

ISO/IEC JTC 1/SC 32 N 0247

Date: 1999-04-13

REPLACES: --

<p style="text-align: center;">ISO/IEC JTC 1/SC 32</p> <p style="text-align: center;">Data Management and Interchange</p> <p style="text-align: center;">Secretariat: United States of America (ANSI)</p> <p style="text-align: center;">Administered by Pacific Northwest National Laboratory on behalf of ANSI</p>
--

DOCUMENT TYPE	Text for CD ballot or comment
TITLE	ISO/IEC CD 9579 Information technology - Remote database access for SQL (RDA/SQL) - Support for SQL 3
SOURCE	John Hadjioannou – Project Editor
PROJECT NUMBER	31.05.00.00.00
STATUS	It is not yet known whether this work will ultimately be progressed as (a) an amendment to 9579, (b) a new edition of 9579, or (c) a new independent standard. This text has been prepared as a stand-alone document to ease review and progression.
REFERENCES	
ACTION ID.	COM
REQUESTED ACTION	Review for Comments
DUE DATE	
Number of Pages	79
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile: +1 703 681 9180; E-mail: MannD@battelle.org

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

Title: Proposed CD ISO/IEC SSSS:200y (E)
Information technology — Distributed Database Access for SQL

Source: UK

Status: UK Proposal

Date: 11 March 1999

Editor's Preface:

This text was initially prepared under the BSI/DTI Consultancy Drafting Scheme project SP6/3.1(355), Distribution Schema for RDA in January 1999.

The background for this work is in:

RDA-KAN-23 (WG3 N2012) Remote Database Access – Strategic Directions

Document Conventions

Informative Paragraphs

This document contains informative paragraphs that serve to clarify aspects of this International Standard, or which provide information about flexibility open to implementers of the protocol defined by this International Standard. Such paragraphs are introduced by the word “NOTE” followed by a number identifying the paragraph and are set out as in the following example:

NOTE 1 – This is an example of an informative note.

Discussion Paragraphs

This document contains informative paragraphs that serve to explain why certain design decisions have been taken to facilitate review by experts involved in the development of this International Standard. Such paragraphs are introduced by the word “DISCUSSION” followed by a number identifying the paragraph and are set out as in the following example:

DISCUSSION 1 – This is an example of a discussion note.

Discussion paragraphs will be deleted from this document prior to publication as an International Standard.

Incomplete Marker

This document contains markers that indicate that additional text is required, or existing text may need to be modified, prior to completion of this document.

Such markers appear with a surrounding box and a grey background as in the example below:

Example of Incomplete Marker

Incomplete markers will be satisfied and removed from this document prior to publication.

Editor:

*John Hadjioannou
Blue Star Information Systems Ltd
197 Grasmere Way
Linslade
Leighton Buzzard
LU7 7QB*

Tel: +44 1525 374667

Fax: +44 1525 850427

Email: *john@minster.co.uk*

NOTE TO REVIEWERS

Both the SC32 project relating to this work calls for the production of an Amendment.

It is not yet known whether this work will ultimately be progressed as (a) an amendment to 9579, (b) a new edition of 9579, or (c) a new independent standard.

This text has been prepared as a stand-alone document to ease review and progression.

PROPOSED CD
INTERNATIONAL
STANDARD

ISO/IEC
SSSS

First Edition
200y-xx-xx

**Information technology —
Distributed Database Access for SQL**

Version: Proposed CD
Standard: ISO/IEC SSSS:200y (E)
Date: 11 March 1999
Copyright: © BSI



Reference number
ISO/IEC SSSS:200y (E)

Contents

Contents ii

Tables v

Figures vi

Foreword	vii
Introduction.....	viii
1 Scope.....	1
2 Normative References.....	3
3 Definitions, Conventions and Notations	4
3.1 Definitions	4
3.2 Conventions.....	5
3.2.1 Convention for Figures	5
3.2.2 Naming of Concepts	5
3.2.3 Specification of DDA elements.....	5
3.2.4 Evaluation of Rules	5
3.3 Notations	6
4 Model, Concepts and Facilities	7
4.1 Model	7
4.1.1 Processor Model.....	7
4.1.2 Privilege Model.....	8
4.1.3 Transaction Model.....	8
4.1.4 Security Model.....	9
4.2 DDA concepts.....	10
4.2.1 DDA-distribution Schema.....	10
4.2.2 DDA-distribution Statements.....	10

4.2.3	DDA-information Schema	10
4.2.4	DDA-services	10
4.2.5	DDA-mapped Tables	11
4.2.6	Limitations and Requirements	11
4.3	The mapping of SQL concepts	13
4.3.1	SQL Identifiers	13
4.3.2	SQL User Defined Types	13
4.3.3	SQL Routines	13
5	DDA-information Schema	14
5.1	DISTRIBUTED_INFORMATION_SCHEMA Schema	15
5.2	DDA-INFORMATION SCHEMA Domains	16
5.3	DDA-INFORMATION SCHEMA Tables	17
6	DDA-distribution Schema.....	20
6.1	CHARACTER_SETS base table	21
6.2	COLLATIONS base table	23
6.3	COLUMNS base table	24
6.4	FRAGMENT TABLE base table.....	27
6.5	REPLICATED FRAGMENT base table.....	28
6.6	ROLE_AUTHORIZATION_DESCRIPTORs base table	30
6.7	ROLES base table.....	31
6.8	SCHEMATA base table	32
6.9	SERVER AUTHENTICATION base table.....	33
6.10	SERVER LOCATION base table.....	34
6.11	SQL_FEATURES base table.....	35
6.12	SQL_IMPLEMENTATION_INFO base table.....	37
6.13	SQL_SIZING base table	38
6.14	SQL_SIZING_PROFILES base table.....	39
6.15	SQL_LANGUAGES base table.....	40
6.16	TABLES base table.....	45
6.17	USERS base table	47
6.18	VIEWS base table.....	48
6.19	VIEW_COLUMN_USAGE base table.....	50
6.20	VIEW_TABLE_USAGE base table	51

7	DDA-distribution Statements	52
7.1	General constructs	52
7.1.1	Expressions	52
7.2	DDA-distribution Schema Definition	53
7.3	DDA-distribution Schema Elements.....	55
7.3.1	Server Location Definition.....	55
7.3.2	Server Authentication	55
7.3.3	Mapped Table Definition.....	56
8	Exceptions	58
8.1	Exception codes for DDA-specific Conditions	58
9	Conformance.....	59
9.1	DDA Conformance	59
9.2	Claims of Conformance	59
Annex A	Conformance Proforma.....	61
A.1.	Identification.....	61
A.2.	Supplier Details	61
A.3.	Implementation Details	62
A.4.	DDA Support.....	62
Annex B	Extensions to RDA.....	63
Annex C	Examples of DDA architecture.....	65
C.1.	Local SQL-servers	65
C.2.	Remote SQL-servers.....	65
C.3.	Remote SQL-servers.....	66

Tables

Table 1– DDA-information Schema Tables	17
Table 2–SQLSTATE class and subclass values for DDA-specific conditions.....	58

Figures

Figure 1—Model of SQL-environment.....	7
Figure 2—Model of DDA-environment.....	7
Figure 3—Model of a DDA-server	8
Figure 4—Local SQL-servers.....	65
Figure 5—Remote SQL-servers	65
Figure 6—Remote DDA-services	66

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 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 SSSS was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This is the first edition of an International Standard addressing the Management and Operation of facilities to provide access to SQL-data, independent of its actual location. It addresses the requirements of geographically distributed enterprises that need to both maintain a level of local autonomy and provide a simple interface to multiple SQL-implementations managing SQL-data.

Annexes A to B form an integral part of this International Standard. Annex C is for information only.

Introduction

The Distributed Database Access for SQL International Standard is complementary to the standards for Database Language SQL and Remote Database Access for SQL. It addresses the need for access to a geographically dispersed set of SQL-implementations which may be under independent administration.

Distributed Database Access for SQL defines how multiple SQL-implementations, each providing selected SQL-data can be regarded as a composite SQL-implementation. The component SQL-data may be:

- at different sites,
- on different hardware systems,
- managed by different Database Management Systems,
- managed by their own Database Administrators.

Distributed Database Access for SQL supports the Database Language SQL and uses the Remote Database Access for SQL protocol for system interworking. Extensions to these that are required to support Distributed Database Access are defined in this International Standard.

Information technology — Distributed Database Access for SQL

1 Scope

This International Standard, Distributed Database Access for SQL (DDA), defines a model for the architecture and operation of a geographically distributed data management facility where each site contains SQL-implementations supporting SQL-catalogs, SQL-schemas and SQL-data. The sites are connected by communications facilities that support the RDA/SQL protocol carried over a Transport Layer.

The architecture identifies a functional Distribution Controller, which provides data location independence and interface to SQL-data such that execution of a single SQL-statement by the user may involve manipulation of data from more than one SQL-implementation. The operation of the Distribution Controller is determined by a Distribution Schema, the DDA-distribution Schema, which identifies the SQL-data available from each participating SQL-implementation and specifies the mapping of this data to a single SQL-cluster of Catalogs.

