

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)



Documents

Document Sharing Hub

Table of Contents

Foreword	vi
Introduction	vii
1 Scope	1
2 Normative references	1
3 Definitions and abbreviations	2
3.1 Terms defined or referenced in the IRDS Framework (ISO/IEC 10027) and used in this International Standard	2
3.2 Terms defined in this International Standard	2
3.3 Data Item Name abbreviations	4
4 Conventions	5
4.1 Specification of concepts and facilities	5
4.2 Specification of data structures	5
4.3 Specification of constraints - overview	5
4.4 Specification of service data structures	5
4.5 Specification of services	5
4.6 Data Structure Diagrams	5
4.7 Specification of constraints - detail	5
4.7.1 Types of constraint	5
4.7.2 Overview of referential constraints	5
4.7.3 Optional one-to-many referential constraint	6
4.7.4 Required uni-directional one-to-many referential constraint	7
4.7.5 Required uni-directional one-to-one referential constraint	7
4.7.6 Self-referencing tables	7
4.7.7 Required bi-directional referential constraint	7
4.7.8 Mutually-exclusive referential constraints	8
4.7.9 Subtables	9
4.7.10 Principles for expressing constraints	10
4.8 Working Set Diagrams	10
5 IRDS concepts and facilities	11
5.1 IRDS Environment concepts	11
5.2 Categories of table	11
5.3 Overview of IRD Definition tables	12

© ISO/IEC 1993

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

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

Printed in Switzerland

5.4	Overview of IRD tables	15
5.4.1	Overview	15
5.4.2	Internal and common tables	17
5.4.3	IRD-specific tables	17
5.5	Data and the objects to which the data refers	17
5.5.1	Definition objects comprising data modelling facility	17
5.5.2	Definition objects dependent on an IRD Schema Group	18
5.5.3	Content of IRD tables	18
5.5.4	Accessibility of tables to users	18
5.6	Version Control concepts	18
5.6.1	Objects and Versions of Objects	18
5.6.2	Working Sets	18
5.6.3	Working sets and users	19
5.6.4	Basing one working set on another	19
5.6.5	Materialization of a working set	20
5.6.6	References from one working set to another	20
5.6.7	References to multiple versions of an object	22
5.6.8	Context	22
5.6.9	IRD content status	23
5.6.10	References in the IRD	23
5.6.11	Granularity of Version Control	23
5.6.12	Access control	23
5.7	Naming facilities	23
5.7.1	Names	23
5.7.2	IRDS names	24
5.7.3	Variation name	24
5.7.4	Working set name and working set version name	24
5.8	Definable limits and installation defaults	24
5.8.1	Implementation-defined limits	24
5.8.2	Installation defaults	24
5.9	Creating and dropping IRDs	24
5.10	IRD schema modification	24
5.11	Other added value functionality	24
5.11.1	Audit attributes	24
5.11.2	IRDS content modules	25
5.11.3	System-maintained values	25
6	Abstract data structures	27
6.1	IRD Definition Level	27
6.1.1	IRD Definition Level data structure	27
6.1.2	IRD Definition Level Schema	27
6.1.2.1	Schema IRD Definition	27
6.1.3	IRD Definition Level Domains	27
6.1.3.1	Domain SQL Name	27
6.1.3.2	Domain IRDS Key	28
6.1.3.3	Domain Char Data	28
6.1.3.4	Domain Cardinal	28
6.1.3.5	Domain Boolean	29
6.1.4	IRD Definition Level Tables	29
6.1.4.1	Table IRD Object	29
6.1.4.2	Table IRD Working Set	30
6.1.4.3	Table IRD Object Version	31
6.1.4.4	Table IRD Reference Path	33
6.1.4.5	Table IRDS User	33
6.1.4.6	Table Implementation Limits	34
6.1.4.7	Table IRDS Dictionary	35
6.1.4.8	Table IRD Schema Group	36
6.1.4.9	Table IRD Schema	37
6.1.4.10	Table IRD Schema Reference	37
6.1.4.11	Table IRD Data Type Descriptor	38
6.1.4.12	Table IRD Domain	39

