEY, IRD_WORKING_SET_KEY, IRD_COLUMN_KEY) ,

UNIQUE (IRD_OBJECT_KEY, IRD_WORKING_SET_KEY, IRD_COLUMN_POSITION),

CONSTRAINT IRD_TABLE_CONSTRAINT_TYPE_RANGE_CHECK
    CHECK
        ( ( IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY ) IN
            (SELECT IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY
                FROM IRD_TABLE_CONSTRAINT
                WHERE IRD_CONSTRAINT_TYPE IN
                    ('UNIQUE',   'PRIMARY KEY', 'FOREIGN KEY')
            )
        ) ,

CONSTRAINT IRD_COLUMN_CONTAINED_IN_CONSTRAINED_IRD_TABLE
    CHECK ( NOT EXISTS
        (SELECT COLUMN_TABLE_OBJ_KEY, COLUMN_TABLE_WS_KEY,
            CONSTRAINT_TABLE_OBJ_KEY, CONSTRAINT_TABLE_WS_KEY
        FROM ( ( SELECT     IRD_TABLE_OBJ_KEY AS COLUMN_TABLE_OBJ_KEY,
                            IRD_TABLE_WS_KEY AS COLUMN_TABLE_WS_KEY,
                            IRD_TABLE_CONSTRAINT_OBJ_KEY,
                            IRD_TABLE_CONSTRAINT_WS_KEY
                    FROM ( IRD_COLUMN JOIN IRDS_KEY_COLUMN_USAGE
                                    USING (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY)
                            )
                )
                JOIN
                (SELECT     IRD_TABLE_OBJ_KEY AS CONSTRAINT_TABLE_OBJ_KEY,
                            IRD_TABLE_WS_KEY AS CONSTRAINT_TABLE_WS_KEY ,
                            IRD_TABLE_CONSTRAINT_OBJ_KEY,
                            IRD_TABLE_CONSTRAINT_WS_KEY
                    FROM IRD_TABLE_CONSTRAINT
                )
                USING (IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY)
            )
        WHERE ( COLUMN_TABLE_OBJ_KEY <> CONSTRAINT_TABLE_OBJ_KEY ) OR
                ( COLUMN_TABLE_WS_KEY <> CONSTRAINT_TABLE_WS_KEY )
        )
)
```

Description

1 IRD_KEY_COLUMN_USAGE_OBJ_KEY and IRD_KEY_COLUMN_USAGE_WS_KEY together form the primary key of the IRD Key Column Usage table.

2 IRD_TABLE_CONSTRAINT_OBJ_KEY and IRD_TABLE_CONSTRAINT_WS_KEY together form the primary key of the IRD table constraint to which the IRD key column usage belongs.

3 IRD_COLUMN_OBJ_KEY and IRD_COLUMN_WS_KEY together form the primary key of the IRD column that is used in the IRD unique constraint or foreign key constraint.

4 IRD_COLUMN_POSITION is the ordinal position of the column in the key.

5 The second check ensures that all columns belonging to any constraint belong to the same table as that constraint.

6.1.4.20 Table IRD Referential Constraint

Function

Each row in this table specifies a referential constraint on an IRD table.

Definition

```
CREATE TABLE IRD_REF_CONSTRAINT
    (
    IRD_REF_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_REF_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_REF_CONSTRAINT_OBJ_KEY, IRD_REF_CONSTRAINT_WS_KEY) ,

    IRD_CONSTRAINT_NAME SQL_NAME NOT NULL ,

    IRD_TABLE_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,

    UNIQUE_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    UNIQUE_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,

    MATCH_OPTION CHAR_DATA
        CONSTRAINT MATCH_OPTION_RANGE_CHECK
            CHECK (MATCH_OPTION IN ('NONE', 'PARTIAL', 'FULL')),

    UPDATE_RULE CHAR_DATA
        CONSTRAINT UPDATE_RULE_RANGE_CHECK
            CHECK
                (UPDATE_RULE IN ('CASCADE', 'SET NULL', 'SET DEFAULT')) ,

    DELETE_RULE CHAR_DATA
        CONSTRAINT DELETE_RULE_RANGE_CHECK
            CHECK
                (DELETE_RULE IN ('CASCADE', 'SET NULL', 'SET DEFAULT')) ,

    CONSTRAINT IRD_REF_CONSTRAINT_IS_IRD_TABLE_CONSTRAINT
        CHECK
            ( ( IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY ) IN
                    (SELECT ( IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY )
                        FROM IRD_TABLE_CONSTRAINT
                        WHERE IRD_CONSTRAINT_TYPE = 'FOREIGN KEY' )
            ) ,

    CONSTRAINT IRD_REFED_COLUMNS_ARE_PRIMARY_KEY
        CHECK
            ( ( UNIQUE_CONSTRAINT_OBJ_KEY, UNIQUE_CONSTRAINT_WS_KEY ) IN
                    (SELECT ( IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY )
                        FROM IRD_TABLE_CONSTRAINT
                        WHERE IRD_CONSTRAINT_TYPE = 'PRIMARY KEY' )
            )
    )
```

Description

1 IRD_REF_CONSTRAINT_OBJ_KEY and IRD_REF_CONSTRAINT_WS_KEY together form the primary key of the IRD Ref Constraint table.

2 IRD_CONSTRAINT_NAME is the SQL_name of the IRD_table constraint.

3 IRD_TABLE_CONSTRAINT_OBJ_KEY and IRD_TABLE_CONSTRAINT_WS_KEY together form the primary key of the IRD table constraint that is the IRD referential constraint.

4 UNIQUE_CONSTRAINT_OBJ_KEY and UNIQUE_CONSTRAINT_WS_KEY together form the primary key of the unique constraint that constrains the referenced columns of the referenced table.

Each row in this table references some row in IRD Table Constraint, for which IRD_CONSTRAINT_TYPE is 'FOREIGN KEY' (whose set of key columns, defined by the corresponding columns of IRD Key Column Usage, are known as the referencing columns).

Each row in this table references some row in IRD Table constraint, for which IRD_CONSTRAINT_TYPE is 'PRIMARY KEY' (whose set of key columns, defined by the corresponding columns of IRD Key Column Usage, are known as the referenced columns).

The constraint means that, for each row in the referencing table the values of the referencing columns shall not be different from the values of the referenced columns in every row of the referenced table.

5 If IRD_CONSTRAINT_TYPE is 'FOREIGN KEY', then

a) If UPDATE_RULE is null, no Update Rule applies.

b) If DELETE_RULE is null, no Delete Rule applies.

6 The values of MATCH_OPTION have the following meanings for a referential constraint:

NONE: No <match type> was specified.

PARTIAL: A <match type> of PARTIAL was specified.

FULL: A <match type> of FULL was specified.

7 The values of UPDATE_RULE have the following meanings for a referential constraint that has an update rule:

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.

8 The values of DELETE_RULE have the following meanings for a referential constraint that has a <delete rule>:

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.

6.1.4.21 Table IRD Check Constraint

Function

The IRD_CHECK_CONSTRAINTS table has one row for each <check constraint definition> contained in a <domain constraint> or <table constraint>, or <assertion check>.

Definition

```
CREATE TABLE IRD_CHECK_CONSTRAINT
    (
    IRD_CHECK_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CHECK_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_CHECK_CONSTRAINT_OBJ_KEY, IRD_CHECK_CONSTRAINT_WS_KEY) ,

    IRD_CHECK_CLAUSE CHAR_DATA ,

    IRD_TABLE_CONSTRAINT_OBJ_KEY IRDS_KEY ,
    IRD_TABLE_CONSTRAINT_WS_KEY IRDS_KEY ,

    CONSTRAINT IRD_CHECK_CONSTRAINT_IS_TABLE_CONSTRAINT
        FOREIGN KEY (IRD_TABLE_CONSTRAINT_OBJ_KEY, IRD_TABLE_CONSTRAINT_WS_KEY, 'CHECK')
            REFERENCES IRD_TABLE_CONSTRAINT (IRD_TABLE_CONSTRAINT_OBJ_KEY,
                            IRD_TABLE_CONSTRAINT_WS_KEY, IRD_CONSTRAINT_TYPE)
                ON DELETE CASCADE ,

    IRD_ASSERTION_OBJ_KEY IRDS_KEY ,
    IRD_ASSERTION_WS_KEY IRDS_KEY ,

    CONSTRAINT IRD_CHECK_CONSTRAINT_IS_ASSERTION
        FOREIGN KEY (IRD_ASSERTION_OBJ_KEY, IRD_ASSERTION_WS_KEY)
            REFERENCES IRD_ASSERTION
                ON DELETE CASCADE ,

    IRD_DOMAIN_OBJ_KEY IRDS_KEY ,
    IRD_DOMAIN_WS_KEY IRDS_KEY ,

    CONSTRAINT IRD_CHECK_CONSTRAINT_CONSTRAINS_DOMAIN
        FOREIGN KEY (IRD_DOMAIN_OBJ_KEY, IRD_DOMAIN_WS_KEY)
            REFERENCES IRD_DOMAIN
                ON DELETE CASCADE ,

    CONSTRAINT IRD_CHECK_CONSTRAINT_HAS_UNIQUE_TYPE
        CHECK
        ( SELECT COUNT(*) FROM IRD_DOMAIN AS X
            WHERE (IRD_DOMAIN_CONSTRAINT_OBJ_KEY = X.IRD_DOMAIN_CONSTRAINT_OBJ_KEY
                AND IRD_DOMAIN_CONSTRAINT_WS_KEY = X.IRD_DOMAIN_CONSTRAINT_WS_KEY
                    )
        + SELECT COUNT(*) FROM IRD_ASSERTION AS Y
            WHERE (IRD_ASSERTION_OBJ_KEY = Y.IRD_ASSERTION_OBJ_KEY
                AND IRD_ASSERTION_WS_KEY = Y.IRD_ASSERTION_WS_KEY
                    )
        + SELECT COUNT(*) FROM IRD_TABLE_CONSTRAINT AS Z
            WHERE (IRD_TABLE_CONSTRAINT_OBJ_KEY = Z.IRD_TABLE_CONSTRAINT_OBJ_KEY
                AND IRD_TABLE_CONSTRAINT_WS_KEY = Z.IRD_TABLE_CONSTRAINT_WS_KEY
                    )
        = 1 )
    )
```

Description

1 IRD_CHECK_CONSTRAINT_OBJ_KEY and IRD_CHECK_CONSTRAINT_WS_KEY together form the primary key of the IRD Check Constraint table.

2 The value of IRD_CHECK_CLAUSE is a character representation of the <search condition> in the associated <check constraint definition> or <assertion definition>. It is to be interpreted according to the definitions of those terms in ISO/IEC 9075.

Case:

If the IRD Check Constraint is referenced by an IRD Table Constraint, then the IRD_CHECK_CLAUSE may contain outer references to columns of the IRD Table to which the IRD Table Constraint belongs.

If the IRD Check Constraint is referenced by an IRD Domain, then the IRD_CHECK_CLAUSE may contain the keyword VALUE to refer to a value that is to be checked as being in that IRD Domain.

3 IRD_TABLE_CONSTRAINT_OBJ_KEY and IRD_TABLE_CONSTRAINT_WS_KEY, together with the constant 'CHECK', form a foreign key referring to the table IRD Table Constraint. The constant is required because IRD Table Constraint has other subtypes, and this ensures that the referenced constraint is the correct subtype.

4 IRD_ASSERTION_OBJ_KEY and IRD_ASSERTION_WS_KEY together form a foreign key referring to the table IRD Assertion. Every row in the IRD Assertion table has a corresponding row in this table.

5 IRD_DOMAIN_OBJ_KEY and IRD_DOMAIN_WS_KEY together form a foreign key referring to the table IRD Domain, if the row in this table represents a constraint on a domain.

6 The <check constraint definition> ensures that every check constraint is associated with one and only one of a domain or a table or an assertion.

7 Where the <search condition> includes a join, the columns on which the join is made shall not include the object key portion of the primary key of any table unless they also include the working set key portion of the primary key of that same table.

6.1.4.22 Table IRD Check Table Usage

Function

The IRD_CHECK_TABLE_USAGE table has one row for each table referenced by a <check constraint definition> or <assertion check>.

Definition