The user's view of the data and facilities available is defined in a distributed information schema, the DDA-information Schema. This has the same form as the SQL-information Schema but the information provided is derived from the Distribution Schema and the SQL-information Schemas of the component SQL-implementations.

This International Standard relies upon the facilities provided by ISO/IEC 9075 (SQL), ISO/IEC 9075-3 (SQL/CLI) and ISO/IEC 9579 (RDA).

This International Standard defines:

- a model for Distributed Database Access,
- the DDA-information Schema as an SQL-information Schema derived from the SQL-information Schemas of participating SQL-implementations
- the DDA-distribution Schema,
- DDA-distribution Statements, specified as extensions to SQL, for defining and maintaining the DDA-distribution Schema

Normative annexes provide:

- a Conformance Proforma,
- amendments enhancing ISO/IEC 9579 (RDA)

DISCUSSION 1 – These amendments may be processed separately and removed from this document.

Informative annexes provide:

- a collection of examples of architectures that conform to this International Standard

This International Standard does not constrain:

- conforming implementations to use any particular processor decomposition,

This International Standard does not define:

- algorithms for query decomposition or for the combining of partial results,

1.1

- recovery mechanisms in the event that transaction co-ordination fails,
- mechanisms for the asynchronous replication of data

NOTE 1 – A new reader may wish to read clauses 4, 7, 5, and 6 in that sequence

2 Normative References

The following International 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 ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9075-1:199x *Information technology – Database Languages SQL*

ISO/IEC 9075-3:199x *Information technology – Database Languages SQL – Part 3: Call Level Interface*

ISO/IEC 9579:199x *Information technology – Remote Database Access for SQL*

ISO/IEC 10032:1995 *Information technology – Reference Model of Data Management*

3.1 Definitions

3 Definitions, Conventions and Notations

3.1 Definitions

For the purposes of this International Standard, the definitions given in the following International Standards apply:

- ISO/IEC 9075 (SQL),
- ISO/IEC 9075-3 (SQL/CLI),
- ISO/IEC 9579 (RDA).

The following terms are defined in ISO/IEC 10032:

- Distribution Controller,
- Distribution Schema.

In addition, the following definitions apply:

Distributed Database: an SQL-implementation that presents the appearance of a single SQL-implementation whilst deriving its SQL-data from other SQL-implementations according to a Distribution Schema.

NOTE 2 – The term “Database” without qualification is not used in this International Standard since it is so commonplace as to be ambiguous.

3.2 Conventions

3.2.1 Convention for Figures

The convention used for figures is that defined in the Reference Model of Data Management, ISO/IEC 10032.

3.2.2 Naming of Concepts

Where a concept has been defined in ISO/IEC 9075 (SQL) and used in this International Standard, the name used in ISO/IEC 9075 (SQL) is used for that concept.

Where a concept has been defined in ISO/IEC 9579 (RDA) and used in this International Standard, the name used in ISO/IEC 9579 (RDA) is used for that concept.

Concepts whose name begins with 'DDA-' are defined in this International Standard. Concepts whose name begins with 'SQL-' are defined in ISO/IEC 9075 (SQL). Concepts whose name begins with 'RDA-' are defined in ISO/IEC 9579 (RDA).

3.2.3 Specification of DDA elements

The definition of each DDA element, depending on the nature of the element, may have the following parts:

- *function*, a short statement of the purpose of the element,
- *definition*, the definition of the element in SQL,
- *syntax*, the syntax of the element in BNF,
- *description*, a sequence of Rules describing the element,
- *syntax rules*, a sequence of Rules as defined in ISO/IEC 9075 (SQL) for Syntax Rules,
- *general rules*, a sequence of Rules as defined in ISO/IEC 9075 (SQL) for General Rules.

3.2.4 Evaluation of Rules

The evaluation of Descriptions, Syntax Rules and General Rules is as defined in ISO/IEC 9075 (SQL)

3.3 Notations

3.3 Notations

DISCUSSION 2 – *There are no notations at present. This subclause is retained as a placemarker.*

4 Model, Concepts and Facilities

4.1 Model

4.1.1 Processor Model

ISO/IEC 9075 (SQL) defines an SQL-environment, SQL-client, SQL-server and SQL-cluster of catalogs. ISO/IEC 9579 (RDA) defines a Service User, RDA-client, RDA-server and Transport Provider. Figure 1 shows how these interrelate.

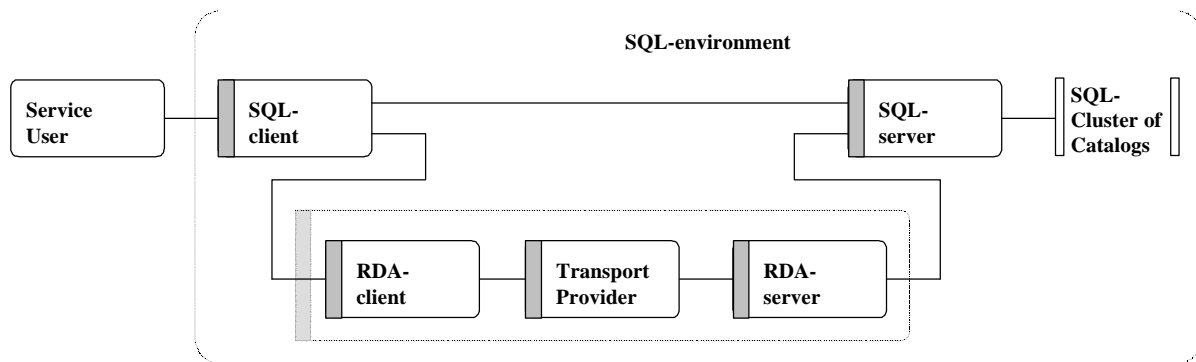


Figure 1 – Model of SQL-environment

This International Standard defines facilities for presenting multiple SQL-servers to a Service Provider in such a way that they appear as a single (composite) SQL-server with a (composite) SQL-cluster of catalogs. These are termed DDA-server and DDA-cluster of catalogs respectively, as shown in Figure 2.

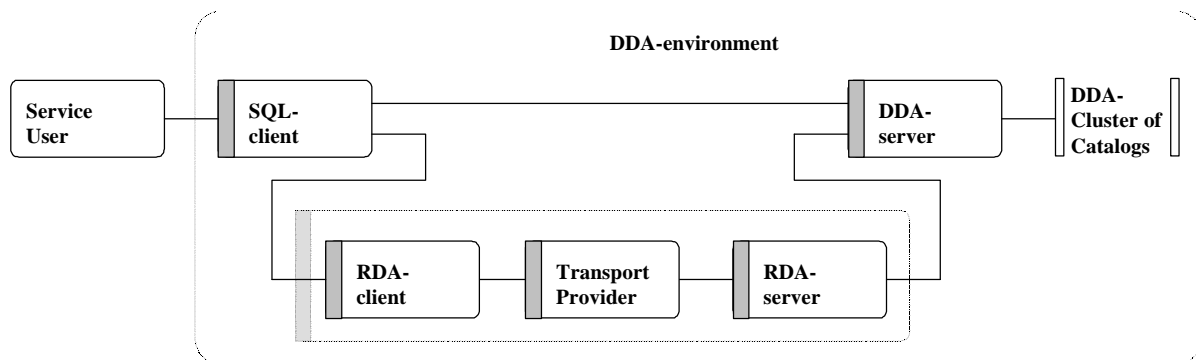


Figure 2 – Model of DDA-environment

4.1 Model

A DDA-server is an SQL-server with an internal composition defined in this International Standard and modelled in Figure 3. Within a DDA-server, DDA-services acts as an SQL-client in respect to one or more component SQL-servers.

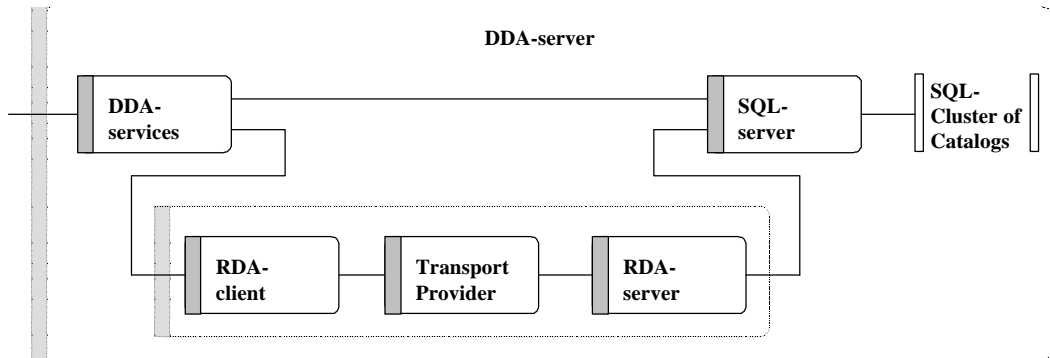


Figure 3 – Model of a DDA-server

An SQL-server included directly in a DDA-server is termed *component SQL-server* where to refer to it as an SQL-server would be ambiguous.