6.1.4.13 Table IRD Table	41
6.1.4.14 Table IRD View	42
6.1.4.15 Table IRD Column	43
6.1.4.16 Table IRD View Table Usage	45
6.1.4.17 Table IRD View Column Usage	46
6.1.4.18 Table IRD Table Constraint	47
6.1.4.19 Table IRD Key Column Usage	48
6.1.4.20 Table IRD Referential Constraint	50
6.1.4.21 Table IRD Check Constraint	51
6.1.4.22 Table IRD Check Table Usage	53
6.1.4.23 Table IRD Check Column Usage	54
6.1.4.24 Table IRD Assertion	54
6.1.4.25 Table IRD Module	55
6.1.4.26 Table IRD Content Status	56
6.1.4.27 Table Installation Default	57
6.1.4.28 Table IRD Working Set Privilege	57
6.1.5 IRD Definition Level Views	59
6.1.5.1 View All SQL Names	59
6.1.5.2 View IRD Object Version	60
6.1.5.3 View IRD Working Set	60
6.1.5.4 View IRD Reference Path	60
6.1.6 IRD Definition Level Change Control	61
6.1.7 IRD Definition Level Initial Contents	61
6.2 IRD Level	63
6.2.1 IRD Level data structure	63
6.2.2 IRD Level Initial Contents	63
6.3 IRD General Rules	65
6.3.1 Use of primary key	65
6.3.2 References and content status	66
6.3.3 Resolution of references	66
6.3.4 Resolution of references within a version path	66
6.3.5 References depending on a reference path	66
6.3.6 Reference paths and version paths	66
7 Services concepts and facilities	67
7.1 Levels and parallelism	67
7.2 Access to IRDS data via Database Services Processor	67
7.2.1 Prevention of circumvention of IRDS security and integrity	67
7.2.2 Access to IRDS Data using a standard Database Language	67
7.3 Connecting an application to the IRDS Services Interface Processor	67
7.3.1 Sessions and transactions	67
7.3.2 IRDS users and privileges	68
7.4 Object selection	68
7.5 Sets and cursors	68
7.6 Diagnostics	68
7.7 Version control	69
7.8 Operations on Abstract Data Structures	69
8 Service data structures	71
8.1 Basic data constants	71
8.1.1 Name Length Limits	71
8.1.2 Attribute Length Limits	71
8.1.3 Control Identifier Length Limits	71
8.1.4 Data Types	71
8.1.5 IRD Content Status Classes	72
8.1.6 Close Type parameter	72
8.2 Service data types	72

8.2.1	Column data types	72
8.2.2	Object Names	73
8.2.3	Control Identifiers	73
8.2.4	Diagnostics Area	73
8.2.5	Service Return Code	73
8.2.6	Column List Parameters	74
9	Service Formats and Descriptions	75
9.1	Operational services	75
9.1.1	Create IRD Definition Service	75
9.1.2	Drop IRD Definition Service	75
9.1.3	Open IRDS Service	76
9.1.4	Prepare Service	77
9.1.5	Commit Service	77
9.1.6	Rollback Service	78
9.1.7	Close IRDS Service	78
9.1.8	Get Diagnostics Service	78
9.2	Level independent services	79
9.2.1	Set Context Service	79
9.2.2	Add Object Service	79
9.2.3	Open Cursor Service	81
9.2.4	Retrieve Object Service	82
9.2.5	Modify Object Service	83
9.2.6	Delete Object Service	84
9.2.7	Declassify Object Service	85
9.2.8	Reclassify Object Service	86
9.2.9	Close Cursor Service	87
9.2.10	Create Working Set Service	87
9.2.11	Drop Working Set Service	88
9.2.12	Modify Content Status Service	89
9.2.13	Create Reference Path Service	90
9.2.14	Modify Reference Path Service	90
9.2.15	Drop Reference Path Service	91
9.3	IRD Definition Level specific services	92
9.3.1	Create IRD Service	92
9.3.2	Drop IRD Service	92
9.3.3	Deactivate IRD Service	93
9.3.4	Reactivate IRD Service	93
9.3.5	Validate IRD Schema Group Service	94
9.4	Sequence of permitted service invocation	95
9.4.1	Specification of valid sequences of IRDS service invocations	95
9.4.2	General rules	95
10	Conformance	97
Annexes		
A	State classes and subclasses	99
A.1	State classes	99
A.2	State subclasses	100
A.3	State record	101
B	User-defined tables	103

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.

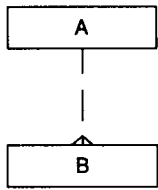


Figure 1 - Optional referential constraint

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.