```
CREATE TABLE IRD_CHECK_TABLE_USAGE
    (
    IRD_CHECK_TABLE_USAGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CHECK_TABLE_USAGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_CHECK_TABLE_USAGE_OBJ_KEY, IRD_CHECK_TABLE_USAGE_WS_KEY) ,

    IRD_CHECK_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CHECK_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_CHECK_TABLE_USAGE_USED_BY_IRD_CHECK_CONSTRAINT
        FOREIGN KEY (IRD_CHECK_CONSTRAINT_OBJ_KEY, IRD_CHECK_CONSTRAINT_WS_KEY)
            REFERENCES IRD_CHECK_CONSTRAINT
            ON DELETE CASCADE ,

    IRD_TABLE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_TABLE_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_CHECK_TABLE_USAGE_USES_IRD_TABLE
        FOREIGN KEY (IRD_TABLE_OBJ_KEY, IRD_TABLE_WS_KEY)
            REFERENCES IRD_TABLE
            ON DELETE CASCADE
    )
```

Description

1 IRD_CHECK_TABLE_USAGE_OBJ_KEY and IRD_CHECK_TABLE_USAGE_WS_KEY together form the primary key of the IRD Check Table Usage table.

2 IRD_CHECK_CONSTRAINT_OBJ_KEY and IRD_CHECK_CONSTRAINT_WS_KEY together form the primary key of the IRD Check Constraint that uses the IRD Table.

3 IRD_TABLE_OBJ_KEY and IRD_TABLE_WS_KEY together form the primary key of the IRD Table used by the IRD Check Constraint.

6.1.4.23 Table IRD Check Column Usage

Function

The CHECK_COLUMN_USAGE table has one row for each column referenced by a <check constraint definition> or <assertion check>.

Definition

```
CREATE TABLE IRD_CHECK_COLUMN_USAGE
    (
    IRD_CHECK_COLUMN_USAGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CHECK_COLUMN_USAGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_CHECK_COLUMN_USAGE_OBJ_KEY, IRD_CHECK_COLUMN_USAGE_WS_KEY) ,

    IRD_CHECK_CONSTRAINT_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CHECK_CONSTRAINT_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_CHECK_COLUMN_USAGE_USED_BY_IRD_CHECK_CONSTRAINT
        FOREIGN KEY (IRD_CHECK_CONSTRAINT_OBJ_KEY, IRD_CHECK_CONSTRAINT_WS_KEY)
            REFERENCES IRD_CHECK_CONSTRAINT
            ON DELETE CASCADE ,

    IRD_COLUMN_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_COLUMN_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_CHECK_COLUMN_USAGE_USES_IRD_COLUMN
        FOREIGN KEY (IRD_COLUMN_OBJ_KEY, IRD_COLUMN_WS_KEY)
            REFERENCES IRD_COLUMN
            ON DELETE CASCADE
    )
```

Description

1 IRD_CHECK_COLUMN_USAGE_OBJ_KEY and IRD_CHECK_COLUMN_USAGE_WS_KEY together form the primary key of the IRD Check Column Usage table.

2 IRD_CHECK_CONSTRAINT_OBJ_KEY and IRD_CHECK_CONSTRAINT_WS_KEY together form the primary key of the IRD Check Constraint that uses the IRD Column.

3 IRD_COLUMN_OBJ_KEY and IRD_COLUMN_WS_KEY together form the primary key of the IRD Column used by the IRD Check Constraint.

6.1.4.24 Table IRD Assertion

Function

Each row in this table represents an IRD Assertion, which shall never be false (unless it is temporarily deferred).

Definition

```
CREATE TABLE IRD_ASSERTION
    (
    IRD_ASSERTION_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_ASSERTION_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_ASSERTION_OBJ_KEY, IRD_ASSERTION_WS_KEY) ,
IRD_CONSTRAINT_SCHEMA_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_CONSTRAINT_SCHEMA_WS_KEY IRDS_KEY NOT NULL ,
CONSTRAINT IRD_ASSERTION_CONTAINED_IN_IRD_SCHEMA
    FOREIGN KEY (IRD_CONSTRAINT_SCHEMA_OBJ_KEY, IRD_CONSTRAINT_SCHEMA_WS_KEY)
        REFERENCES IRD_SCHEMA
        ON DELETE CASCADE ,

IRD_ASSERTION_CHECK_OBJ_KEY IRDS_KEY NOT NULL ,
IRD_ASSERTION_CHECK_WS_KEY IRDS_KEY NOT NULL ,
CONSTRAINT IRD_ASSERTION_IS_CHECK_CONSTRAINT
    FOREIGN KEY (IRD_ASSERTION_CHECK_OBJ_KEY, IRD_ASSERTION_CHECK_WS_KEY)
        REFERENCES IRD_CHECK_CONSTRAINT
        ON DELETE CASCADE ,

IRD_CONSTRAINT_NAME SQL_NAME ,

UNIQUE (IRD_CONSTRAINT_SCHEMA_OBJ_KEY, IRD_CONSTRAINT_SCHEMA_WS_KEY,
         IRD_CONSTRAINT_NAME)
)
```

Description

1 IRD_ASSERTION_OBJ_KEY and IRD_ASSERTION_WS_KEY together form the primary key of the IRD Assertion table.

2 IRD_CONSTRAINT_SCHEMA_OBJ_KEY and IRD_CONSTRAINT_SCHEMA_WS_KEY together form the primary key of the IRD schema that contains the assertion.

3 IRD_ASSERTION_CHECK_OBJ_KEY and IRD_ASSERTION_CHECK_WS_KEY together form the primary key of the IRD check constraint that expresses the assertion.

4 IRD_CONSTRAINT_NAME is the name of the IRD assertion.

NOTE - The remainder of the assertion is a check clause that is stored in the referenced row in IRD Check Constraint.

6.1.4.25 Table IRD Module

Function

This Common table contains a row for each module of IRD Definition or IRD content controlled in an IRD Database by the data modelling facility defined in this standard.

The IRDS name of the module is defined as follows:

a) for an International Standard, 'ISnnnnnn-p', where nnnnnn is the number of the international standard. For a multi-part or multi-module standard, p is the part or module number. For a single-part standard, the last two characters ("-p") are omitted.

b) for a National Standard, begins with 'NS', followed by the two-character code identifying the relevant ISO member body in accordance with ISO 3166, the remainder being at the discretion of that national standards body.

c) for a vendor extension, begins with 'V', the remainder being at the discretion of the vendor.

d) for an installation extension, begins with 'IN', the remainder being at the discretion of the installation.

Definition

```
CREATE TABLE IRD_MODULE
    (
    IRD_MODULE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_MODULE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_MODULE_OBJ_KEY, IRD_MODULE_WS_KEY) ,

    STANDARD_MODE BOOLEAN NOT NULL
    )
```

Description

1 IRD_MODULE_OBJ_KEY and IRD_MODULE_WS_KEY together form the primary key of the IRD Module table.

2 If STANDARD_MODE is 'TRUE', it specifies that the implementation of this Module, as installed, is to provide only standard IRDS services, without any vendor extensions enabled. If STANDARD_MODE is 'FALSE', any vendor extensions are available.

6.1.4.26 Table IRD Content Status

Function

Each row of this Common table defines a status in which objects or definition objects can exist. Each IRD working set has one and only one IRD content status.

All data with a specific content status has the same IRD content status class (uncontrolled, controlled or archived).

Definition

```
CREATE TABLE IRD_CONTENT_STATUS
    (
    IRD_CONTENT_STATUS_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_CONTENT_STATUS_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_CONTENT_STATUS_OBJ_KEY, IRD_CONTENT_STATUS_WS_KEY) ,

    CONSTRAINT_ENFORCED BOOLEAN NOT NULL ,

    IRD_CONTENT_STATUS_CLASS CHAR_DATA
        CONSTRAINT IRD_CONTENT_STATUS_CLASS_RANGE_CHECK
        CHECK  (IRD_CONTENT_STATUS_CLASS IN ('UNCONTROLLED',
                                            'CONTROLLED', 'ARCHIVED')
                  )
    )
```

Description

1 IRD_CONTENT_STATUS_OBJ_KEY and IRD_CONTENT_STATUS_WS_KEY together form the primary key of the IRD Content Status table.

2 If CONSTRAINT_ENFORCED is 'TRUE', all IRD constraints shall be enforced for data in any working set associated with this IRD content status.

3 IRD_CONTENT_STATUS_CLASS - The status class has three possible values as follows:

'UNCONTROLLED', 'CONTROLLED', 'ARCHIVED'

4 The value of CONSTRAINT_ENFORCED is independent of the value of IRD_CONTENT_STATUS_CLASS.

6.1.4.27 Table Installation Default

Function

This Common table contains exactly one row that contains installation-specified default values for various IRD columns of IRD tables.

Definition

```
CREATE TABLE INSTALLATION_DEFAULT
    (
    INSTALLATION_DEFAULT_OBJ_KEY IRDS_KEY NOT NULL ,
    INSTALLATION_DEFAULT_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (INSTALLATION_DEFAULT_OBJ_KEY, INSTALLATION_DEFAULT_WS_KEY) ,

    INST_NAME_MAX CARDINAL NOT NULL
        CONSTRAINT INST_NAME_MAX_RANGE_CHECK
        CHECK (INST_NAME_MAX >= 10) ,

    INST_NAME_MIN CARDINAL NOT NULL
        CONSTRAINT INST_NAME_MIN_RANGE_CHECK
        CHECK (INST_NAME_MIN > 0) ,

    CONSTRAINT INSTALLATION_DEFAULT_HAS_EXACTLY_ONE_ROW
        CHECK ( 1 = SELECT COUNT(*) FROM INSTALLATION_DEFAULT )

    )
```

Description

1 INSTALLATION_DEFAULT_OBJ_KEY and INSTALLATION_DEFAULT_WS_KEY together form the primary key of the (single row of the) Installation Default table.

2 INST_NAME_MAX - installation default for the maximum length of an IRDS name for a row in any IRD Table.

3 INST_NAME_MIN - installation default for the minimum length of an IRDS name for a row in any IRD Table. No IRDS name shall be shorter than the length specified here.

6.1.4.28 Table IRD Working Set Privilege

Function

Each row of this Common table represents a permission granted to a specific IRDS User to access a specific IRD working set.

Definition

```
CREATE TABLE IRD_WORKING_SET_PRIVILEGE
    (
    IRD_WORKING_SET_PRIVILEGE_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_WORKING_SET_PRIVILEGE_WS_KEY IRDS_KEY NOT NULL ,
    PRIMARY KEY (IRD_WORKING_SET_PRIVILEGE_OBJ_KEY, IRD_WORKING_SET_PRIVILEGE_WS_KEY) ,