The SQL-server components in Figure 3 may themselves be DDA-servers.

NOTE 3 – Thus, the definition of a DDA-server is recursive. A Distributed Database may therefore be composed of SQL-servers that are themselves Distributed Databases.

4.1.2 Privilege Model

A DDA-server is an SQL-server with the privilege model and mechanisms defined in ISO/IEC 9075 (SQL).

The DDA Privilege Model assumes a database administrator with authority to create users and roles and to grant them privileges, in the same way as for SQL.

The DDA Privilege Model relates schema definition and ownership in the Distributed Database environment to the privilege models for the SQL Server on each site.

The Distributed Database database administrator is the owner of the DDA-information Schema and the DDA-distribution Schema. Users of a DDA-server are identified by their `auth_ids`. The database administrator may define roles and users may be privileged to play these roles.

Each component SQL-server has its own database administrator and its own privilege control. When an SQL-session is established with a component SQL-server, authorization information provided by the DDA-server is used. This gives access to public objects and to objects accessible for that authorization.

4.1.3 Transaction Model

DISCUSSION 3 – Nothing beyond SQL is needed at present. This subclause is retained as a placemaker.

NOTE 4 – Because of the semantics of SQL-transactions, all component SQL-implementations that participate in the evaluation of an SQL-statement by a DDA-server must be available for the evaluation to complete successfully.

4.1.4 Security Model

There is assumed to be total mutual trust either between the service user and DDA-services, or between DDA-services and the component SQL-servers.

DISCUSSION 4 – A detailed definition of how SQL-authorisation and RDA-authentication information is used across multiple connections defining how the authentication and access control mechanisms can take into account both the Service User and the DDA-services could be developed. This would enable the trust requirement to be relaxed.

This enhancement to the security model could be provided as an amendment to this International Standard in the same way as was done for RDA3.

4.2 DDA concepts

4.2 DDA concepts

4.2.1 DDA-distribution Schema

The DDA-cluster of catalogs of a DDA-server is derived from the SQL-cluster of catalogs of the component SQL-servers according to a mapping defined by data defined by an SQL-schema available to DDA-services termed the DDA-distribution Schema.

The DDA-distribution Schema describes data which:

- Identifies the location of component SQL-servers by reference to Transport Mapping and Transport Address,
- Identifies characteristics of component SQL-servers,
- Identifies the SQL-server that manages each element of SQL-data defined within the DDA-cluster of catalogs.

NOTE 5 – The DDA-distribution Schema may be itself distributed.

The DDA-distribution Schema is defined in clause 6

4.2.2 DDA-distribution Statements

A Service User can not update the DDA-distribution Schema directly. Statements for defining and maintaining the DDA-distribution Schema are defined in clause 7. These extend ISO/IEC 9075 (SQL).

4.2.3 DDA-information Schema

Each SQL-catalog within a DDA-cluster of catalogs of a DDA-server contains an SQL-information Schema. This SQL-information Schema is derived from the SQL-information Schemas of the component SQL-servers of the DDA-server, and augmented by information to support the DDA-distribution Schema. This DDA-information Schema is defined in clause 5.

NOTE 6 – An implementation conforming to this International Standard need not materialise the DDA-information schema and may derive it as required.

4.2.4 DDA-services

DDA-services map an SQL-session established between an SQL-client and a DDA-server to SQL-sessions between DDA-services and component SQL-servers.

DDA- Services:

- Process the operations that maintain the DDA-distribution Schema,
- Combine the SQL-information Schemas of the component SQL-servers to provide a DDA-information Schema,
- Establish SQL-sessions with component SQL-servers,
- Decompose SQL-statements that address the composite SQL-cluster of Catalogs into SQL-statements and send these to the relevant component SQL-servers,
- Combine the results of component SQL-statements

NOTE 7 – The Decomposition of SQL-statements and the combination of associated results are not further defined by this International Standard.

An implementation of DDA-services may restrict the repertoire of SQL-statements that it may process.

DISCUSSION 5 – It may be appropriate to provide a “profiling” of these for conformance purposes – for example, permitting only <select statement>s may be a commercially useful and viable subset.

DDA-services are an instance of a Distribution Controller as defined in ISO/IEC 10032 (RMDM).

4.2.5 DDA-mapped Tables

DISCUSSION 6 – Is this definition required?

DDA-mapped tables are tables that are constructed from tables on more than one SQL-implementation. The SQL-implementations that contain parts of a mapped table are termed contributing SQL-implementations.

A mapped table may be partitioned into a set of disjoint fragments, such that any given row of the table is in one and only one fragment.

Fragments (or the whole table if the table is not fragmented) may be stored on more than one SQL-implementation, in which case each replicate shall have equivalent rows.

New rows are allocated to fragments according to a sequence of <insert condition>s defined for the fragments.

NOTE 8 – For database integrity there should be check conditions in the contributing SQL-implementation schemas to ensure that rows cannot be inserted locally into the wrong contributing SQL-implementation.

The wording of the above note needs to be reviewed, and perhaps promoted from a note to a requirement.

4.2.6 Limitations and Requirements

DISCUSSION 7 – These need to be kept under review

NOTE 9 – Providing a mapping from an arbitrary schema to a set of underlying dispersed schemas is inherently complex and the most general solution is probably too complex to be economically implemented. This International Standard aims to limit the complexity by specifying constraints that can reasonably be met when the underlying databases have the type of common features that would apply if they belong to a single Enterprise (or association) with data definition standards.

The integrity of the underlying component SQL-servers is managed by their own administrators, i.e. by the owners of the Schemas for the Databases extant on each SQL-implementation.

The local schema owner has full control over access to data and the subset of the data that is available to users of a distributed database.

Data in the distributed database may be derived from data in component databases according to derivation rules.

If Domain Names or UDTs are to be used for the DDB data then the definitions of these must be identical on all SQL-implementations. The DDB Administrator is responsible for ensuring that functions invoked exist and are correct on all the sites where they may be executed.

DISCUSSION 8 – There is a requirement for database statistics but the method for gathering stats is both server and implementation dependent and hence may be different on each site, do we need to include some common mandatory elements or functions?

4.2 DDA concepts

The DDB is not a mechanism for defining data or constraints at remote SQL-implementations, hence certain IS Tables (Assertions, constraints?) exist but are empty?

4.3 The mapping of SQL concepts

4.3.1 SQL Identifiers

Elements defined in ISO/IEC 9075 are referenced using identifiers of the form <catalog name>.<schema name>.<object name>.

Within the DDA-distribution Schema, the SQL-server from which an element derives may need to be identified.

The postfix “@<server name>” may be applied to any identifier which has a <catalog name>. Thus the general form of an identifier is now <catalog name>.<schema name>.<object name>@<server name>.

The <server name> qualifies the <catalog name> and identifies the referenced catalog as one of the cluster of catalogs at the SQL-implementation identified by <server name>.

The <server name> suffix may also be applied when the <catalog name> is absent but implied, for example when a server has only one catalog.

4.3.2 SQL User Defined Types

DISCUSSION 9 – UDTs defined at two sites may be similarly named but have different definitions. In a distributed environment we could either (a) place a requirement on the implementer that UDTs have common definition, or (b) exclude UDTs from distributed operations. Reviewers are asked to comment.

4.3.3 SQL Routines

DISCUSSION 10 – Routines defined at two sites may be similarly named but have different definitions. In a distributed environment we could either (a) place a requirement on the implementer that Routines have common definition, or (b) exclude Routines from distributed operations. Reviewers are asked to comment.

5 DDA-information Schema

The Information Schema for a single SQL-implementation is defined in ISO/IEC 9075 (SQL). That Information Schema, SQL/IS, defines both the content of the Information Schema (e.g. Tables and Columns) and the way that this content is derived from the Definition Schema.

This International Standard defines an Information Schema for the Distributed Database, DDB/IS, with the same Tables, Columns, Constraints and Assertions as the SQL/IS. However the DDB/IS derives its content from:

1. The Information Schema which exists on each contributing SQL Server which consists largely of Views derived from tables defined by the SQL Definition Schema on that SQL_implementation;
2. The DDB tables which are defined by the DDA-distribution Schema. These tables define :-
 - users of the Distributed Database, and their privileges
 - the SQL-implementations that contribute to the Distributed Database
 - optionally, additional views to be provided in the Distributed Database
 - mapped tables, in which rows are mapped to multiple tables in the contributing SQL-implementations.

Following clauses describe components of the DDA-information Schema and their derivation. The base tables are all defined in a <schema definition> for the schema named **DISTRIBUTED_INFORMATION_SCHEMA**. The table definitions are as complete as the definitional power of SQL allows. Table definitions are supplemented with assertions where appropriate.