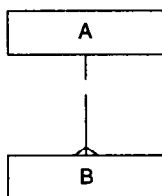


Figure 2 - Required referential constraint (one to many)

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

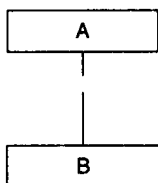


Figure 3 - Required referential constraint (one to one)

Figure 3 illustrates a required uni-directional one-to-one referential constraint. The removal of the crow's 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

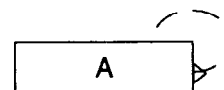


Figure 4 - Self-referencing constraint

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.

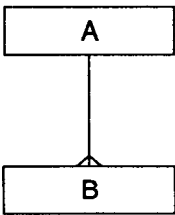


Figure 5 - Required bi-directional referential constraint

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.

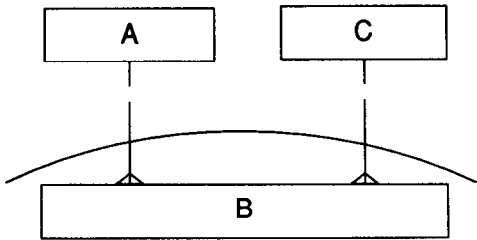


Figure 6 - Type 1 mutually-exclusive constraints

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.

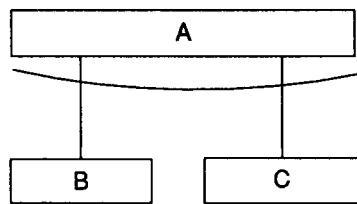


Figure 7 - Type 2 mutually-exclusive constraints

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.

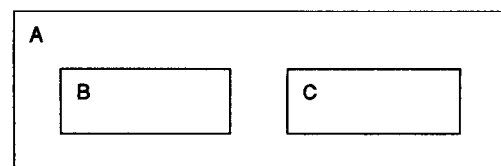


Figure 8 - Subtables

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.

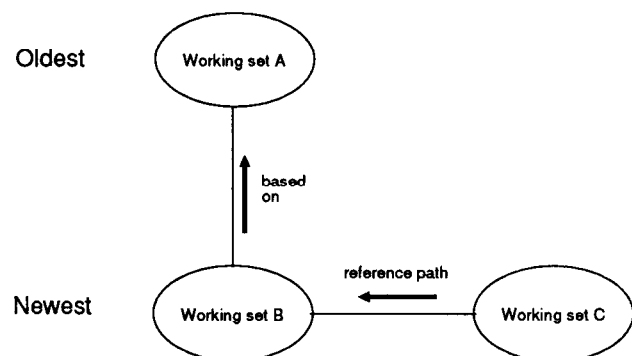


Figure 9 - Working set diagram conventions

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.