    IRD_GRANTOR_OBJ_KEY IRDS_KEY NOT NULL ,
    IRD_GRANTOR_WS_KEY IRDS_KEY NOT NULL ,
        CONSTRAINT IRD_WORKING_SET_PRIVILEGE_GRANTED_BY_IRDS_USER
        FOREIGN KEY (IRD_GRANTOR_OBJ_KEY, IRD_GRANTOR_WS_KEY)
            REFERENCES IRDS_USER ,
IRD_GRANTEE_OBJ_KEY IRDS_KEY NOT NULL
IRD_GRANTEE_WS_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT IRD_WORKING_SET_PRIVILEGE_GRANTED_TO_IRDS_USER
    FOREIGN KEY (IRD_GRANTEE_OBJ_KEY, IRD_GRANTEE_WS_KEY)
        REFERENCES IRDS_USER
        ON DELETE CASCADE ,

ON_WORKING_SET_KEY IRDS_KEY NOT NULL ,
    CONSTRAINT
        IRD_WORKING_SET_PRIVILEGE_GRANTED_ON_IRD_WORKING_SET
        REFERENCES IRD_WORKING_SET
        ON DELETE CASCADE ,

PRIVILEGE CHAR_DATA
    CONSTRAINT IRD_WORKING_SET_PRIVILEGE_RANGE_CHECK
    CHECK (PRIVILEGE IN ('SELECT', 'INSERT', 'DELETE', 'UPDATE') ) ,

GRANTABLE BOOLEAN NOT NULL ,

UNIQUE (IRD_GRANTEE_OBJ_KEY, IRD_GRANTEE_WS_KEY, ON_WORKING_SET_KEY,
                            IRD_GRANTOR_OBJ_KEY, IRD_GRANTOR_WS_KEY, PRIVILEGE ),
)
```

Description

1 IRD_WORKING_SET_PRIVILEGE_OBJ_KEY and IRD_WORKING_SET_PRIVILEGE_WS_KEY together form the primary key of the IRD Working Set Privilege table.

2 IRD_GRANTOR_OBJ_KEY and IRD_GRANTOR_WS_KEY together form the primary key of the user who granted access privileges on the working set being described.

3 IRD_GRANTEE_OBJ_KEY and IRD_GRANTEE_WS_KEY together form the primary key of the IRDS User, possibly 'PUBLIC', to whom access privileges have been granted.

4 ON_WORKING_SET_KEY is the primary key of the working set on which access privileges have been granted.

5 The values of PRIVILEGE have the following meanings:

SELECT - The user has select privileges on this working set.

DELETE - The user has delete privileges on this working set.

INSERT - The user has insert privileges on this working set.

UPDATE - The user has update privileges on this working set.

6 If GRANTABLE is 'TRUE', the grantee may grant the privilege to other IRD Users. Otherwise he may not.

6.1.5 IRD Definition Level Views

6.1.5.1 View All SQL Names

Function

This view contains one row for each SQL Name.

Definition

```
CREATE VIEW SQL_NAMES AS
    (
    (SELECT IRD_OBJECT_KEY AS NAME_KEY, IRD_WORKING_SET_KEY
              FROM IRD_OBJECT )
    JOIN
    (SELECT IRD_SCHEMA_KEY AS NAME_KEY, 'SCHEMA' AS NAME_TYPE,
                            IRD_SCHEMA_NAME AS NAME ,
                            CAST (NULL AS IRDS_KEY) AS OWNER_KEY
         FROM IRD_SCHEMA
    UNION
    SELECT IRD_DOMAIN_KEY, 'DOMAIN',  IRD_DOMAIN_NAME,
                            IRD_DOMAIN_SCHEMA_KEY
         FROM IRD_DOMAIN
    UNION
    SELECT IRD_TABLE_KEY, 'TABLE', IRD_TABLE_NAME ,
                            IRD_TABLE_SCHEMA_KEY
         FROM IRD_TABLE
    UNION
    SELECT IRD_COLUMN_KEY, 'COLUMN', IRD_COLUMN_NAME,
                            IRD_TABLE_KEY
         FROM IRD_COLUMN
    UNION
    SELECT IRD_CONSTRAINT_KEY, 'TABLE CONSTRAINT',
                            IRD_CONSTRAINT_NAME, IRD_TABLE_KEY
         FROM IRD_TABLE_CONSTRAINT
    UNION
    SELECT IRD_CONSTRAINT_KEY, 'ASSERTION',
                            IRD_CONSTRAINT_NAME, IRD_SCHEMA_KEY
         FROM IRD_ASSERTION
    UNION
    SELECT IRD_DOMAIN_KEY, 'DOMAIN CONSTRAINT',
                            IRD_CONSTRAINT_NAME, IRD_DOMAIN_SCHEMA_KEY
         FROM IRD_DOMAIN
    )
    USING (NAME_KEY)
    )
```

Description

1 NAME_KEY is the key that identifies the row in which the name is located

2 NAME_TYPE identifies the type of name.

3 NAME is the name itself.

4 OWNER_KEY is the IRDS_KEY of the object on which the named object is dependent, for example the owner of a table is a schema. A schema has no owner.

6.1.5.2 View IRD Object Version

Function

This view joins the tables IRD Object, IRD Object Version and IRD Working Set, so that the names of objects and working sets are available when accessing object versions.

Definition

```
CREATE VIEW IRD_OBJECT_VERSION_VIEW AS
    SELECT *
    FROM IRD_OBJECT
    JOIN
    IRD_OBJECT_VERSION USING (IRD_OBJECT_KEY)
    JOIN
    IRD_WORKING_SET USING (IRD_WORKING_SET_KEY)
```

Description

1 This view contains all the columns of the IRD Object, IRD Object Version and IRD Working Set tables.

6.1.5.3 View IRD Working Set

Function

This view contains one row for each row in the IRD Working Set table. It is provided to make the contents of that Internal table visible.

Definition

```
CREATE VIEW IRD_WORKING_SET_VIEW AS
    SELECT *
    FROM IRD_WORKING_SET
```

Description

1 This view contains all the columns of the IRD Working Set table.

6.1.5.4 View IRD Reference Path

Function

This view contains one row for each row in the IRD Reference Path table. It is provided to make the contents of that Internal table visible.

Definition