Each definition consists of three parts:

- The function of the definition is stated.
- The SQL definition of the object is presented.
- An explanation of the object.

The specification provides only a model of the base tables that are required, and does not imply that an implementation shall provide the functionality in the manner described in this clause.

An instance of the Distribution Information Schema describes an instance of a cluster of distribution catalogs.

NOTE 10 – As far as practical, tables are defined by reference to ISO/IEC 9075

5.1 DISTRIBUTED_INFORMATION_SCHEMA Schema

Function

Identify the schema that is to contain the DDA-information Schema tables.

Definition

```
CREATE SCHEMA DISTRIBUTED_INFORMATION_SCHEMA  
AUTHORIZATION DISTRIBUTED_INFORMATION_SCHEMA
```

Conformance Rules

The following restrictions apply for Core SQL:

None.

5.2 DDA-INFORMATION SCHEMA Domains

The following domains are defined in ISO/IEC 9075 and are included identically in the DDA-information Schema.

```
INFORMATION_SCHEMA.CHARACTER_DATA domain  
INFORMATION_SCHEMA.SQL_IDENTIFIER domain  
INFORMATION_SCHEMA.TIME_STAMP domain
```

5.3 DDA-INFORMATION SCHEMA Tables

Table 1 lists the tables that comprise the DDA-information Schema. The structure of each table and Assertions and Domains are defined in IS9075.

The content of the Information_Schema_Catalog_Name table is defined, for each DDA-information Schema. The content of other tables is derived from the DDA-distribution Schema and the Information Schemas of the SQL Servers that contribute to the Distributed Database.

When a <catalog name> is included in a view, the value in the DDA-information Schema is a concatenation of the <catalog name> in the SQL Server Information Schema and the string @<server name>.

Table 1 – DDA-information Schema Tables

Table Name	Table type	Data source	Note
INFORMATION_SCHEMA_CATALOG_NAME	base table	new data	
ADMINISTRABLE_ROLE_AUTHORIZATIONS	view	DDA-distribution Schema	1
APPLICABLE_ROLES	view	DDA-distribution Schema	1
ASSERTIONS	view	Union	
CHARACTER_SETS	view	DDA-distribution Schema	1
CHECK_CONSTRAINTS	view	Union	
COLLATIONS	view	DDA-distribution Schema	1
COLUMN_DOMAIN_USAGE	view	Union	
COLUMN_PRIVILEGES	view	Union	
COLUMN_USER_DEFINED_TYPE_USAGE	view	Union	
COLUMNS	view	Union plus DDA-distribution Schema	4
CONSTRAINT_COLUMN_USAGE	view	Union plus DDA-distribution Schema	4
CONSTRAINT_TABLE_USAGE	view	Union plus DDA-distribution Schema	4
DIRECT_SUPERTABLES	view	Union	
DOMAIN_CONSTRAINTS	view	Union	
DOMAIN_USER_DEFINED_TYPE_USAGE	view	Union	
DOMAINS	view	Union	
ENABLED_ROLES	view	Dynamic	
KEY_COLUMN_USAGE	view	Union	

METHOD_SIGNATURES	view	Union	
METHOD_SIGNATURE_PARAMETERS	view	Union	
PARAMETERS	view	Union	
REFERENTIAL_CONSTRAINTS	view	Union plus DDA-distribution Schema	4
ROLE_COLUMN_GRANTS	view	DDA-distribution Schema	1
ROLE_ROUTINE_GRANTS	view	DDA-distribution Schema	1
ROLE_TABLE_GRANTS	view	DDA-distribution Schema	1
ROLE_USAGE_GRANTS	view	DDA-distribution Schema	1
ROLE_USER_DEFINED_TYPE_GRANTS	view	DDA-distribution Schema	1
ROUTINE_COLUMN_USAGE	view	Union	5
ROUTINE_PRIVILEGES	view	Union	5
ROUTINE_TABLE_USAGE	view	Union	5
ROUTINES	view	Union	5
SCHEMATA	view	DDA-distribution Schema	1
SQL_FEATURES	view	DDA-distribution Schema	2
SQL_IMPLEMENTATION_INFO	view	DDA-distribution Schema	3
SQL_SIZING	view	DDA-distribution Schema	
SQL_SIZING_PROFILES	view	DDA-distribution Schema	
SQL_LANGUAGES	view	DDA-distribution Schema	
TABLE_CONSTRAINTS	view	Union plus DDA-distribution Schema	
TABLE_PRIVILEGES	view	Union plus DDA-distribution Schema	
TABLES	view	Union plus DDA-distribution Schema	4
TRANSFORMS	view	DDA-distribution Schema	
TRANSLATIONS	view	DDA-distribution Schema	
TRIGGERED_UPDATE_COLUMNS	view	Union	
TRIGGER_COLUMN_USAGE	view	Union	
TRIGGER_TABLE_USAGE	view	Union	
TRIGGERS	view	Union	
USAGE_PRIVILEGES	view	DDA-distribution Schema	1

USER_DEFINED_TYPE_PRIVILEGES	view	DDA-distribution Schema	1
USER_DEFINED_TYPES	view	Union	
VIEW_COLUMN_USAGE	view	Union plus DDA-distribution Schema	
VIEW_TABLE_USAGE	view	Union plus DDA-distribution Schema	
VIEWS	view	Union plus DDA-distribution Schema	

Notes

1. These tables all relate to Access Control, Users and Privileges, which are defined in the DDA-distribution Schema with identical syntax and semantics as in SQL. The definitions of these tables are identical to those in the Information Schema, except that references to the Definition Schema are replaced by references to the DDA-distribution Schema.
2. This table describes SQL language features as defined in IS9075 Annex F.
3. SQL Implementation Information - implementor defined.
4. Tables and Columns are defined in contributing SQL-implementations, and also in temporary tables, views and mapped tables.
5. *Do we want routines in the distributes schema too?*

6 DDA-distribution Schema

The DDA-distribution Schema identifies the SQL-servers that participate in the Distributed Database and the information necessary to derive the DDA-information Schema tables from the Information Schema tables of these SQL-servers.

```
CREATE SCHEMA DISTRIBUTION_SCHEMA
    AUTHORIZATION DISTRIBUTION_SCHEMA
```

DISCUSSION 11 – Is there a relationship between the Authorization DISTRIBUTION_SCHEMA and the Authorization RDA_SUPERVISOR of the base standard?

The DDA-distribution Schema contains:

- Access Control information related to users of the Distributed Database.
- Character Set information.
- SQL-implementation information.
- The definition of tables that are replicated across multiple servers
- The definition of tables whose rows are distributed amongst multiple servers.
- Views specified for the Distributed Database.

The base tables are all defined in a <schema definition> for the schema named DISTRIBUTION_SCHEMA. The table definitions are as complete as the definitional power of SQL allows. Table definitions are supplemented with assertions where appropriate.

Each definition consists of three parts:

- The function of the definition is stated.
- The SQL definition of the object is presented.
- An explanation of the object.

The specification provides only a model of the base tables that are required, and does not imply that an implementation shall provide the functionality in the manner described in this clause.

An instance of the DDA-distribution Schema describes an instance of a cluster of distribution catalogs.

DISCUSSION 12 – It may be appropriate to forbid self-reference and other circularities.

6.1 CHARACTER_SETS base table

Function

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

Definition

```
CREATE TABLE CHARACTER_SETS
(
  CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FORM_OF_USE INFORMATION_SCHEMA.SQL_IDENTIFIER,
  NUMBER_OF_CHARACTERS INFORMATION_SCHEMA.CARDINAL_NUMBER,
  DEFAULT_COLLATE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_COLLATE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_SCHEMA_NOT_NULL NOT NULL,
  DEFAULT_COLLATE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT CHARACTER_SETS_DEFAULT_COLLATE_NAME_NOT_NULL NOT NULL,
  CONSTRAINT CHARACTER_SETS_PRIMARY_KEY PRIMARY KEY (
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ),
  CONSTRAINT CHARACTER_SETS_FOREIGN_KEY_SCHEMATA FOREIGN KEY (
    CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA ) REFERENCES SCHEMATA,
  CONSTRAINT CHARACTER_SETS_CHECK_REFERENCES_COLLATIONS CHECK (
    DEFAULT_COLLATE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA ) OR
    ( DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, DEFAULT_COLLATE_NAME
    ) IN ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME FROM
    COLLATIONS ) )
)
```

Description

- 1) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set being described.
- 2) The value of FORM_OF_USE is a zero-length string.
- 3) The value of NUMBER_OF_CHARACTERS is a zero-length string.
- 4) The values of DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the explicit or implicit default collation for the character set.
- 5) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_TEXT. For that row:
 - a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_TEXT', respectively.