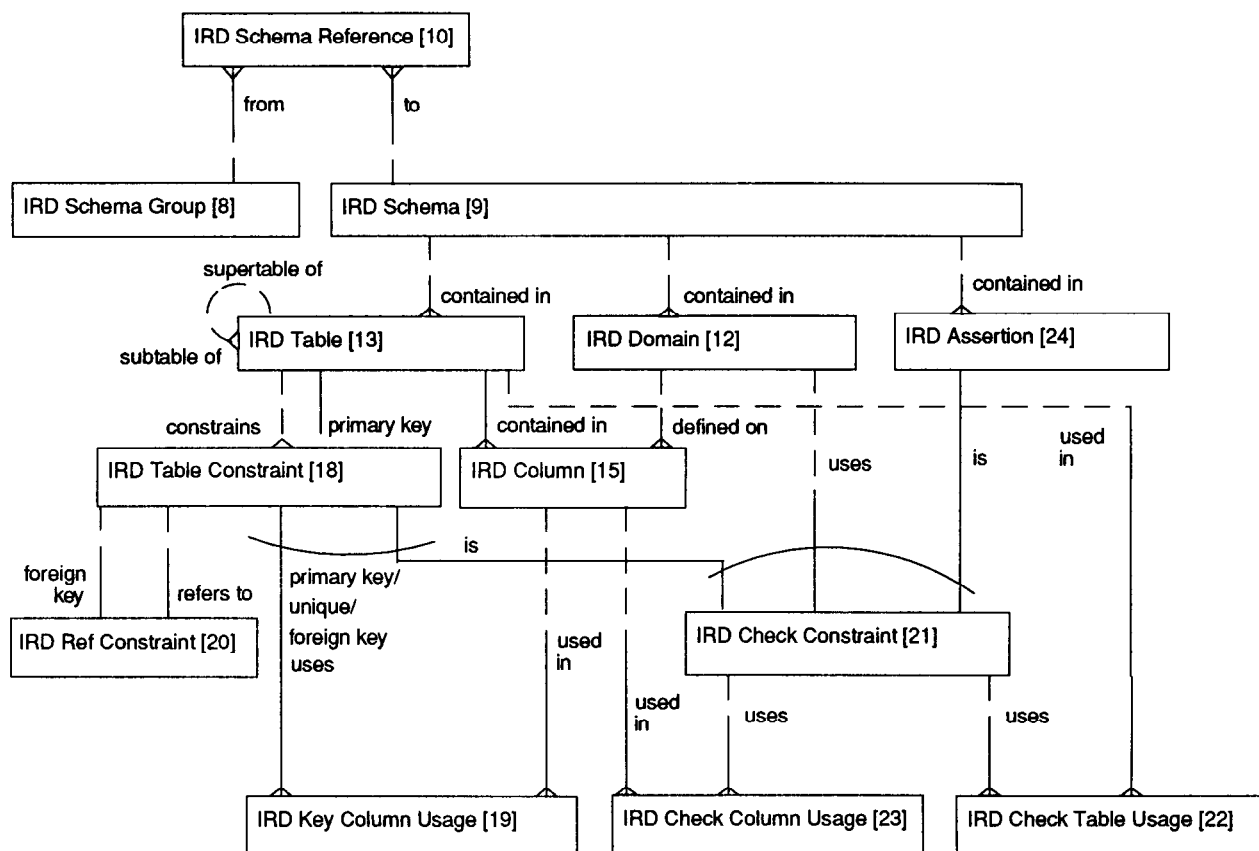


Figure 10 - Structure diagram for the IRD Definition Level

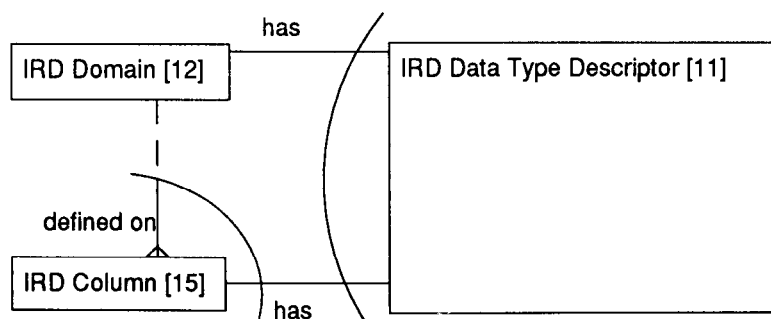


Figure 11 - Use of Data Type Descriptors

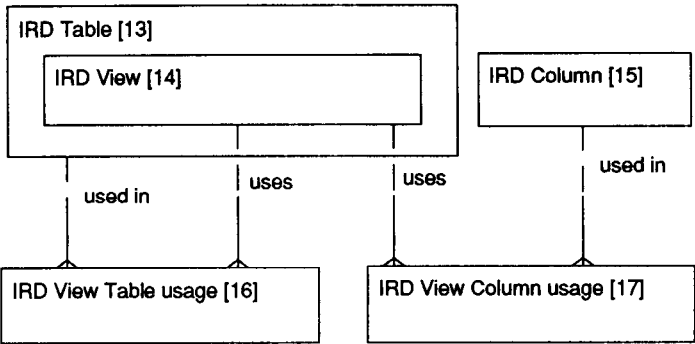


Figure 12 - IRD View usages

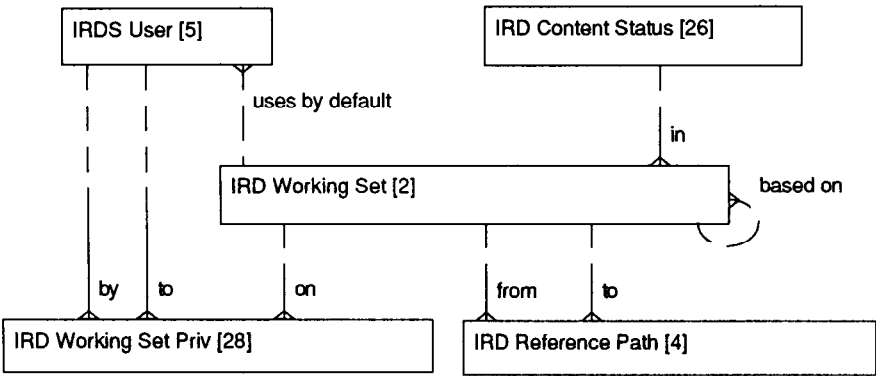


Figure 13 - IRD Definition Level Version Control tables

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.