```
CREATE VIEW IRD_REFERENCE_PATH_VIEW AS
    SELECT *
    FROM IRD_REFERENCE_PATH
```

Description

1 This view contains all the columns of the IRD Reference Path table.

6.1.6 IRD Definition Level Change Control

The following rules shall apply at all times to the contents of any active IRD and the IRD Schema Group controlling by it:

a) Every reference from the IRD to the applicable IRD schema group shall be valid;

b) All data in the IRD shall be consistent with any constraints specified in the applicable IRD schema group.

6.1.7 IRD Definition Level Initial Contents

NOTE - The initial contents specified below are the minimum necessary to support the operation of an IRDS. They are not intended to imply any restriction on possible contents, other than those specified elsewhere in this standard.

In any newly-created IRD Definition, rows shall exist as follows.

NOTES

1 - Where a value is provided for an object key or working set key, this is done purely to make the references between rows clear. In an actual implementation the values used in such columns are implementation-dependent.

2 - Where a reference from one table to another exists in the form of a pair of keys, one to the object and one to the working set, for clarity they are shown below as a single column, with a pair of values separated by commas. The object key always comes first.

3 - Corresponding rows in the Object and Object Version tables shall also exist, conforming to the structure and constraints specified in 6.1.4.

4 - For readability, column headings in the tables within this clause are shown in the informal style introduced in clause 5. They can be related to the formal SQL names as described there; for example, Working Set Name refers to WORKING_SET_NAME.

a) Rows in the IRD Working Set table to reflect the existence of two non-versionable working sets named Environment and Common, each with a null working set version name.

Table 2 - IRD Working Set

Working Set Key	Working Set Name	Working Set Version Name	Based on Working Set Key	IRD Content Status Key	Versionable
00020001	Environment	null	null	00260002, 00020002	FALSE
00020002	Common	null	null	00260002, 00020002	FALSE

b) Rows in the IRDS User, IRD Object Version and IRD Object tables to reflect the existence of two IRDS Users named IRDS and IRDS Administrator; the rows in IRD Object Version shall refer to the Environment working set.

Table 5 - IRDS User

Object Key	Working Set Key	User Name	User Active	Default Working Set Name	Default Working Set Version Name
00050001	00020001	IRDS	TRUE	null	null
00050002	00020001	IRDS Administrator	TRUE	Environment	null

In each case the values of the columns IRDS_USER_MAY_CREATE_IRD and IRDS_USER_MAY_CREATE_WS shall be TRUE.

c) A row in the Implementation Limits table establishing values for the implementation limits specified in 6.1.4.6, with corresponding rows in the IRD Object Version and IRD Object tables; the row in IRD Object Version shall refer to the Environment working set.

Table 6 - Implementation Limits

Object Key	Working Set Key	Name Limit	Integer Limit	Text Limit	Variation Name Limit
00060001	00020001	impl1	impl2	impl3	impl4

where impl1, impl2, impl3 and impl4 represent the implementor-defined limits as specified in 6.1.4.6.

d) Rows in the IRDS Dictionary, IRD Object Version and IRD Object tables to reflect the existence of the new IRD Definition; the row in IRD Object Version shall refer to the Environment working set.

Table 7 - IRDS Dictionary

Dictionary Object Key	Dictionary Working Set Key	Dictionary Name	Dictionary Active	Schema Group Object Key	Schema Group Working Set Key
00070001	00020001	as specified	True	as specified	as specified

where "as specified" represents a value specified in the service request that created the IRD Definition.

e) Rows in the IRD Content Status, IRD Object Version and IRD Object tables to reflect the existence of the IRD Content Status values 'UNCONTROLLED', 'CONTROLLED' and 'ARCHIVED', each belonging to the IRD Content Status Class with the same name; the rows in IRD Object Version shall refer to the Common working set.

Table 26 - IRD Content Status

Object Key	Working Set Key	Constraint Enforced	Content Status Class
00260001	00020002	FALSE	UNCONTROLLED
00260002	00020002	TRUE	CONTROLLED
00260003	00020002	TRUE	ARCHIVED

There is no implication that the value of CONSTRAINT_ENFORCED for a particular IRD_CONTENT_STATUS_CLASS is restricted to that shown here.

f) Sets of rows in the IRD Working Set Privilege, IRD Object Version and IRD Object tables representing privileges for the IRDS User named IRDS Administrator to access the Environment and Common working sets; each set shall contain rows representing the existence of four privileges with the values of PRIVILEGE being SELECT, DELETE, INSERT and UPDATE respectively. The rows in IRD Object Version shall refer to the Common working set.

Table 28 - IRD Working Set Privilege

Object Key	Working Set Key	Grantor Key	Grantee Key	Working Set Key	Privilege	Grantable
00280001	00020002	00050001, 00020001	00050002, 00020001	00020001	UPDATE	TRUE
00280002	00020002	00050001, 00020001	00050002, 00020001	00020001	DELETE	TRUE
00280003	00020002	00050001, 00020001	00050002, 00020001	00020001	INSERT	TRUE
00280004	00020002	00050001, 00020001	00050002, 00020001	00020001	SELECT	TRUE
00280005	00020002	00050001, 00020001	00050002, 00020001	00020002	UPDATE	TRUE
00280006	00020002	00050001, 00020001	00050002, 00020001	00020002	DELETE	TRUE
00280007	00020002	00050001, 00020001	00050002, 00020001	00020002	INSERT	TRUE
00280008	00020002	00050001, 00020001	00050002, 00020001	00020002	SELECT	TRUE

g) Such rows in other tables, with corresponding rows in the IRD Object Version and IRD Object tables, as are needed to ensure the existence in any newly-created IRD of all the Common tables listed in 5.2; the rows in IRD Object Version shall refer to the Common working set.

6.2 IRD Level

6.2.1 IRD Level data structure

The IRD level data structure is described informally in 5.4, and formally in that part of an IRD Definition which was activated when each IRD was created; this will always include all the Common tables. The structure represented in figure 14 is defined for the purpose of specifying rules and describing the operation of IRDS services. There is no requirement that any implementation actually uses this structure.

Each IRD contains:

a) An instance of each Internal table defined in 6.1.4;

b) An instance of each Common table defined in 6.1.4;

c) Such further IRD-specific tables as have been defined in the applicable part of the associated IRD Definition.

6.2.2 IRD Level Initial Contents

NOTES

1 - The initial contents specified below are the minimum necessary to support the operation of an IRDS. They are not intended to imply any restriction on possible contents, other than those specified elsewhere in this International Standard.

2 - Where a value is provided for an object key or working set key, this is done purely to make the references between rows clear. In an actual implementation the values used in such columns are implementation-dependent.

3 - Where a reference from one table to another exists in the form of a pair of keys, one to the object and one to the working set, for clarity they are shown below as a single column, with a pair of values separated by commas. The object key always comes first.

4 - Corresponding rows in the Object and Object Version tables shall also exist, conforming to the structure and constraints specified in 6.1.4.

In any newly-created IRD, rows shall exist in the newly-created IRD tables as follows:

a) A row in the IRD Working Set table to reflect the existence of a non-versionable working set in this IRD named Common with a null working set version name.

Table 2 - IRD Working Set

Working Set Key	Working Set Name	Working Set Version Name	Based On Working Set Key	IRD Content Status Key	Versionable
00021002	Common	null	null	00261002, 00021002	FALSE

b) Rows in the IRD Content Status, IRD Object Version and IRD Object tables to reflect the existence of the IRD Content Status values 'UNCONTROLLED', CONTROLLED' and 'ARCHIVED', each belonging to the IRD Content Status Class with the same name. The rows in IRD Object Version shall refer to the Common working set in this IRD.

Table 26 - IRD Content Status

Object Key	Working Set Key	Constraint Enforced	Content Status Class
00261001	00021002	FALSE	UNCONTROLLED
00261002	00021002	TRUE	CONTROLLED
00261003	00021002	TRUE	ARCHIVED

There is no implication that the value of CONSTRAINT_ENFORCED for a particular IRD_CONTENT_STATUS_CLASS is restricted to that shown here.

c) A set of rows in the IRD Working Set Privilege, IRD Object Version and IRD Object tables representing privileges for the IRDS User named IRDS Administrator to access the Common working set in this IRD; this set shall contain four rows representing privileges with the values of PRIVILEGE being SELECT, DELETE, INSERT and UPDATE respectively. The rows in IRD Object Version shall refer to the Common working set in this IRD.

Table 28 - IRD Working Set Privilege

Privilege Object Key	Privilege Working Set Key	Grantor Key	Grantee Key	Working Set Key	Privilege	Grantable
00281001	00021002	00050001, 00020001	00050002, 00020001	00021002	UPDATE	TRUE
00281002	00021002	00050001, 00020001	00050002, 00020001	00021002	DELETE	TRUE
00281003	00021002	00050001, 00020001	00050002, 00020001	00021002	INSERT	TRUE
00281004	00021002	00050001, 00020001	00050002, 00020001	00021002	SELECT	TRUE

In addition, a row shall be added to the IRDS Dictionary table in the current IRD Definition representing the newly-created IRD as shown below; the corresponding row in IRD Object Version shall reference the Environment working set in the current IRD Definition.

Table 7 - IRDS Dictionary

Dictionary Object Key	Dictionary Working Set Key	Dictionary Name	Dictionary Active	Schema Group Object Key	Schema Group Working Set Key
00070002	00020001	as specified	TRUE	as specified	as specified

where "as specified" represents a value specified in the service request that created the IRD.

6.3 IRD General Rules

The following rules shall apply at all times to the contents of any IRD Definition or IRD:

6.3.1 Use of primary key

a) All IRD tables except internal tables contain the columns

i) IRD_OBJECT_KEY

ii) IRD_WORKING_SET_KEY

These columns form the primary key.

b) When a referential constraint is defined it shall reference only those columns that comprise the primary key of the referenced table.

c) Where a new row is created, or an existing row is modified, in any IRD table for which a referential constraint is specified, and a delete rule or update rule is specified for that referential constraint, the referenced row and the referencing row shall both exist in the materialization of the current working set.

d) The value of the IRD_WORKING_SET_KEY of any row in an Environment table shall be the value of the IRD_WORKING_SET_KEY of the Environment working set within the same IRD Definition.

e) The value of the IRD_WORKING_SET_KEY of any row in a Common table shall be the value of the IRD_WORKING_SET_KEY of the Common working set within the same IRD or IRD Definition.

6.3.2 References and content status

To preserve the integrity of each IRD while allowing evolutionary change, it is necessary to ensure that no object version with a Controlled IRD content status references any object version with an Archived or Uncontrolled IRD content status. It is also necessary to ensure that no object version with an Uncontrolled IRD content status references any object version with an Archived IRD content status, or vice versa.

References outside a working set may arise either within a version path (see 6.3.3 below) or along a reference path. To ensure that no referenced object may be changed outside the working set containing the reference, causing unexpected side effects, the following rules shall be enforced by any service which could cause them to be violated:

a) Any working set whose working set key is referenced in the BASED_ON_WORKING_SET_KEY column of one or more other working sets shall refer to an IRD content status belonging to the Controlled content status class;

b) Any working set whose working set key is referenced in the REFTO_WORKING_SET_KEY column of one or more IRD Reference Paths shall refer to an IRD content status belonging to the Controlled content status class.

6.3.3 Resolution of references

All working sets within the current context are materialized before referential constraints are interpreted.

6.3.4 Resolution of references within a version path

The "version path" of a working set WS_n is a sequence of working sets (WS_n, ..., WS_1) where each working set WS_j (n>=j>1) is based on WS_j-1.

Whenever a reference is made to an object version in the current version path, the working set identifier of the referenced object version is taken to be that of the working set in the current version path containing the latest version of the referenced object.

6.3.5 References depending on a reference path

Where a reference is created or amended so as to refer to an object version outside the current working set materialization, such a reference shall only be allowed if there is a reference path defined from the referencing working set to the referenced working set.

References along a reference path are resolved after materialization of both the From and To working sets; when this materialization occurs, the To working set shall be controlled (see 6.3.2). The working set identifier of an object version in the materialization of the To working set is taken to be the identifier of the To working set for the purposes of resolving references along the reference path.

6.3.6 Reference paths and version paths

No reference path shall exist from a working set A to any working set which is in the version path of A.

7 Services concepts and facilities

7.1 Levels and parallelism

The IRDS Services Interface provides services on two levels - the IRD level and the IRD definition level. The IRD level contains objects and attributes. The IRD definition level contains definition objects and definition attributes, some of which describe the structure of the IRD level. The IRDS services at this level are usually used only by the IRDS itself and by very special user applications.

The parallelism between the IRD level and the IRD definition level is reflected in the great similarity between the IRD and IRD definition services. In the remainder of clause 7, any topic refers to both levels unless otherwise explicitly stated.

7.2 Access to IRDS data via Database Services Processor

An IRDS Services Interface Processor may utilize the services of a database services processor to access the **IRDS databases** under its control.

7.2.1 Prevention of circumvention of IRDS security and integrity

Where an IRDS Services Interface Processor utilizes the services of a database services processor, the implementation shall be such that it shall not be possible to circumvent the security and integrity provisions of the IRDS Services Interface Processor through direct use of the database services processor.

One possible way of preventing such circumvention would be to define the IRDS Services Interface Processor as the only authorized processor of the IRDS databases via the database services processor.

7.2.2 Access to IRDS Data using a standard Database Language

An IRDS Services Interface Processor may optionally allow the use of a standard database language syntax to access the IRDS database. The IRDS Services Interface Processor is responsible for monitoring all such access to ensure that the IRDS security and integrity constraints are not subverted. Any statements that may potentially subvert the IRDS shall be rejected. The user or application issuing the database language requests must be prepared to handle error codes from the IRDS Services Interface Processor as well as those that may be produced by the database language processor.

7.3 Connecting an application to the IRDS Services Interface Processor

7.3.1 Sessions and transactions

To begin using the IRDS Services Interface, the first service that must normally be invoked is the Open IRDS service. This service connects the client application to the IRDS Services Interface Processor, and opens an IRDS session against a named IRD or IRD Definition. To end using the IRDS Services Interface, the last service that must be invoked is the Close IRDS service. This service disconnects the client application from the IRDS Services Interface Processor.

There are two exceptions to the above: the Create IRD Definition service and Drop IRD Definition services. These do not require a prior Open IRDS service. The Create IRD Definition service creates a new IRD Definition, and includes an implicit Open IRDS against the newly created IRD Definition. The Drop IRD Definition service drops an existing IRD Definition.

The entire time in which a client application is connected to the IRDS Services Interface Processor is called an IRDS session. Thus, an IRDS session always begins when the Open IRDS service is invoked and ends when the Close IRDS service is invoked.

A client transaction may include activity in multiple concurrent IRDS sessions. The activity of a client transaction within a single IRDS session is here referred to as an IRDS transaction.

An IRDS session can be divided into IRDS transactions. IRDS transaction boundaries are indicated by the client application using the Open IRDS, Commit, Rollback, and Close IRDS services. Since the scope of one IRDS transaction is within a single IRDS session, it is the client's responsibility to manage IRDS transactions among concurrent IRDS sessions.

All updates made to either level are made as part of an IRDS transaction. These updates are not considered permanent until that IRDS transaction is completed using either the Commit service or the Close IRDS service. In the event of a task or system failure, the IRDS Services Interface Processor will automatically roll back any IRDS transaction that was in progress as if that IRDS transaction had never been started. If for whatever reason the client application wishes to cancel all of the updates made as part of the current IRDS transaction, the Rollback service can be used. If the implementation of an IRDS Services Interface Processor uses a locking mechanism to ensure updating integrity in a multi-user IRDS environment, all locks shall be released by the Commit, Rollback and Close IRDS services.

The Open IRDS service starts the IRDS session and the first IRDS transaction of that IRDS session. The Prepare service indicates to the IRDS Services Interface Processor that the caller is about to invoke the Commit service, and wishes to ensure that it will be successful; this service is provided in order to enable a two-phase commit protocol to be used where an application needs to synchronize updates in more than one IRDS session, or in an IRDS session and a session of some other processor. The Commit service ends the current IRDS transaction and starts the next IRDS transaction by making permanent all of the IRD and IRD definition updates made during that IRDS transaction. The Rollback service ends the current IRDS transaction and starts the next IRDS transaction by cancelling all of the IRD and IRD definition updates made during that IRDS transaction. The Close IRDS service ends the last transaction and the session.

Throughout the remainder of this document the terms 'IRDS transaction' and 'IRDS session' are abbreviated to 'transaction' and 'session' respectively.

7.3.2 IRDS users and privileges

A user of the IRDS must be predefined to the system before that user can access the IRD or IRD definition. This is done by creating an instance of the IRDS User object type with an IRDS name equal to the name of the user.

When a client application requests IRDS services, it always does so on behalf of a user. At open time, the services interface validates the user's identifier and ensures that there is a matching IRDS User object. If there is not, the open fails and access is denied.

The IRDS User object for the user is potentially related to other objects that define access privileges. These privileges control the user's capabilities in accessing the IRD and IRD definition levels.

7.4 Object selection

In accessing data in an IRD or IRD definition, the client application must select objects and sets of objects. When modifying or deleting an object, the required object must first be selected and retrieved.

An IRDS user requesting a service does not need to specify which object versions to work with on each service call. The user selects objects, and the specific object versions selected are the ones associated with the user's current working set context. When specifying information to be retrieved, the IRDS user must specify that object versions are to be selected from the user's current context, or from selected working sets, or from all working sets accessible by the user.

An object can be added at any time without any prior selection, though a working set context must first have been established in which the object will be added.

7.5 Sets and cursors

Access to the IRD and IRD definition through the IRDS services interface is performed in terms of cursors, which provide access to selected sets of objects. This type of processing is similar in concept to the way in which the SQL DML is used to access relational data bases (ISO/IEC 9075).

The application specifies criteria for selecting a set of rows from a table. This is done by opening a cursor. The cursor then becomes associated with the specified criteria and the matching set of rows and is used in subsequent retrieval and update operations to process each row in the set. Rows can be fetched until a 'no data' indication is given. The cursor can then be closed to free any resources associated with it.

Cursors are used to establish position on rows representing object versions. When a cursor is first opened, the cursor is positioned before the first row. Each successful Retrieve Object service moves the cursor forward to the next row before retrieval, leaving the cursor positioned on the row retrieved.

The Modify Object service does not change the position of the cursor. The Delete Object service leaves the cursor positioned before the row following that deleted, if any. If the last row is deleted, or the Retrieve Object service is requested when the cursor is already positioned on the last row, the cursor is positioned after the last row.

This International Standard places no limit on the number of cursors that may be opened in a single session, nor on the number of sessions that may be active at the same time.

If multiple cursors are opened at the same time, any of which are for update, and any rows are visible through more than one cursor, the results are implementation-dependent.

All cursors are automatically closed at the end of a transaction or at the end of a session.

7.6 Diagnostics

At the completion of any service, the application receives immediate feedback about the outcome of the service by means of a status number returned in the RetCode parameter. This is a parameter to each service request.

The status number comprises a 2-character class followed by a 3-character subclass. Annex A provides details of the possible values and their meanings. Further details of the error can be obtained using the Get Diagnostics service.

In cases where multiple error conditions are detected by the services interface, the sequence in which status values are returned is implementation-dependent. The errors that occurred for any service except Get Diagnostics can be examined one by one through use of the Get Diagnostics service.

7.7 Version control

Invoking the Open IRDS service establishes an IRDS session for activity against either a specific IRD or a specific IRD Definition, and may establish a default working set as the current context of the session. This is referred to below as the 'current working set'.

Alternatively, the current working set may be established explicitly through use of the Set Context service.

Services invoked during a session allow objects to be added to, modified in, and deleted from the current working set.

With the Common working set as the current context, the Add Object service can be used to add a working set privilege. The Create Working Set service can be used to create a new working set.

When creating a new working set using the Create Working Set service, the parameters BasisWkgSetName and BasisWkgSetVerName together may specify a working set from which object versions are to be copied. This effectively causes copies to be created of all the object versions currently within the specified working set, assigning to each of them the working set name and version identifier associated with the new working set. If the new working set is to be non-versionable, these parameters shall be null.

The Modify Content Status service can be used to alter the IRD Content Status of a specified working set. Altering the IRD Content Status requires the IRDS Services Interface Processor to perform all consistency checks necessary to ensure that no referential integrity constraints are violated.

7.8 Operations on Abstract Data Structures

The effect of the services in clause 9 is described in terms of changes to the abstract data structures described in clause 6. As stated in clause 6, it is not required that an implementation use physical data structures corresponding directly to these abstract data structures. It is also not required that the implementation of a particular service cause operations on the physical data structures that correspond directly to the operations specified in clause 9.

8 Service data structures

8.1 Basic data constants

Each implementation of this international standard can specify, within certain constraints, values for certain limits.

These limits are defined in the tables IMP_LIMITS and INSTALLATION_DEFAULT. In order for the IRDS services interface to be adaptable, the services interface and client application both require access to this information in the form of constants. These constant values allow an application to be readily ported from one IRDS implementation to another.

Many of the constants defined in this section fulfil this goal. Others are particular to the IRDS Services Interface. Note that where non-numeric values beginning with 't' are given, the actual value is always taken from the similarly named row in one of the tables specified above. Where non-numeric values beginning with 'n' are given, the actual value is implementor-defined, but is not represented in one of the above tables. In either case, each standard-conforming implementation will provide documentation of the appropriate actual values.

NOTE - For the Pascal language binding that is implicit in this version of this International Standard, enumerated types are used wherever possible in preference to numeric or character constants with predefined values. For bindings to other languages, a similar mechanism should be used where available. If no directly equivalent mechanism exists, the most similar possible mechanism should be used.

8.1.1 Name Length Limits

Within the IMP_LIMITS table, there are two columns that define limits on the lengths of various IRDS names. The corresponding services interface constants are in the following form:

```
IrdsNameLim      = t1;
IrdsVarLim       = t2;
```

IrdsNameLim is derived from the column IMP_NAME_LIM of table IMP_LIMITS (see 6.1), and defines the maximum length of an IRDS name.

IrdsVarLim is derived from the column IMP_VAR_LIM of table IMP_LIMITS (see 6.1), and defines the maximum length of a variation name.

8.1.2 Attribute Length Limits

Within the IMP_LIMITS table, there is a column that defines a limit on the lengths of the text domain format. The corresponding services interface constant is:

```
IrdsTextLim      = t3;
```

IrdsTextLim is derived from the column IMP_TEXT_LIM of table IMP_LIMITS (see 6.1), and defines the maximum length of a text value.

8.1.3 Control Identifier Length Limits

The IRDS services interface has several data types that are used to define various control identifiers. Constants that define the actual lengths of these character string types are

```
IrdsSessIdLim        = n1;
IrdsCurIdLim         = n2;
IrdsImpDicNameLen    = n3;
IrdsKeyLen           = n4;
```

where n1, n2, n3, n4 are implementor-defined.

IrdsSessIdLim defines the length of the IrdsSessId data type (defined below).

IrdsCurIdLim defines the length of the IrdsCursorId data type (defined below).

IrdsImpDicNameLen is derived from the column IMP_DIC_NAME_LEN of the single row in the table IMP_LIMITS (see 6.1), and defines the length of the IrdsImpDicName data type (defined below).

IrdsKeyLen defines the length of the IrdsKey data type, defined formally by the IRDS_KEY domain in clause 6, and for the services interface in 8.2.1 below

8.1.4 Data Types

Each IRD_DOMAIN defined in the IRD Definition has a value for the column DATA_TYPE, corresponding to one of the data types specified in ISO/IEC 9075. The allowable values for this column are defined in 6.1. For the services interface, these values have been equated to the enumerated type IrdsDataType:

```
IrdsDataType=(
    IrdsDataTypeChar,      {CHARACTER}
    IrdsDataTypeCharVar, {CHARACTER VARYING}
    IrdsDataTypeNatChar, {NATIONAL CHARACTER}
    IrdsDataTypeNatCharVar,
                          {NATIONAL CHARACTER VARYING}
    IrdsDataTypeReal,      {REAL}
    IrdsDataTypeDouble,    {DOUBLE PRECISION}
    IrdsDataTypeFloat,     {FLOAT}
    IrdsDataTypeInteger,   {INTEGER}
    IrdsDataTypeSmallint, {SMALLINT}
    IrdsDataTypeNumeric, {NUMERIC}
    IrdsDataTypeDecimal, {DECIMAL}
    IrdsDataTypeDate,      {DATE}
    IrdsDataTypeTime,      {TIME}
    IrdsDataTypeTimestamp,   {TIMESTAMP}
    IrdsDataTypeInterval, {INTERVAL}
    IrdsDataTypeIrdsKey   {IRDS KEY}
    );
```

8.1.5 IRD Content Status Classes

Each IRD_CONTENT_STATUS defined in the IRD Definition has a value for the column IRD_CONTENT_STATUS_CLASS. The allowable values for this column are defined in 6.1. For the services interface, these values have been equated to the following values of the enumerated type IrdsDcsCls:

```
IrdsDcsCls = (
     IrdsDcsClsUcntl,  {UNCONTROLLED}
     IrdsDcsClsCntl,   {CONTROLLED}
     IrdsDcsClsArch    {ARCHIVED}
     );
```

8.1.6 Close Type parameter

The Close IRDS service has a parameter that indicates whether any uncommitted transaction should be committed or rolled back. The possible values of this parameter are defined by the following enumerated type IrdsCloseType:

```
IrdsCloseType =  (
     RequestIrdsCommit,        { COMMIT }
     RequestIrdsRollback       { ROLLBACK }
     );
```

8.2 Service data types

Based on the constants defined in the previous section, the corresponding types can now be defined. While the values of some of the constants are implementation-defined, the types are implementation-independent due to their references to the constants previously defined.

8.2.1 Column data types

A column has a specific data type as specified in the SQL standard (ISO/IEC 9075), which corresponds to a particular Pascal data type, as follows:

Columns with a data type of INTEGER or SMALLINT, also NUMERIC or DECIMAL where both NUMERIC_SCALE and NUMERIC_PRECISION are zero, map to the Pascal integer type.

Columns with a data type of REAL, DOUBLE PRECISION or FLOAT, also NUMERIC or DECIMAL where either NUMERIC_SCALE or NUMERIC_PRECISION is non-zero, map to the Pascal real type.

Columns with a data type of BOOLEAN map to the Pascal boolean type.

Columns with a data type of CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER or NATIONAL CHARACTER VARYING map to a Pascal type defined as follows:

```
IrdsText = packed array [1..IrdsTextLim] of char;
```

Columns with a data type of DATE map to a Pascal type defined as follows:

```
IrdsDate = record
     Year         :     packed array [1..4] of char;
     Sep1         :     char;
     Month        :     packed array [1..2] of char;
     Sep2         :     char;
     Day          :     packed array [1..2] of char;
end;
```

where the value of each of Sep1 and Sep2 is ignored on input, and is '-' on output.

Columns with a data type of TIME map to a Pascal type defined as follows:

```
IrdsTime = record
     Hour         :     packed array [1..2] of char;
     Sep1         :     char;
     Minute       :     packed array [1..2] of char;
     Sep2         :     char;
     Second       :     packed array [1..2] of char;
     Sep3         :     char;
     Fraction     :     packed array [1..3] of char;
end;
```

where the value of each of Sep1 and Sep2 is ignored on input, and is ':' on output, and the value of Sep3 is ignored on input, and is '.' on output.

Columns with a data type of TIMESTAMP map to a Pascal type defined as follows:

```
IrdsTimestamp = record
     Date         :     IrdsDate;
     SepT         :     char;
     Time         :     IrdsTime;
end;
```

where the value of SepT is ignored on input, and is ':' on output.

Columns with a data type of INTERVAL map to a Pascal type defined as follows:

```
IrdsInterval = record
     Days         :     packed array[1..7] of char;
     SepI         :     char;
     Time         :     IrdsTime;
end;
```

where the value of SepI is ignored on input, and is ':' on output.

Columns with a data type of IRDS KEY map to a Pascal type defined as follows:

```
IrdsKey  = packed array [1..IrdsKeyLen] of char;
```

NOTE - Note that in the absence of Pascal data types corresponding directly with some of the SQL data types, some of these mappings imply a possible loss of precision.

8.2.2 Object Names

Each type of name in the services interface is given a separate data type as specified below. The specific length of each type is controlled by the corresponding implementation-defined constant previously defined.