6.1 CHARACTER_SETS base table

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

6.2 COLLATIONS base table

Function

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

Definition

```
CREATE TABLE COLLATIONS
(
  COLLATION_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER CONSTRAINT
  COLLATIONS_CHARACTER_SET_CATALOG_NOT_NULL NOT NULL,
  CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER CONSTRAINT
  COLLATIONS_CHARACTER_SET_SCHEMA_NOT_NULL NOT NULL,
  CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER CONSTRAINT
  COLLATIONS_CHARACTER_SET_NAME_NOT_NULL NOT NULL,
  PAD_ATTRIBUTE INFORMATION_SCHEMA.CHARACTER_DATA CONSTRAINT
  COLLATIONS_PAD_ATTRIBUTE_CHECK CHECK ( PAD_ATTRIBUTE IN ( 'NO PAD',
  'PAD SPACE' ) ),
  COLLATION_TYPE INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_DEFINITION INFORMATION_SCHEMA.CHARACTER_DATA,
  COLLATION_DICTIONARY INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT COLLATIONS_PAD_PRIMARY_KEY PRIMARY KEY (
  COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ),
  CONSTRAINT COLLATIONS_PAD_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA ) REFERENCES
  SCHEMATA,
  CONSTRAINT COLLATIONS_CHECK_REFERENCES_CHARACTER_SETS
  CHECK ( CHARACTER_SET_CATALOG NOT IN ( SELECT CATALOG_NAME FROM
  SCHEMATA )
  OR
  ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME )
  IN
  ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
  CHARACTER_SET_NAME FROM CHARACTER_SETS ) )
)
```

Description

- 1) The values of COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation being described.
- 2) The values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set on which the collation is defined.

6.3 COLUMNS base table

Function

The COLUMNS table has one row for each non-inherited column. It effectively contains a representation of the column descriptors.

NOTE 11 – There are no column entries for columns of tables referenced in views of tables at remote sites, but the table is required to support temporary tables.

Definition

```
CREATE TABLE COLUMNS
(
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  ORDINAL_POSITION INFORMATION_SCHEMA.CARDINAL_NUMBER CONSTRAINT
COLUMN_POSITION_NOT_NULL NOT NULL,
  DOMAIN_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  DOMAIN_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  DOMAIN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_DEFAULT INFORMATION_SCHEMA.CHARACTER_DATA,
  IS_NULLABLE INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT IS_NULLABLE_NOT_NULL NOT NULL
CONSTRAINT IS_NULLABLE_CHECK CHECK ( IS_NULLABLE IN ( 'YES', 'NO' ) ),
  USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.CHARACTER_DATA,
  USER_DEFINED_TYPE_SCHEMA INFORMATION_SCHEMA.CHARACTER_DATA,
  USER_DEFINED_TYPE_NAME INFORMATION_SCHEMA.CHARACTER_DATA,
  CHECK_REFERENCES INFORMATION_SCHEMA.CHARACTER_DATA CONSTRAINT
CHECK_REFERENCES_CHECK
CHECK (CHECK_REFERENCES IN ( 'YES', 'NO' )),
  CHECK_ACTION INFORMATION_SCHEMA.CHARACTER_DATA CONSTRAINT
CHECK_ACTION_CHECK CHECK (CHECK_ACTION IN ( 'RESTRICT', 'SET NULL' ))
CONSTRAINT COLUMNS_PRIMARY_KEY PRIMARY KEY ( TABLE_CATALOG,
TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),
  CONSTRAINT COLUMNS_UNIQUE UNIQUE ( TABLE_CATALOG, TABLE_SCHEMA,
TABLE_NAME, ORDINAL_POSITION ),
  CONSTRAINT COLUMNS_FOREIGN_KEY_TABLES FOREIGN KEY ( TABLE_CATALOG,
TABLE_SCHEMA, TABLE_NAME ) REFERENCES TABLES,
  CONSTRAINT COLUMNS_CHECK_REFERENCES_DOMAIN CHECK ( DOMAIN_CATALOG
NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
OR
( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN SELECT
DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME FROM DOMAINS ) ),
  CONSTRAINT COLUMN_CHECK_DATA_TYPE CHECK ( DOMAIN_CATALOG NOT IN (
SELECT CATALOG_NAME FROM SCHEMATA )
OR
( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NOT NULL
AND
( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) NOT IN (
SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
OBJECT_NAME, COLUMN_NAME FROM DATA_TYPE_DESCRIPTOR )
```

```

OR
( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NULL
AND
( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, COLUMN_NAME FROM
DATA_TYPE_DESCRIPTOR )
))
)

```

Description

- 1) Case:
 - a) If a column is described by a column descriptor included in a table descriptor, then the table descriptor and the column descriptor are associated with that column.
 - b) If a column is described by a column descriptor included in a view descriptor, then the view descriptor and the corresponding column descriptor of the table of the <query expression> are associated with that column.
- 2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the column being described.
- 3) The value of COLUMN_NAME is the name of the column being described.
- 4) The values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are null if the column being described is not defined using a <domain name>. Otherwise, the values of DOMAIN_CATALOG, DOMAIN_SCHEMA, and DOMAIN_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the domain used by the column being described.
- 5) The value of ORDINAL_POSITION is the ordinal position of the column in the table.
- 6) The value of COLUMN_DEFAULT is null if the column being described has no explicit default value or if its default value comes only from a domain. If the character representation of the default value cannot be represented without truncation, then the value of COLUMN_DEFAULT is "TRUNCATED". Otherwise, the value of COLUMN_DEFAULT is a character representation of the default value for the column that obeys the rules specified for <default option> in subclause 11.6, "<default clause>" of ISO/IEC 9075.

NOTE 12 – "TRUNCATED" is different from other values like CURRENT_USER or CURRENT_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.
- 7) The values of IS_NULLABLE have the following meanings:

YES The column is possibly nullable.

NO The column is known not nullable.
- 8) If the data type of the column being defined is not a user-defined type, then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value; otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type used by the column being described.

6.3 COLUMNS base table

- 9) The value of CHECK_REFERENCES is null if the column is associated with a view or the data type of the column is not a reference type. Otherwise, the values of CHECK_REFERENCES have the following meanings:

YES Reference values are checked.

NO Reference values are not checked.

- 10) The value of CHECK_ACTION is null if the column is associated with a view or the data type of the column is not a reference type. Otherwise, the values of CHECK_ACTION have the following meanings:

RESTRICT <reference scope check action> was specified to be RESTRICT.

SET NULL <reference scope check action> was specified to be SET NULL.

6.4 FRAGMENT TABLE base table

Function

Defines a logical component of a mapped table, that is, a table whose rows are stored on a set of SQL_implementations. There is one row in the fragment table for each mapped table fragment.

DISCUSSION 13 – We may not need this table, since it has only key data. It is needed if we attach the condition to the fragment, not if we put the condition on TABLES

Definition

```
CREATE TABLE FRAGMENT_TABLE
(
  CATALOG_NAME INFORMATION_SCHEMA.IDENTIFIER,
  SCHEMA_NAME INFORMATION_SCHEMA.IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.IDENTIFIER,
  FRAGMENT INTEGER,
  IN_CONDITION CHARACTER VARYING (2000);
  CONSTRAINT FRAGMENT_PRIMARY_KEY PRIMARY KEY (CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME, FRAGMENT),
  CONSTRAINT FRAGMENT_SERVER_FOREIGN_KEY (SERVER_NAME) REFERENCES
  SERVER_LOCATION,
  CONSTRAINT FRAGMENT_MAPPED_TABLE_CHECK CHECK ((CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME) IN (SELECT CATALOG_NAME, SCHEMA_NAME,
  TABLE_NAME FROM TABLES WHERE TABLE_TYPE = 'MAPPED'))
)
```

DISCUSSION 14 – Is 2000 a suitable length for IN_CONDITION? What is DBL's policy on such things?

Description

The values of TABLE_CATALOG and TABLE_SCHEMA are the catalog name and unqualified schema name, respectively, of the schema in which the table is defined.

The FRAGMENT numbers for table T form a dense set starting from 1.

DISCUSSION 15 – Is “dense set” sufficiently explicit?

DISCUSSION 16 – Do we need to define the syntax and semantics of IN_CONDITION?

If there are more than one fragment for a mapped table the fragment for storing a row is determined by evaluation of the IN_CONDITIONs in turn starting with fragment 1. The row is stored in the first fragment for which the condition evaluates TRUE. An exception is raised if no fragment is selected.

DISCUSSION 17 – Alternatively, we could require that at least one condition evaluates to TRUE, or we could specify that a particular fragment (first or last) is assumed true.

6.5 REPLICATED FRAGMENT base table

Function

Defines an instance of a table fragment that may or may not be replicated on multiple SQL_Servers. There is one row for each <server table reference> in a CREATE MAPPED TABLE statement.