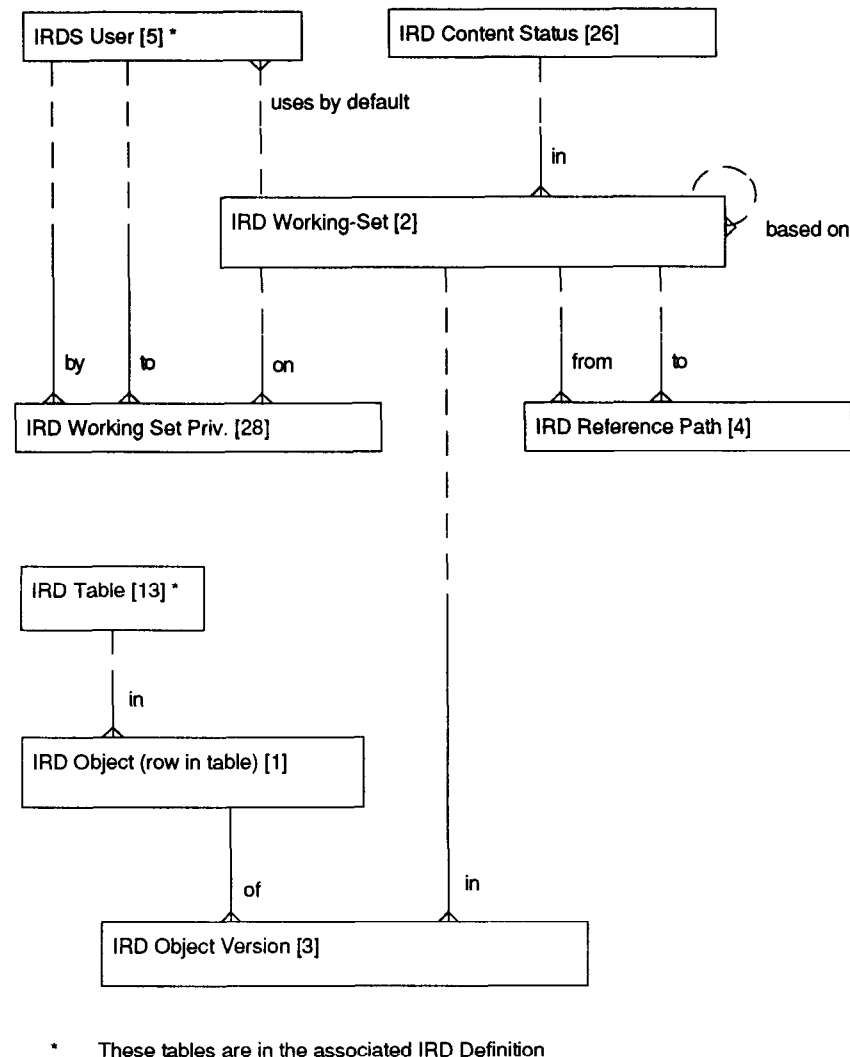


Figure 14 - Tables on the IRD Level

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:

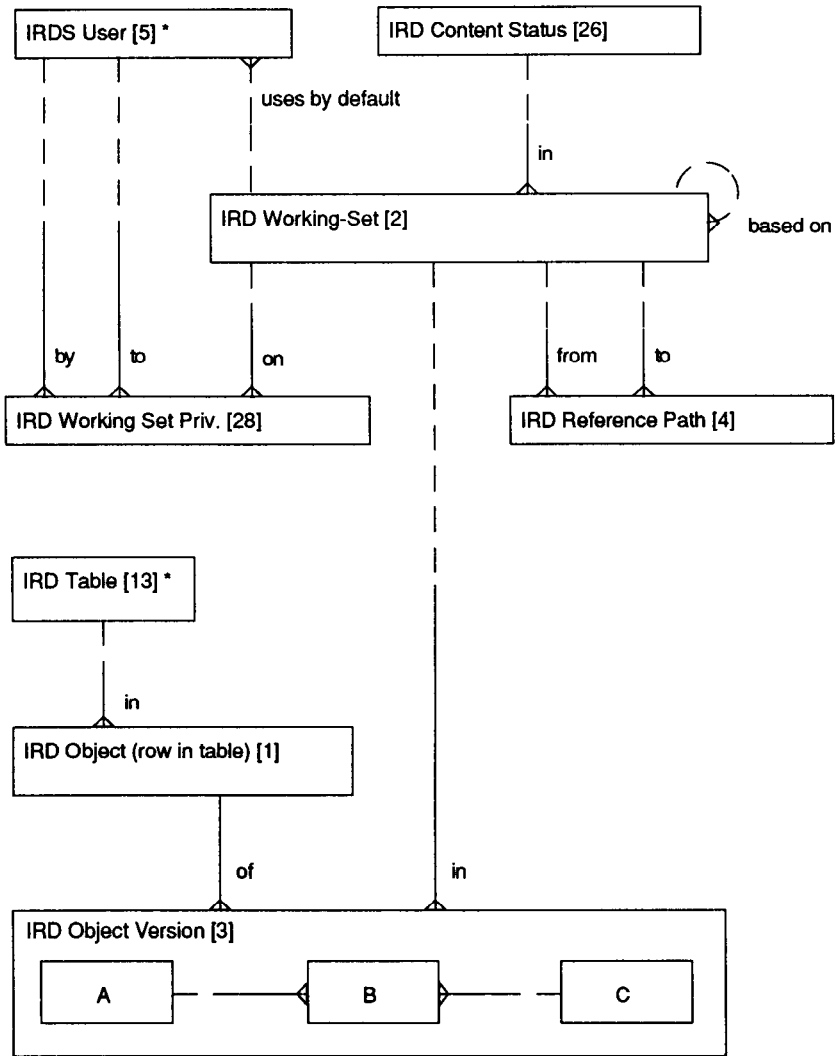
- Each IRD is described by an IRD Schema Group. Through rows in the IRD Schema Reference table, that IRD Schema Group references one or more IRD Schemas that provide the detailed definition of the IRD.

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.

- 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;
- 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.

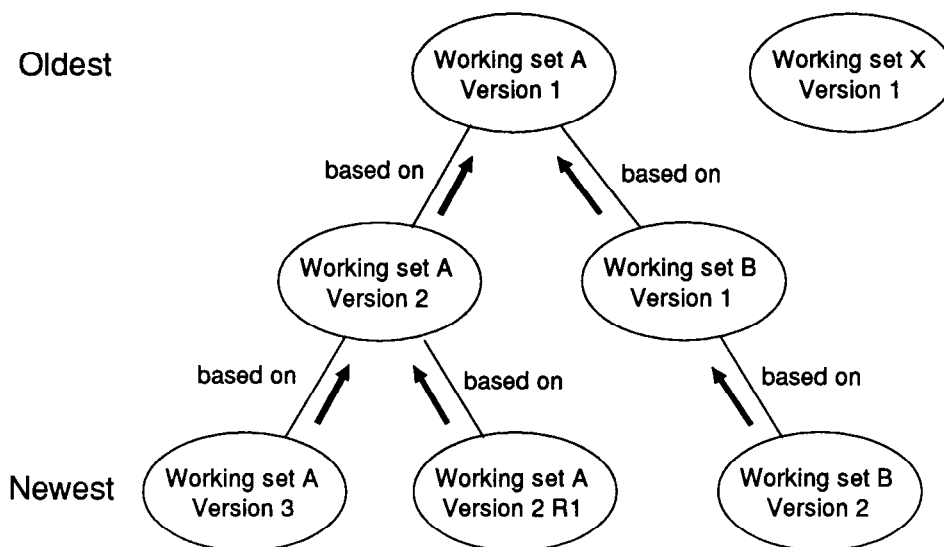


Figure 17 - Basing one working set on another

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 $(WS_n, \dots, WS_1)$ where each working set WS_j ($n \geq j > 1$) 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.