```
IrdsSQLName = packed array [1..128] of char;
IrdsName =
    packed array [1..IrdsNameLim] of char;
IrdsVarName =
    packed array [1..IrdsVarLim] of char;
UserId =
    packed array [1..IrdsNameLim] of char;
```

IrdsSQLName is the data type used to define all IRD Definition level table, column, domain and constraint names.

IrdsName is the data type used to define all IRDS names.

IrdsVarName is the data type used to define all variation names.

UserId is the data type used to define the identifiers, in implementation-defined format, by which IRDS users are identified to the IRDS..

8.2.3 Control Identifiers

The data types for control identifiers used as parameters to various services are specified below. Reference is made to some implementation-defined constants previously defined.

```
IrdsSessId =
    packed array [1..IrdsSessIdLim] of char;
IrdsCurId =
    packed array [1..IrdsCurIdLim] of char;
IrdsImpDicName =
    packed array [1..IrdsImpDicNameLen] of char;
```

IrdsSessId is the data type used to define the session identifier parameters used in clause 9.

IrdsCurId is the data type used to define the cursor identifier parameters used in clause 9.

IrdsImpDicName is the data type used to define the IRD Definition name and IRD name parameters used in clause 9.

8.2.4 Diagnostics Area

At the completion of each service except Get Diagnostics, regardless of the outcome, one or more return states are recorded in the diagnostics area by the services interface. These state records can be retrieved, one at a time, using the Get Diagnostics service. The structure of a state record is specified below:

```
IrdsState= record
    StateClass       : packed array [1..2] of char;
    StateSubclass    : packed array [1..3] of char;
end;

IrdsStateRec = record
    IrdStateSeq             : integer;
    IrdReturnedState        : IrdsState;
    IrdConstraintSchema     : IrdsSQLName;
    IrdConstraintName       : IrdsSQLName;
    IrdSchemaName           : IrdsSQLName;
    IrdTableName            : IrdsSQLName;
    IrdColumnNumber         : integer;
    IrdColumnName           : IrdsSQLName;
end;
```

IrdState defines the state number data type for the IRDS. The IRDS state is of the form CCNNN where CC is the two digit class and NNN is the three digit subclass.

IrdStateSeq is the sequence number of the state record. All state records created by the same service invocation are consecutively numbered beginning with one.

IrdReturnedState is the state that would have been returned if this had been the only condition raised.

IrdConstraintSchema, IrdConstraintName, IrdSchemaName, IrdTableName, IrdColumnNumber and IrdColumnName contain additional information depending on the state being reported. Their values are described fully in Annex A.

8.2.5 Service Return Code

The following type definition defines the structure of the parameter in which each service reports its success or failure.

```
IrdsRetCode = record
    NumStates   : integer;
    State       : IrdState;
end;
```

NumStates defines the number of non-zero state records stored in the diagnostics area as a result of the previous service.

State defines the state number of the numerically highest state detected during performance of the service.

8.2.6 Column List Parameters

The following type definition defines the structure of the column list parameters used to pass data values to and from the services interface.

```
IrdsColList    = record
     NumCols      : integer;
     ColSpec      : array [1..NumCols] of
          record
               ColName      :     IrdsSQLName;
               ColNull      :     boolean;
               ColType      :     IrdsDataType;
               case ColType of
                    IrdsDataTypeChar,
                    IrdsDataTypeCharVar,
                    IrdsDataTypeNatChar,
                    IrdsDataTypeNatCharVar:
                         (ColValText: IrdsText);
                    IrdsDataTypeReal,
                    IrdsDataTypeDouble,
                    IrdsDataTypeFloat:
                         (ColValReal:  real);
                    IrdsDataTypeInteger,
                    IrdsDataTypeSmallint,
                    IrdsDataTypeNumeric,
                    IrdsDataTypeDecimal:
                         (ColValInteger: integer);
                    IrdsDataTypeDate:
                         (ColValDate: IrdsDate);
                    IrdsDataTypeTime:
                         (ColValTime: IrdsTime);
                    IrdsDataTypeTimestamp:
                         (ColValTimestamp:
                              IrdsTimestamp);
                    IrdsDataTypeInterval:
                         (ColValInterval:
                              IrdsInterval);
                    end;
               end;
     end;
```

NumCols defines the number of columns specified by the column list.

ColName specifies the name of each column.

ColNull specifies whether the specified column has the NULL value.

ColType specifies the data type of each column.

The variant appropriate to ColType contains the supplied or returned value.

9 Service Formats and Descriptions

This clause describes all of the services provided by this interface. The services are presented in three groups: operational services, level independent services and level specific services. For each service, the following items of information are supplied where relevant:

Function
Format
Input
Output
General rules
Operations on the Abstract Data Structure

The service Formats are specified in this version of this standard using Pascal as defined in ISO 7185. In each case, a broad indication of the success or failure of the service is provided in the RetCode parameter; subsequent use of the Get Diagnostics service will provide more detailed information about the success or failure of the most recent service request. Specifically, after every service request, RetCode.State is set to the highest state detected during processing of the service request, and RetCode.NumStates is set to the number of different states about which further information may be retrieved using the Get Diagnostics service.

The IRDS services defined in 9.1, 9.2 and 9.3 must be invoked in a prescribed sequence. This sequence is defined in 9.4.

9.1 Operational services

The operational services of the IRDS Services Interface are used by the client to initiate and terminate sessions and to initiate and terminate transactions. These services are defined below.

9.1.1 Create IRD Definition Service

Function

This service creates a new IRD Definition, and initiates an IRDS session at the IRD Definition level. It also initiates the first transaction within that session.

Format

```
procedure    IrdsCreateIRDDefinition
    (     IrdsUser      :    UserId;
          IrdDefName  :    IrdsImpDicName;
    var CurrSessId   :    IrdsSessId;
    var RetCode       :    IrdsRetCode);
```

Input

The following variables shall be set:

IrdsUser := User's name to be used in checking user access privileges;

IrdDefName := the identification of the required IRD Definition, in an implementor-defined format;

Output

If the service is successful, CurrSessId is set to a system-defined session identifier that shall be used as a parameter in all further service requests during this IRDS session.

Possible error states are

21	IRD Definition name already exists
28	User name not known
42	User does not have appropriate privileges

General Rules

All input parameters are validated individually, and then for consistency according to the following rules:

1 IrdDefName shall contain a valid identifier for an IRD Definition, in the format required by the implementation;

2 No IRD Definition shall already exist with the specified name within the real system within which the service is invoked;

3 No working set context is established;

4 The mechanism for determining whether a user may use this service is implementation-defined;

5 On successful completion of this service, the IRD_DEF IRDS schema defined in 6.1.2.1 is effectively mapped to an SQL schema whose <schema name> is 'IRD_DEF'. The effective mapping of this SQL schema to an SQL catalog is implementation-defined, but shall be such that no name clash occurs.

Operations on the Abstract Data Structure

An IRD Definition is created containing the domains and tables specified in 6.1, with initial contents as specified in 6.1.7.

9.1.2 Drop IRD Definition Service

Function

This service drops an IRD Definition.

Format

```
procedure IrdsDropIRDDefinition
      (    IrdsUser      :    UserId;
           IrdDefName :    IrdsImpDicName;
      var  RetCode       :    IrdsRetCode);
```

Input

The following variables shall be set:

IrdsUser := User's name to be used in checking user access privileges;

IrdDefName := the identification of the IRD definition to be dropped, in an implementor-defined format;

Output

Possible error states are

02	IRD Definition does not exist
25	IRD Definition is in use
28	User name not known
42	User does not have appropriate privileges

General Rules

All input parameters are validated individually, and then for consistency according to the following rules:

1 IrdDefName shall contain a valid identifier for an existing IRD Definition, in the format required by the implementation;

2 The mechanism for determining whether a user may use this service is implementation-defined;

3 There shall be no open IRDS session on the specified IRD Definition, for this or any other user, at either IRD level or IRD Definition level;

4 On successful completion of this service the IRD Definition and any constituent IRD Schema Groups, together with any IRDs they control, shall cease to exist;

5 Each invocation of this service forms a separate transaction, which is automatically committed on successful completion.

Operations on the Abstract Data Structure

See general rule 4.

9.1.3 Open IRDS Service

Function

This service initiates an IRDS session at the IRD level or IRD Definition level, depending on the value of the IrdDicName parameter. It also initiates the first transaction within the session.

Format

```
procedure IrdsOpen
      (    IrdsUser       :    UserId;
           IrdDefName  :    IrdsImpDicName;
           IrdDicName   :    IrdsName;
           WillUpdate    :    boolean;
      var  CurrSessId   :    IrdsSessId;
      var  RetCode        :    IrdsRetCode);
```

Input

The following variables shall be set:

IrdsUser := User's name to be used in checking user access privileges;

IrdDefName := the identification of the required IRD definition, in an implementor-defined format;

IrdDicName:= blanks if access to the IRD Definition level is required; the name of the desired IRD if access to the IRD level is required.

WillUpdate := false if access to the specified IRD or IRD Definition is to be read only, or true if update access may be required.

Output

If the service is successful, CurrSessId is set to a system-defined session identifier that shall be used as a parameter in all further service requests during this IRDS session.

Possible error states are

02	Specified component does not exist
28	User name not known
42	User does not have appropriate privileges
52	IRD must first be re-activated

General Rules

All input parameters are validated individually, and then for consistency according to the following rules:

1 IrdDefName shall contain a valid identifier for an IRD Definition, in the format required by the implementation;

2 If IrdDicName is not blank, it shall be the name of an IRD created within the IRD Definition identified by IrdDefName, and that IRD shall have a value of true for DICTIONARY_ACTIVE;

3 The IRDS User requesting this service shall have access privileges to at least one working set in the specified IRD or IRD definition. If WillUpdate is true, these privileges shall not be solely for select access.

4 There shall be no requirement that there should be no existing IRDS session open for the same or a different user for the same IRD or IRD Definition. In other words, an IRDS shall allow multiple IRDS sessions for the same or different users, and for the same or different IRDs or IRD Definitions.

5 If the column Default Working Set Key of the row in the IRDS User table corresponding to the user specified by the IrdsUser parameter is not null, it shall identify a working set in the IRD Definition if access to the IRD Definition level is requested, or a working set in the specified IRD if access to the IRD level is requested. This working set shall be established as the working set context.

Operations on the Abstract Data Structure

None.

9.1.4 Prepare Service

Function

This service prepares the current IRDS transaction of the specified IRDS session for commitment. If the service is successful, the IRDS guarantees that all of the updates that have been made as part of the current transaction can be made permanent or rolled back; i.e. either Commit (9.1.5) or Rollback (9.1.6) will be successful.

Format

```
procedure    IrdsPrepare
    (        CurrSessId    :    IrdsSessId;
         var RetCode       :    IrdsRetCode);
```

Input

The following variable shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service.

Output

Possible error states are

25 Invalid session identifier
91 Transfer failure

General Rules

If no errors are detected, all updates of the current transaction are secured.

Operations on the Abstract Data Structure

None.

9.1.5 Commit Service

Function

This service terminates the current IRDS transaction of the specified IRDS session by permanently applying all of the updates made as part of the current transaction. Any open cursors in the specified IRDS session are closed.

Format

```
procedure    IrdsCommit
    (        CurrSessId    :    IrdsSessId;
         var RetCode       :    IrdsRetCode);
```

Input

The following variable shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

Output

Possible error states are

25 Invalid session identifier
91 Transfer failure

General Rules

1 If no errors are detected, all updates of the current transaction are committed and all open cursors are closed.

Operations on the Abstract Data Structure

None.

9.1.6 Rollback Service

Function

This service terminates the current IRDS transaction of the specified IRDS session by rolling back (or reversing) all of the updates made as part of the current transaction. Any open cursors are closed.

Format

```
procedure   IrdsRollback
    (       CurrSessId   :   IrdsSessId;
        var RetCode      :   IrdsRetCode);
```

Input

The following variable shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

Output

Possible error states are

25 Invalid session identifier
91 Transfer failure

General Rules

If no errors are detected, all updates of the current transaction are backed out and all open cursors are closed.

Operations on the Abstract Data Structure

None.

9.1.7 Close IRDS Service

Function

This service terminates the specified IRDS session. It also terminates the last IRDS transaction of the specified IRDS session with the Commit/Rollback option specified in the CloseType parameter.

Format

```
procedure   IrdsClose
    (       CurrSessId   :   IrdsSessId;
            CloseType    :   IrdsCloseType;
        var RetCode      :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

CloseType := either RequestIrdsCommit or RequestIrdsRollback;

Output

Possible error states are

25 Invalid session identifier
91 Transfer failure

General Rules

1 If no errors are detected, all updates of the current transaction are committed or rolled back respectively according to the value of CloseType, and the specified IRDS session is terminated.