Definition

```
CREATE TABLE REPLICATED_FRAGMENT
(
  CATALOG_NAME INFORMATION_SCHEMA.IDENTIFIER,
  SCHEMA_NAME INFORMATION_SCHEMA.IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.IDENTIFIER,
  FRAGMENT INTEGER,
  SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
  SERVER_CATALOG_NAME INFORMATION_SCHEMA.IDENTIFIER,
  SERVER_SCHEMA_NAME INFORMATION_SCHEMA.IDENTIFIER,
  SERVER_TABLE_NAME INFORMATION_SCHEMA.IDENTIFIER,
  CONSTRAINT REPLICATED_PRIMARY_KEY PRIMARY KEY (CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME, FRAGMENT, SERVER_NAME),
  CONSTRAINT REPLICATED_FRAGMENT_FOREIGN_KEY (CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME, FRAGMENT) REFERENCES FRAGMENT,
  CONSTRAINT REPLICATED_UNIQUE_COMPONENT UNIQUE (SERVER_NAME,
  SERVER_CATALOG_NAME, SERVER_SCHEMA_NAME, SERVER_TABLE_NAME),
  CONSTRAINT REPLICATED_SERVER_FOREIGN_KEY (SERVER_NAME) REFERENCES
  SERVER_LOCATION,
  CONSTRAINT REPLICATED_TABLE_NAME_CHECK (CATALOG_NAME, SCHEMA_NAME,
  TABLE_NAME IN (SELECT CATALOG_NAME, SCHEMA_NAME, TABLE_NAME FROM
  TABLES) WHERE TABLE_TYPE = 'MAPPED')
)
```

Description

- 1) The values of TABLE_CATALOG and TABLE_SCHEMA are the catalog name and unqualified schema name, respectively, of the schema in which the table is defined.
- 2) TABLE_NAME names a table *T* in the distributed database identified by TABLE_CATALOG and TABLE_SCHEMA.
- 3) SERVER_NAME references an SQL_Server *S*.
- 4) SERVER_CATALOG, SERVER_SCHEMA, SERVER_TABLE_NAME together reference a table *ST* that exists at *S*.
- 5) If there is more than one row for the same *T*, then let *R* be the set of all the referenced tables *ST*. All the tables in *R* have the same structure (i.e. the same column names with the same data types, in the same sequence). Tables referenced from rows with the same fragment number are replicates and contain equivalent data. Tables referenced from rows with different fragment numbers contain no common rows.

DISCUSSION 18 – Does “equivalent data” need further definition?

- 6) Statements that retrieve data from T may access any of the replicated tables in each fragment.
- 7) Statements that update T update each replicated table in the fragments affected.

6.6 ROLE_AUTHORIZATION_DESCRIPTORs base table

Function

Contains a representation of the role authorization descriptors.

Definition

```
CREATE TABLE ROLE_AUTHORIZATION_DESCRIPTORs
(
  ROLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE INFORMATION_SCHEMA.SQL_IDENTIFIER
  CHECK ( GRANTEE IN ( SELECT ROLE_NAME FROM ROLES )
OR GRANTEE IN ( SELECT USER_NAME FROM USERS ) ),
  GRANTOR INFORMATION_SCHEMA.SQL_IDENTIFIER
  CHECK ( GRANTOR IN ( SELECT ROLE_NAME FROM ROLES )
OR GRANTOR IN ( SELECT USER_NAME FROM USERS ) ),
  IS_GRANTABLE INFORMATION_SCHEMA.CHARACTER_DATA
  CHECK ( IS_GRANTABLE IN ( 'YES', 'NO' ) ),
  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_PRIMARY_KEY
  PRIMARY KEY ( ROLE_NAME, GRANTEE ),
  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_FOREIGN_KEY_ROLES
  FOREIGN KEY ( ROLE_NAME )
  REFERENCES ROLES,
)
```

Description

- 1) A row is (or rows are) inserted into this table whenever a <grant role statement> or <role definition> is executed unless the necessary row already exists, in which case the existing row may be modified to change the IS_GRANTABLE column. A row is (or rows are) deleted from this table whenever a <revoke role statement> or <drop role> is executed.
- 2) The value of ROLE_NAME is the <role name> of some <role granted> by the <grant role statement> or the <role name> of a <role definition>.
- 3) The value of GRANTEE is an <authorization identifier>, possibly PUBLIC, or <role name> specified as a <grantee> contained in a <grant role statement>, or the <authorization identifier> of the current SQL-session when the <role definition> is executed.
- 4) The value of GRANTOR is the <authorization identifier> of the user or role who granted the role identified by ROLE_NAME to the user or role identified by the value of GRANTEE.
- 5) The values of IS_GRANTABLE have the following meanings:

YES The described role is grantable.
NO The described role is not grantable.
- 6) A role is grantable if the WITH ADMIN OPTION is specified in the <grant role statement> or a <role definition> is executed.

6.7 ROLES base table

Function

The ROLES table has one row for each <role name> for each role known to the database management system.

Definition

```
CREATE TABLE ROLES
(
  ROLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER PRIMARY KEY,
  CONSTRAINT ROLES_CHECK CHECK ( ROLE_NAME NOT IN ( SELECT USER_NAME
FROM USERS ) )
)
```

Description

- 1) A row is inserted into this table each time a <role definition> is executed. A row is deleted from this table each time the <drop role statement> is executed.
- 2) The value of ROLE_NAME is the <role name> defined by <role definition>.

6.8 SCHEMATA base table

Function

The SCHEMATA table has one row for each schema.

Definition

```
CREATE TABLE SCHEMATA
(
  CATALOG_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_OWNER INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT SCHEMA_OWNER_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL NOT NULL,
  SQL_PATH INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SCHEMATA_PRIMARY_KEY PRIMARY KEY ( CATALOG_NAME,
  SCHEMA_NAME ),
  CONSTRAINT SCHEMATA_FOREIGN_KEY FOREIGN KEY ( SCHEMA_OWNER )
  REFERENCES USERS
)
```

Description

- 1) All the values of CATALOG_NAME are the name of the catalog in which the schemata are included.
- 2) The values of SCHEMA_NAME are the unqualified schema names of the schemata in the catalog.
- 3) The values of SCHEMA_OWNER are the authorization identifiers that own the schemata.
- 4) The values of DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, and DEFAULT_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default character set for columns and domains in the schemata.
- 5) Case:
 - a) If <schema path specification> was specified in the <schema definition> that defined the schema described by this row and the character representation of the <schema path specification> can be represented without truncation, then the value of SQL_PATH is that character representation.
 - b) Otherwise, the value of SQL_PATH is the null value.

6.9 SERVER AUTHENTICATION base table

Function

Define the authorisation identifiers to be used for given users on other servers.

Definition

```
CREATE TABLE SERVER_AUTHENTICATION
(
  SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
  USER_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CHECK (USER_NAME IN ( SELECT ROLE_NAME FROM ROLES ) OR USER_NAME IN (
  SELECT USER_NAME FROM USERS ) ),
  AUTH_ID INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT PRIMARY_KEY (SERVER_NAME, USER_NAME),
  CONSTRAINT FOREIGN_KEY_SERVER_NAME FOREIGN KEY (SERVER_NAME)
REFERENCES SERVER_LOCATION,
  CONSTRAINT FOREIGN_KEY_USER_NAME_CHECK
)
```

Description

- 1) SERVER_NAME identifies an SQL_Server *S*.
- 2) USER_NAME identifies either a User or a Role declared for this schema.
- 3) AUTH_ID is the auth_id that this user is entitled to use for the server *S*.

NOTE 13 – The USER_NAME is a foreign key to either ROLES table or USERS table but not both.

6.10 SERVER LOCATION base table

Function

Identify SQL Servers and define access to them.

Definition

```
CREATE TABLE SERVER_LOCATION
  { SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
    TRANSPORT_MAPPING INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
    TRANSPORT_ADDRESS SMALLINT,
    RDA_DESTINATION_SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
    INFORMATION_SCHEMA_CATALOG_NAME INFORMATION_SCHEMA.IDENTIFIER NOT
  NULL,
  CONSTRAINT SERVER_LOCATION_PRIMARY_KEY PRIMARY KEY (SERVER_NAME)
  }
```

DISCUSSION 19 – Is 'smallint' for TRANSPORT_ADDRESS correct?

Description

- 1) For each SQL-Server *S* recorded in SERVER_LOCATION:
 - a) SERVER_NAME is the SQL-Server name of *S* as known to the Service User
 - b) TRANSPORT_MAPPING: the code identifying the transport mapping through which a Transport Connection to *S* can be established as defined in REF table 18.
 - c) TRANSPORT_ADDRESS: the Transport Address of *S* within the Transport Provider identified by TRANSPORT_MAPPING.
 - d) RDA_DESTINATION_SERVER_NAME: The Destination SQL_Server name of *S* within the RDA server Environment identified by the values of TRANSPORT_MAPPING and TRANSPORT_ADDRESS taken together.