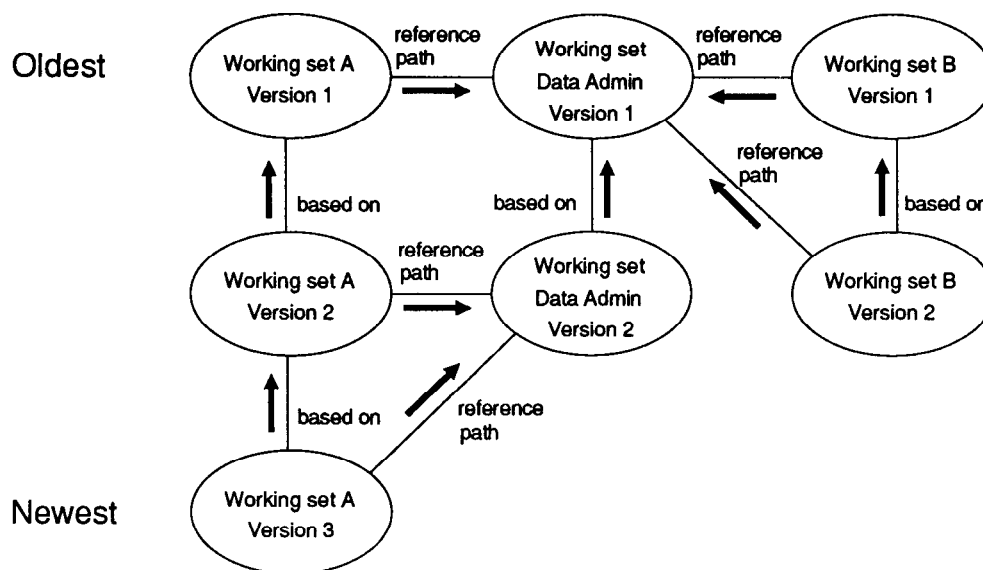


Figure 18 - Example references between working sets

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 reference 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.

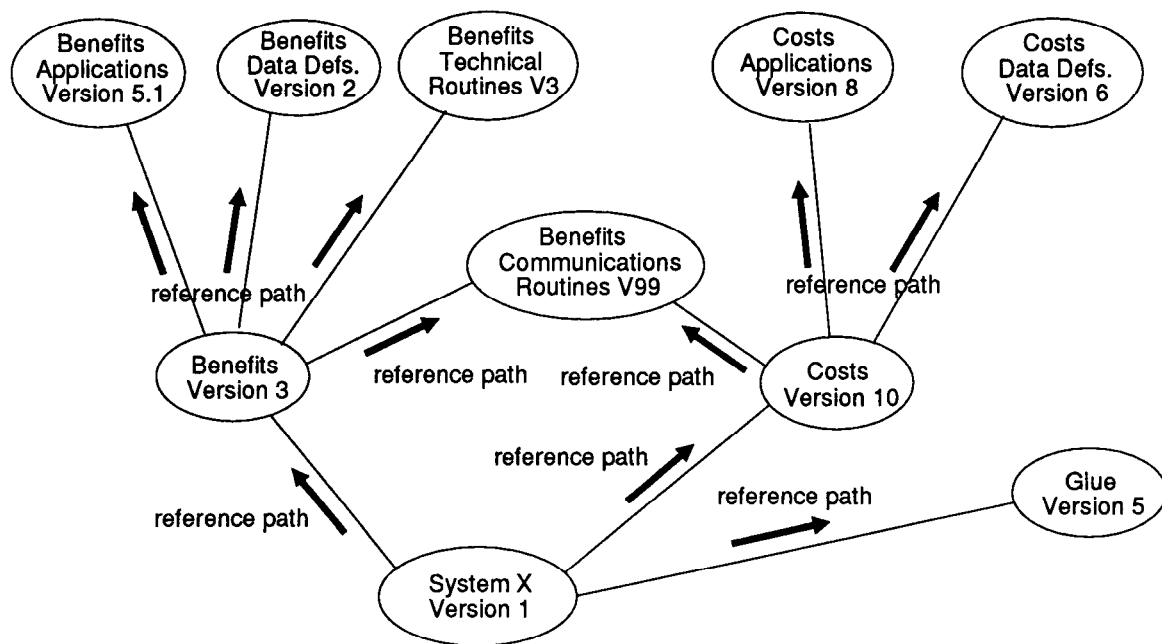


Figure 19 - Building an aggregate using reference paths

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)