Operations on the Abstract Data Structure

None.

9.1.8 Get Diagnostics Service

Function

This service retrieves a state record created by the last invoked service other than Get Diagnostics.

Format

```
procedure   IrdsGetDiagnostics
    (       CurrSessId   :   IrdsSessId;
            StateNum     :   integer;
        var StateRec     :   IrdsStateRec;
        var RetCode      :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by the previous Open IRDS service;

StateNum := an integer value indicating which of the state records in the Diagnostics Area is to be returned in the StateRec parameter.

Output

The state data requested is returned in the StateRec parameter. For details see annex A.

Possible error states are

25 Invalid session identifier
35 State number too high

Note that in the special case of the Get Diagnostics service, no further information is available about the cause of the error.

General Rules

1 The value of StateNum shall be no greater than the value of the RetCode.NumStates field returned by the previous service.

Operations on the Abstract Data Structure

None.

9.2 Level independent services

The level independent services of the IRDS Services Interface are used to manipulate data in the IRD or IRD Definition.

9.2.1 Set Context Service

Function

This service establishes the context for future service requests within the current IRDS session.

Format

```
procedure   IrdsSetContext
    (       CurrSessId         :   IrdsSessId;
            SessWkgSetName     :   IrdsName;
            SessWkgSetVerId    :   IrdsName;
            WillUpdate         :   boolean;
        var RetCode            :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

SessWkgSetName := the name of the required working set;

SessWkgSetVerId := the version identifier of the required working set;

WillUpdate := false if access to the specified working set is to be read only, or true if update access may be required.

Output

Possible error states are

02	Working set does not exist
25	Invalid session identifier
42	User does not have appropriate privileges
53	Content status rule violation

General Rules

1 The IRDS User requesting this service shall have access privileges to the specified working-set. If WillUpdate is true, these privileges shall not be solely for select access.

2 If WillUpdate is true, then

either the specified working set shall have an IRD Content Status that belongs to the Uncontrolled IRD Content Status Class,

or the specified working set shall have an IRD Content Status that belongs to the Controlled IRD Content Status Class

and the specified working set shall be non-versionable.

3 The selected working set shall be at the same level (IRD level or IRD Definition level) as was set for the current session by the Open IRDS service.

4 If a value of False was specified for WillUpdate in the Open IRDS service for the current session, the value of the WillUpdate parameter for this service shall be False.

Operations on the Abstract Data Structure

None.

9.2.2 Add Object Service

Function

This service adds an object version within the current working set context.

Format

```
procedure   IrdsAddObject
    (       CurrSessId   :   IrdsSessId;
            ObjType      :   IrdsName;
            NewCols      :   IrdsColList;
        var RetCode      :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjType := the name of the IRD Table into which a new row is to be inserted;

NewCols.NumCols := the number of columns specified in the column list;

NewCols.ColNull := a null indicator for each specified column: true if the specified column is to have a null value, false if the value is not null.

NewCols.ColName := the names of the columns whose values are supplied; as many names shall be specified as indicated by the NumCols parameter.

NewCols.ColType := the types of values being supplied for the specified columns.

NewCols.ColValxxx := the values of the specified columns, each in the variant corresponding to its stated data type.

Output

Possible error states are

04	No value supplied for required column
06	Object already exists
07	Invalid column name
21	Specified component name already exists
22	Data exception
23	Constraint violation
25	Invalid session identifier
42	User does not have appropriate privileges
53	Content status rule violation
54	No current working set context has been established
55	A value has been specified for a column which is system-maintained
66	A service cannot operate directly on an Internal table
68	Invalid working set context

General Rules

1 A current single working set context shall have been established;

2 The current working set shall have an IRD Content Status belonging to the Uncontrolled IRD Content Status Class, or shall be non-versionable;

3 A non-null value shall be specified for each column that is defined as not null for the specified IRD Table;

4 No column that is defined as system-maintained shall be included in the column list;

5 The IRDS User who invokes this service shall have an Insert privilege on the current working set;

6 Case:

a If the current working set is at the IRD Definition level, ObjType shall specify the name of any IRD Definition table defined in 6.1.4 except an Internal table (see 5.3);

b If the current working set is at the IRD level, ObjType shall specify the name of a Common table defined in 6.1.4, or the name of any IRD table within an IRD Schema referenced by the IRD Schema Group controlling the current IRD;

7 Each name of a column specified in Newcols.Colname shall be the name of a column defined for the named table or for one of its direct or indirect supertables;

8 If the ObjType parameter references an IRD Table that has a value of false for VERSIONABLE, the current working set shall also have a value of false for VERSIONABLE;

9 If the row is to be inserted into a subtable, there shall be no row with the same primary key already existing in any direct or indirect supertable.

Operations on the Abstract Data Structure

1 If the current IRDS session is for IRD level access:

i a row with the specified column values is inserted in the appropriate IRD Table;

ii a matching row is inserted in the IRD Object Version table, with a reference from the first row inserted;

iii a row is inserted in the IRD Object table, with a reference from the second row inserted.

iv If the row is to be inserted into a subtable, then for each supertable a corresponding row is inserted;

2 If the current IRDS session is for IRD Definition level access:

i a row with the specified column values is inserted in the appropriate IRD Definition table;

ii a matching row is inserted in the IRD Object Version table, with a reference from the first row inserted;

iii a row is inserted in the IRD Object table, with a reference from the second row inserted.

9.2.3 Open Cursor Service

Function

This service opens a cursor on the set of object versions selected by the specified query expression. The opened cursor is positioned before the first object version in the set.

This International Standard places no limit on the number of cursors that may be opened in a single session.

The only data returned by this service is an internal cursor identifier that may be used as a parameter to the Retrieve Object, Modify Object, Delete Object or Close Cursor service.

Format

```
procedure   IrdsOpenCursor
    (       CurrSessId      :   IrdsSessId;
            UseContext      :   boolean;
            ObjSelExpr      :   IrdsTxt;
            WkgSetName      :   IrdsName;
            WkgSetVerId     :   IrdsName;
            FullContext     :   boolean;
            WillUpdate      :   boolean;
        var ObjCurId    :   IrdsCurId;
        var RetCode         :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

UseContext := if true, indicates that only the object versions visible in the current context are to be visible through this cursor, and the values of WkgSetName and WkgSetVerId are ignored; if false, indicates that the values of WkgSetName and WkgSetVerId are to be used in establishing the context specifier for the cursor;

ObjSelExpr := an SQL query with the SQL syntax of <cursor-specification> to be used, in the context of the current working set or specified working sets, to establish the objects to be visible through this cursor; if blank, all objects visible in the current working set or specified working sets are included in the cursor;

WkgSetName:= If UseContext is false, the name of the working set to be used as the context specifier for the cursor.

WkgSetVerId:= If UseContext is false, the version identifier of the working set to be used as the context specifier for the cursor.

FullContext:= If true, a full context is to be used; if false, a single context is to be used.

WillUpdate := false if access to objects reached through the new cursor is to be read-only, or true if update access may be required.

Output

ObjCurId := a cursor identifier to be passed as a parameter to any other service using the newly-opened cursor.

Possible error states are

03	No data selected
22	Data exception
25	Invalid session identifier
42	User does not have appropriate privileges
45	Specified query does not define an updatable cursor
46	Current session does not allow updating
47	Cursor specification includes a join that omits working set key
53	Content status rule violation
54	No current working set context has been established

General Rules

1 If UseContext is false, WkgSetName and WkgSetVerId shall together specify a working set, and this shall be used as the context specifier for the cursor.

2 If UseContext is true, there shall already be a current working set context, and this shall be used as the context specifier for the cursor.

3 If FullContext is true, the full context defined by the context specifier (see above) shall be used; if FullContext is false, the single context shall be used;

4 If object selection is specified, the effect of the selection shall be to include only those objects that are within the specified context, that satisfy the selection expression and for which the IRDS User requesting this service has access privileges;

5 If WillUpdate is true, then FullContext shall be false. If UseContext is also false, then

either the working set specified by WkgSetName and WkgSetVerId shall have an IRD Content Status that belongs to the Uncontrolled Content Status Class

or the specified working set shall have an IRD Content Status that belongs to the Controlled Content Status Class,

and the specified working set shall be non-versionable.

6 ObjSelExpr contains text that conforms to the SQL clause <cursor specification>. If WillUpdate is true, this <cursor specification> shall specify an updatable cursor;

7 Where the cursor specification includes a join, the columns on which the join is made shall not include the object key portion of the primary key of any table unless they also include the working set key portion of the primary key of that same table;

8 The IRDS User who invokes this service shall have a Select privilege on the current working set if UseContext is true, or on the specified working set if UseContext is false.

9 If a value of false was specified for WillUpdate in the most recent Open IRDS or Set Context service for the current session, the value of the WillUpdate parameter for this service shall be false;

10 If ObjSelExpr is not specified, or the selection specified results in a heterogeneous collection of object versions of two or more object types, the result of this service shall be as if ObjSelExpr included a <query specification> of "SELECT * FROM IRD_OBJECT_VERSION_VIEW";

11 On successful completion of this service, the cursor is positioned before the first row selected by the cursor.

Operations on the Abstract Data Structure

None.

9.2.4 Retrieve Object Service

Function

This service retrieves the current object of the specified cursor and places the values of the requested columns in the appropriate positions in the column list parameter.