6.11 SQL_FEATURES base table

Function

The SQL_FEATURES base table has one row for each feature and subfeature identified by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_FEATURES
(
  FEATURE_ID INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_NAME INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_FEATURES_FEATURE_NAME_NOT_NULL NOT NULL,
  /* Zero for SUB_FEATURE_ID indicates a feature */
  SUB_FEATURE_ID INFORMATION_SCHEMA.CHARACTER_DATA,
  /* Zero-length string for SUB_FEATURE_NAME indicates a feature */
  SUB_FEATURE_NAME INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_FEATURES_SUB_FEATURE_NAME_NOT_NULL NOT NULL,
  IS_SUPPORTED INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_FEATURES_IS_SUPPORTED_NOT_NULL NOT NULL
  CONSTRAINT SQL_FEATURES_IS_SUPPORTED_CHECK
  CHECK ( IS_SUPPORTED IN ( 'YES', 'NO' ) ),
  IS_VERIFIED_BY INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_COMMENTS INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_FEATURES_PRIMARY_KEY
  PRIMARY KEY ( FEATURE_ID, SUB_FEATURE_ID ),
  CONSTRAINT SQL_FEATURES_CHECK_SUPPORTED_VERIFIED
  CHECK ( IS_SUPPORTED = 'YES' OR IS_VERIFIED_BY IS NULL )
)
```

Description

- 1) The SQL_FEATURES table consists of exactly one row for each SQL feature and subfeature defined in ISO/IEC 9075 Annex F, "SQL Feature Taxonomy", of this part and corresponding Annexes in other parts of ISO/IEC 9075.
- 2) A feature is identified by the value 0 (zero) in the SUB_FEATURE_ID column. A subfeature is identified by a value other than 0 (zero) in the SUB_FEATURE_ID column.
- 3) The FEATURE_ID and FEATURE_NAME columns identify the feature by the feature identifier and name assigned to it.
- 4) The SUB_FEATURE_ID and SUB_FEATURE_NAME columns identify the subfeature by the subfeature identifier and name assigned to it.
- 5) The IS_SUPPORTED column is 'YES' if an implementation fully supports that feature or subfeature when SQL-data in the identified catalog is accessed through that implementation and is 'NO' if the implementation does not fully support that feature or subfeature when accessing SQL-data in that catalog.

6.11 SQL_FEATURES base table

- 6) If full support for the feature or subfeature has been verified by testing, then the IS_VERIFIED_BY column shall contain information identifying the conformance test used to verify the conformance claim; otherwise, IS_VERIFIED_BY shall be the null value.
- 7) If the value of the IS_SUPPORTED column for a feature is 'YES' and if that feature has subfeatures, then the value of the IS_SUPPORTED column in every row identifying subfeatures of the feature shall also be 'YES'.
- 8) The FEATURE_COMMENTS column is intended for any implementer comments pertinent to the identified SQL feature or subfeature.

6.12 SQL_IMPLEMENTATION_INFO base table

Function

The SQL_IMPLEMENTATION_INFO base table has one row for each implementation information item defined by ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_IMPLEMENTATION_INFO
(
  IMPLEMENTATION_INFO_ID INFORMATION_SCHEMA.CHARACTER_DATA,
  IMPLEMENTATION_INFO_NAME INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_IMPLEMENTATION_INFO_NAME_NOT_NULL NOT NULL,
  INTEGER_VALUE INFORMATION_SCHEMA.CARDINAL_NUMBER,
  CHARACTER_VALUE INFORMATION_SCHEMA.CHARACTER_DATA,
  IMPLEMENTATION_INFO_COMMENTS INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
  PRIMARY KEY ( IMPLEMENTATION_INFO_ID )
);
```

Description

- 1) The SQL_IMPLEMENTATION_INFO table consists of exactly one row for each SQL implementation information item defined in ISO/IEC 9075.
- 2) The IMPLEMENTATION_INFO_ID and IMPLEMENTATION_INFO_NAME columns identify the implementation information item by the integer and name assigned to it.
- 3) Depending on the type of information, the value is present in either INTEGER_VALUE or CHARACTER_VALUE; the other column is the null value. Within the applicable column for the item, the values are:

<i>Value</i>	<i>Meaning</i>
0 (zero) or a zero-length string (as appropriate)	The value for this item is unknown.
null	The value is not applicable to the implementation.
Any other value	The value of the item.

- 4) The IMPLEMENTATION_INFO_COMMENTS column is intended for any implementer comments pertinent to the identified item.

6.13 SQL_SIZING base table

Function

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_SIZING
(
  SIZING_ID INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_NAME INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
  /* If SUPPORTED_VALUE is the null value, that means that the item */
  /* being described is not applicable in the implementation. */
  /* If SUPPORTED_VALUE is 0 (zero), that means that the item */
  /* being described has no limit or the limit cannot be determined.*/
  SUPPORTED_VALUE INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_COMMENTS INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_SIZING_PRIMARY_KEY
  PRIMARY KEY (SIZING_ID )
)
```

Description

- 1) The SQL_SIZING table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075.
- 2) The SIZING_ID and SIZING_NAME columns identify the sizing item by the integer and description assigned to it.
- 3) The values of the SUPPORTED_VALUE column are:

<i>Value</i>	<i>Meaning</i>
0	The implementation either places no limit on this sizing item or the implementation cannot determine the limit.
null.	The implementation does support not any features for which this sizing item is applicable
Any other value	The maximum size supported by the implementation for this sizing item.

- 4) The SIZING_COMMENTS column is intended for any implementor comments pertinent to the identified SQL sizing item.

6.14 SQL_SIZING_PROFILES base table

Function

The SQL_SIZING base table has one row for each sizing idem defined in ISO/IEC 9075.

Definition

```
CREATE TABLE SQL_SIZING_PROFILES
(
  SIZING_ID INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_NAME INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
  PROFILE_ID INFORMATION_SCHEMA.CHARACTER_DATA,
  /* If REQUIRED_VALUE is the null value, that means that the item */
  /* being described is not applicable in the profile. */
  /* If REQUIRED_VALUE is 0 (zero), that means that the profile */
  /* does not set a limit for this sizing item. */
  REQUIRED_VALUE INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_PROFILE_COMMENTS INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_SIZING_PROFILE_PRIMARY_KEY
  PRIMARY KEY (SIZING_ID, PROFILE_ID )
)
```

Description

- 1) The SQL_SIZING_PROFILE table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075 for each profile that is defined in the table.
- 2) The SIZING_ID and SIZING_NAME columns identify the sizing item by the integer and description assigned to it.
- 3) The PROFILE_ID column shall contain information identifying a profile.
- 4) The values of the REQUIRED_VALUE column are:

<i>Value</i>	<i>Meaning</i>
0	The profile places no limit on this sizing item.
null	The profile does not require any features for which this sizing item is applicable.
Any other value	The minimum size required by the profile for this sizing item.

- 5) The SIZING_PROFILES_COMMENTS column is intended for any implementor comments pertinent to the identified SQL sizing item within the profile.

6.15 SQL_LANGUAGES base table

Function

The SQL_LANGUAGES table has one row for each ISO and implementation-defined SQL language binding and programming language for which conformance is claimed.

NOTE 14 – The SQL_LANGUAGES base table provides, among other information, the same information provided by the SQL object identifier specified in Subclause 6.3, "Object identifier for Database Language SQL", in ISO/IEC 9075-1.

DISCUSSION 20 – *Is a new Object Identifier required for Distributed Data Access?*

Definition

```

CREATE TABLE SQL_LANGUAGES
(
  SQL_LANGUAGE_SOURCE INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_LANGUAGES_SOURCE_NOT_NULL NOT NULL,
  SQL_LANGUAGE_YEAR INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_LANGUAGES_YEAR_ISO
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_YEAR IS NOT NULL AND
  SQL_LANGUAGE_YEAR IN ( '1987', '1989', '1992', '1998' ) )
  ),
  SQL_LANGUAGE_CONFORMANCE INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_LANGUAGE_CONFORMANCE_ISO_1987
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_YEAR <> '1987'
  OR ( SQL_LANGUAGE_CONFORMANCE IS NOT NULL AND
  SQL_LANGUAGE_CONFORMANCE IN ( '1', '2' ) )
  )
  )
  CONSTRAINT SQL_LANGUAGE_CONFORMANCE_ISO_1989
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_YEAR <> '1989'
  OR ( SQL_LANGUAGE_CONFORMANCE IS NOT NULL AND
  SQL_LANGUAGE_CONFORMANCE IN ( '1', '2' ) )
  )
  )
  CONSTRAINT SQL_LANGUAGE_CONFORMANCE_ISO_1992
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_YEAR <> '1992'
  OR ( SQL_LANGUAGE_CONFORMANCE IS NOT NULL AND
  SQL_LANGUAGE_CONFORMANCE IN
  ( 'ENTRY', 'INTERMEDIATE', 'FULL' ) )
  )
  )
  CONSTRAINT SQL_LANGUAGE_CONFORMANCE_ISO_1998
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_YEAR <> '1998'
  OR ( SQL_LANGUAGE_CONFORMANCE IS NOT NULL AND
  SQL_LANGUAGE_CONFORMANCE IN ( 'CORE' ) )
  )
  ),
  SQL_LANGUAGE_INTEGRITY INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_LANGUAGE_INTEGRITY_ISO_1989
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR ( SQL_LANGUAGE_INTEGRITY IS NULL
  OR ( SQL_LANGUAGE_YEAR = '1989' AND
  SQL_LANGUAGE_INTEGRITY IS NOT NULL AND
  SQL_LANGUAGE_INTEGRITY IN ( 'NO', 'YES' ) )
  )
  ),
  SQL_LANGUAGE_IMPLEMENTATION INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_LANGUAGE_IMPLEMENTATION_ISO
  CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
  OR SQL_LANGUAGES_IMPLEMENTATION IS NULL ),
  SQL_LANGUAGE_BINDING_STYLE INFORMATION_SCHEMA.CHARACTER_DATA

```