Format

```
procedure   IrdsRetrieveObject
    (       CurrSessId      :   IrdsSessId;
            ObjCurId      :   IrdsCurId;
        var RequestedCols   :   IrdsColList;
        var RetCode         :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjCurId := the object cursor identifier returned by a previous Open Cursor service;

RequestedCols.NumCols := the number of columns specified in the column list;

RequestedCols.ColName := the names of the columns whose values are required; as many names shall be specified as indicated by the NumCols parameter.

Output

RequestedCols.ColNull := a null indicator for each requested column: true if the specified column has a null value, false if the value is not null.

RequestedCols.ColType := the types of the requested columns, indicating the variant in which each value will be found.

RequestedCols.ColValxxx := the values of the requested columns, each in the variant corresponding to its stated data type.

Possible error states are:

03	No further data
07	Invalid column name
24	Invalid cursor state
25	Invalid session identifier

General Rules

1 If the cursor is positioned before a row, it shall be positioned on that row;

2 If the cursor is positioned on a row, it shall be positioned on the next row. If there is no such row, the cursor shall be positioned after the last row, and the service shall return the error state appropriate to "No further data";

3 Each name of a column specified in RequestedCols.Colname shall be the name of a column defined for the named table or for one of its direct or indirect supertables;

4 On successful completion of this service, the cursor is positioned on the row retrieved;

5 If this service is unsuccessful, the cursor position is unchanged, and whether any values are returned is implementation-dependent.

Operations on the Abstract Data Structure

None.

9.2.5 Modify Object Service

Function

This service modifies the current object version of the specified cursor by replacing the values of the specified columns with those in the appropriate positions in the column list parameter.

Format

```
procedure   IrdsModifyObject
    (       CurrSessId      :     IrdsSessId;
            ObjCurId    :   IrdsCurId;
        var ModifiedCols    :     IrdsColList;
        var RetCode         :     IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjCurId := the object cursor identifier returned by a previous Open Cursor service;

ModifiedCols.NumCols := the number of columns specified in the column list;

ModifiedCols.ColName := the names of the columns whose values are to be modified; as many names shall be specified as indicated by the NumCols parameter.

ModifiedCols.ColNull := a null indicator for each specified column: true if the specified column is to be set to a null value, false if a non-null value is provided.

ModifiedCols.ColType := the types of the specified columns, indicating the variant in which each new value has been placed.

ModifiedCols.ColValxxx := the new values of the specified columns, each in the variant corresponding to its stated data type.

Output

Possible error states are:

04	No value supplied for required column
07	Invalid column name
22	Data exception
23	Constraint violation
24	Invalid cursor state
25	Invalid session identifier
27	Triggered data change violation
42	User does not have appropriate privileges
43	Invalid referential action
44	With Check Option violation
53	Content status rule violation
54	No current working set context has been established
55	A value has been specified for a column which is system-maintained

General Rules

1 A current single working set context shall have been established, and the cursor referenced by ObjCurId shall be updatable and shall be positioned on a row. The object version on which the cursor is positioned shall be within the materialization of the working set specified by the context;

2 A null value shall not be specified for any column that is defined as not null;

3 No column that is defined as system-maintained shall be included in the column list;

4 The IRDS User who invokes this service shall have an Update privilege on the current working set;

5 Each name of a column specified in ModifiedCols.ColName shall be the name of a column defined for the named table or for one of its direct or indirect supertables;

6 The current working set shall have an IRD content status belonging to the Uncontrolled content status class, or shall be a non-versionable working set;

7 If the <cursor specification> for the cursor identified by ObjCurId contained an <order by clause>, no column referenced by that <order by clause> shall be included in the columns referenced in the ModifiedCols parameter;

8 The cursor position shall not be changed by this service.

Operations on the Abstract Data Structure

1 If the modified object version is one of the superimposed object versions belonging to the current working set then:

Case:

a If the IRDS was opened for IRD level access, the specified columns in the current row in the appropriate IRD table are modified as requested; if any of these columns is actually defined in the IRD Object Version table, the referenced row in that table is modified accordingly.

If the table containing the row to be modified is a subtable and any inherited columns are specified in ModifiedCols.Colname, the corresponding row in each supertable containing a modified column is also modified.

b If the IRDS was opened for IRD Definition level access, the specified columns in the current row in the appropriate IRD Definition table are modified as requested; if any of these columns is actually defined in the IRD Object Version table, the referenced row in that table is modified accordingly.

2 If the modified object version exists in the materialization of the working set that is the basis for the current working set then:

Case:

a If the IRDS was opened for IRD level access, a row is inserted in the IRD Object Version table with the IRD Working Set key of the current working set and with the same IRD Object key, and a row is created in the appropriate IRD table with a reference to the new IRD Object Version row. The column values in each new row are taken from those of the original object version, modified by the specified new values.

If the table containing the row to be modified is a subtable and any inherited columns are specified in ModifiedCols.ColName, a corresponding row is created in each of its supertables.

b If the IRDS was opened for IRD Definition level access, a row is inserted in the IRD Object Version table with the IRD Working Set key of the current working set and with the same IRD Object key, and a row is created in the appropriate IRD Definition table with a reference to the new IRD Object Version row. The column values in each new row are taken from those of the original object version, modified by the specified new values.

9.2.6 Delete Object Service

Function

This service deletes the current object version of the specified cursor.

Format

```
procedure    IrdsDeleteObject
    (        CurrSessId  :   IrdsSessId;
             ObjCurId:   IrdsCurId;
         var RetCode     :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjCurId := the object cursor identifier returned by a previous Open Cursor service;

Output

Possible error states are

23	Constraint violation
24	Invalid cursor state
25	Invalid session identifier
42	User does not have appropriate privileges
43	Invalid referential action
53	Content status rule violation
54	No current working set context has been established

General Rules

1 A current single working set context shall have been established, and the cursor identified by ObjCurId shall be updatable and shall be positioned on a row. The object version on which the cursor is positioned shall be within the materialization of the working set specified by the context;

2 Deletion shall not violate any constraint that is currently being enforced;

3 The current working set shall have an IRD content status belonging to the Uncontrolled content status class, or shall be a non-versionable working set;

4 The IRDS User who invokes this service shall have a Delete privilege on the current working set;

5 On successful completion of this service, the cursor shall be positioned before the row following the row deleted. If there is no such row, the cursor shall be positioned after the last row in the cursor.

Operations on the Abstract Data Structure

1 If the object version is one of the superimposed object versions belonging to the current working set, and if the IRDS was opened for IRD level access, the specified row in the appropriate IRD Table is marked as deleted, together with the referenced row in the IRD Object Version table. If the latter was the only row that referenced a particular row in the IRD Object table, that row is also marked as deleted;

2 If the object version is one of the superimposed object versions belonging to the current working set, and if the IRDS was opened for IRD Definition level access, the specified row in the appropriate IRD Definition table is marked as deleted, together with the referenced row in the IRD Object Version table. If the latter was the only row that referenced a particular row in the IRD Object table, that row is also marked as deleted;

3 If the object version exists in the materialization of the working set that is the basis for the current working set, and if the IRDS was opened for IRD level access, then a row with the same IRD Object key and an IRD Working Set key referencing the current working set, and marked as deleted, is inserted in the appropriate IRD table, and a matching row marked as deleted is inserted in the IRD Object Version table, with a reference from the first row inserted;

4 If the object version exists in the materialization of the working set that is the basis for the current working set, and if the IRDS was opened for IRD Definition level access, then a row with the same IRD Object key and an IRD Working Set key referencing the current working set, and marked as deleted, is inserted in the appropriate IRD Definition table, and a matching row marked as deleted is inserted in the IRD Object Version table, with a reference from the first row inserted.

5 If the table containing the row to be deleted is a supertable then each row of a direct or indirect subtable that has the same primary key as the subject row is marked as deleted in accordance with operations 1 and 3 above;

6 If the table containing the row to be deleted is a subtable then each row of a direct or indirect supertable that has the same primary key as the subject row is marked as deleted in accordance with operations 1 and 3 above.

9.2.7 Declassify Object Service

Function

This service reduces the degree of classification of the current object version of the specified cursor by deleting the current row of the specified subtable and related rows in any further subtables.

Format

```
procedure    IrdsDeclassifyObject
    (        CurrSessId   :   IrdsSessId;
             ObjCurId:    IrdsCurId;
        var  RetCode      :   IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjCurId := the object cursor identifier returned by a previous Open Cursor service;

Output

Possible error states are

08	Current row must be in a subtable
23	Constraint violation
24	Invalid cursor state
25	Invalid session identifier
42	User does not have appropriate privileges
43	Invalid referential action
53	Content status rule violation
54	No current working set context has been established

General Rules

1 A current IRD level single working set context shall have been established, and the cursor identified by ObjCurId shall be updatable and shall be positioned on a row. The object version on which the cursor is positioned shall be within the materialization of the working set specified by the context;

2 Deletion shall not violate any constraint that is currently being enforced;

3 The current working set shall have an IRD content status belonging to the Uncontrolled content status class, or shall be a non-versionable working set;

4 The IRDS User who invokes this service shall have a Delete privilege on the current working set;

5 The current row of the specified cursor shall be in a table that is a subtable;

6 On successful completion of this service, the cursor shall be positioned before the row following the row deleted. If there is no such row, the cursor shall be positioned after the last row in the cursor.

Operations on the Abstract Data Structure

1 If the object version is one of the superimposed object versions belonging to the current working set, the specified row in the appropriate IRD Table is marked as deleted. The referenced rows in the IRD Object Version and IRD Object tables are not marked as deleted, since there will still be a row in some supertable with the same primary key;

2 If the object version exists in the materialization of the working set that is the basis for the current working set, a row with the same IRD Object key and an IRD Working Set key referencing the current working set, and marked as deleted is inserted in the appropriate IRD table, and a matching row marked as deleted is inserted in the IRD Object Version table, with a reference from the subject row inserted;

3 If the table containing the row to be deleted is a supertable then each row of a direct or indirect subtable that has the same primary key as the subject row is marked as deleted in accordance with operations 1 and 2 above.

9.2.8 Reclassify Object Service

Function

This service increases the degree of classification of the current object of the specified cursor by inserting rows representing the object version in the specified subtable and in any supertables of that subtable up the inheritance hierarchy as far as the table containing the current row.

Format

```
procedure   IrdsReclassifyObject
    (       CurrSessId  :    IrdsSessId;
            ObjCurId:   IrdsCurId;
            NewObjType :    IrdsName;
            NewCols     :    IrdsColList;
        var RetCode     :    IrdsRetCode);
```

Input

The following variables shall be set:

CurrSessId := the session identifier returned by a previous Open IRDS service;

ObjCurId := the object cursor identifier returned by a previous Open Cursor service;

NewCols.NumCols := the number of columns specified in the column list;

NewCols.ColName := the names of the columns whose values are to be inserted in the new rows; as many names shall be specified as indicated by the NumCols parameter.

NewCols.ColNull := a null indicator for each specified column: true if the specified column is to be set to a null value, false if a non-null value is provided.

NewCols.ColType := the types of the specified columns, indicating the variant in which each new value has been placed.

NewCols.ColValxxx := the new values of the specified columns, each in the variant corresponding to its stated data type.

Output

Possible error states are

04	No value supplied for required column
07	Invalid column name
09	Table containing current row does not have specified table as a subtable
22	Data exception
23	Constraint violation
24	Invalid cursor state
25	Invalid session identifier
42	User does not have appropriate privileges
43	Invalid referential action
54	No current working set context has been established
55	A value has been specified for a column which is system-maintained

General Rules

1 A current IRD level single working set context shall have been established, and the cursor referenced by ObjCurId shall be updatable and shall be positioned on a row. The object version on which the cursor is positioned shall be within the materialization of the working set specified by the context;

2 A non-null value shall be supplied for any column in any row to be inserted that is defined as not null;

3 No column that is defined as system-maintained shall be included in the column list;

4 The IRDS User who invokes this service shall have an Update privilege on the current working set;

5 NewObjType shall specify the name of an IRD table in the current IRD;

6 The current row shall be in a table that is a supertable of the table named by NewObjType;