```
CONSTRAINT SQL_LANGUAGE_BINDING_STYLE_ISO_1987
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1987'
OR ( SQL_LANGUAGE_BINDING_STYLE IS NOT NULL AND
SQL_LANGUAGE_BINDING_STYLE IN
( 'DIRECT', 'EMBEDDED', 'MODULE' ) )
)
)
CONSTRAINT SQL_LANGUAGE_BINDING_STYLE_ISO_1989
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1989'
OR ( SQL_LANGUAGE_BINDING_STYLE IS NOT NULL AND
SQL_LANGUAGE_BINDING_STYLE IN
( 'DIRECT', 'EMBEDDED', 'MODULE' ) )
)
)
CONSTRAINT SQL_LANGUAGE_BINDING_STYLE_ISO_1992
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1992'
OR ( SQL_LANGUAGE_BINDING_STYLE IS NOT NULL AND
SQL_LANGUAGE_BINDING_STYLE IN
( 'DIRECT', 'EMBEDDED', 'MODULE' ) )
)
),
SQL_LANGUAGE_PROGRAMMING_LANGUAGE INFORMATION_SCHEMA.CHARACTER_DATA,
SQL_LANGUAGES_STANDARD_VALID_CHECK_ISO_1987
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1987'
OR ( SQL_LANGUAGE_BINDING_STYLE = 'DIRECT' AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IS NULL )
OR ( SQL_LANGUAGE_BINDING_STYLE IN
( 'EMBEDDED', 'MODULE' ) AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
( 'COBOL', 'FORTRAN', 'PASCAL', 'PLI' ) )
)
)
SQL_LANGUAGES_STANDARD_VALID_CHECK_ISO_1989
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1989'
OR ( SQL_LANGUAGE_BINDING_STYLE = 'DIRECT' AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IS NULL )
OR ( SQL_LANGUAGE_BINDING_STYLE IN
( 'EMBEDDED', 'MODULE' ) AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
( 'COBOL', 'FORTRAN', 'PASCAL', 'PLI' ) )
)
)
SQL_LANGUAGES_STANDARD_VALID_CHECK_ISO_1992
CHECK ( SQL_LANGUAGE_SOURCE <> 'ISO 9075'
OR ( SQL_LANGUAGE_YEAR <> '1992'
OR ( SQL_LANGUAGE_BINDING_STYLE = 'DIRECT' AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IS NULL )
OR ( SQL_LANGUAGE_BINDING_STYLE IN
( 'EMBEDDED', 'MODULE' ) AND
SQL_LANGUAGE_PROGRAMMING_LANGUAGE IN
```

```
( 'ADA', 'C', 'COBOL', 'FORTRAN',
  'MUMPS', 'PASCAL', 'PLI' ) )
)
)
)
```

Description

- 1) Each row represents one binding of an ISO or implementation-defined SQL language.
- 2) The value of SQL_LANGUAGE_SOURCE is the name of the source of the language definition. The source of standard SQL language is the value 'ISO 9075', while the source of an implementation-defined version of SQL is implementation-defined.
- 3) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_YEAR is the year that the ISO standard was approved. Otherwise, the value of SQL_LANGUAGE_YEAR is implementation-defined.

NOTE 15 – As each new ISO SQL standard revision is approved, a new valid value of SQL_LANGUAGE_YEAR must be added to the CHECK constraint for this column.

- 4) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_CONFORMANCE is the conformance level to which conformance is claimed for the ISO standard. Otherwise, the value of SQL_LANGUAGE_CONFORMANCE is implementation-defined.
- 5) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075' and that language contains an optional integrity enhancement feature, then the value of SQL_LANGUAGE_INTEGRITY is 'YES' if conformance is claimed to the integrity enhancement feature, and 'NO' otherwise. Otherwise, the value of SQL_LANGUAGE_INTEGRITY is implementation-defined.
- 6) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_IMPLEMENTATION is null. Otherwise, the value of SQL_LANGUAGE_IMPLEMENTATION is an implementation-defined character string value.
- 7) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_BINDING_STYLE is the style of binding of the SQL language. If the value of SQL_LANGUAGE_BINDING_STYLE is 'MODULE', then the binding style of <SQL-client module> is supported. If the value of SQL_LANGUAGE_BINDING_STYLE is 'EMBEDDED', then the binding style of <embedded SQL host program> is supported. If the value of SQL_LANGUAGE_BINDING_STYLE is 'DIRECT', then the binding style of <direct SQL statement> is supported. Otherwise, the value of SQL_LANGUAGE_BINDING_STYLE is implementation-defined.
- 8) If the value of SQL_LANGUAGE_SOURCE is 'ISO 9075', then the value of SQL_LANGUAGE_PROGRAMMING_LANGUAGE is the programming language supported by the binding style indicated by the value of SQL_LANGUAGE_BINDING_STYLE. If the value of SQL_LANGUAGE_BINDING_STYLE is 'DIRECT', then SQL_LANGUAGE_PROGRAMMING_LANGUAGE is the null value. If the value of SQL_LANGUAGE_BINDING_STYLE is 'MODULE' or 'EMBEDDED', then SQL_LANGUAGE_PROGRAMMING_LANGUAGE has the value 'ADA', 'C', 'COBOL', 'FORTRAN', 'MUMPS', 'PASCAL', or 'PLI'.
- 9) Case:

6.15 SQL_LANGUAGES base table

- a) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is ;ADA;, then Ada is supported with the given binding style.
- b) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'C', then C is supported with the given binding style.
- c) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'COBOL', then COBOL is supported with the given binding style.
- d) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'FORTRAN', then Fortran is supported with the given binding style.
- e) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'MUMPS', then MUMPS is supported with the given binding style.
- f) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'PASCAL', then Pascal is supported with the given binding style.
- g) If SQL_LANGUAGE_PROGRAMMING_LANGUAGE is 'PLI', then PL/I is supported with the given binding style.
- h) Otherwise, the value of SQL_LANGUAGE_PROGRAMMING_LANGUAGE is implementation-defined.

6.16 TABLES base table

Function

The TABLES table contains one row for each table defined in the DDA-distribution Schema. It includes views, temporary tables and mapped tables.

Definition

```
CREATE TABLE TABLES
(
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_TYPE INFORMATION_SCHEMA.CHARACTER_DATA,
  FUNCTION_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FUNCTION_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FUNCTION_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT TABLE_TYPE_NOT_NULL NOT NULL,
  CONSTRAINT TABLE_TYPE_CHECK CHECK ( TABLE_TYPE IN ( 'VIEW', 'GLOBAL
TEMPORARY', 'LOCAL TEMPORARY', 'MAPPED' ) ),
  CONSTRAINT CHECK_TABLE_IN_COLUMNS CHECK ( ( TABLE_CATALOG,
TABLE_SCHEMA, TABLE_NAME ) IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA,
TABLE_NAME FROM COLUMNS ) OR TABLE_TYPE = 'MAPPED' ),
  CONSTRAINT FUNCTION_CHECK CHECK ((TABLE_TYPE = 'MAPPED' AND
CONDITION IS NOT NULL) OR CONDITION IS NULL)
  CONSTRAINT TABLES_PRIMARY_KEY
  PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),
  CONSTRAINT TABLES_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA ) REFERENCES SCHEMATA,
  CONSTRAINT TABLES_CHECK_NOT_VIEW CHECK ( NOT EXISTS ( SELECT
TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME FROM TABLES
WHERE TABLE_TYPE = 'VIEW'
EXCEPT
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME FROM VIEWS ) )
  CONSTRAINT TABLES_CHECK_NOT_MAPPED CHECK ( NOT EXISTS ( SELECT
TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME FROM TABLES
WHERE TABLE_TYPE = ' MAPPED'
EXCEPT
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
FROM MAPPED ) )
)
```

Description

- 1) The values of TABLE_CATALOG and TABLE_SCHEMA are the catalog name and unqualified schema name, respectively, of the schema in which the table is defined.
- 2) The value of TABLE_NAME is the name of the table.
- 3) The values of TABLE_TYPE have the following meanings:

VIEW	The table being described is a viewed table
------	---

6.16 TABLES base table

GLOBAL TEMPORARY	The table being described is a global temporary table
LOCAL TEMPORARY	The table being described is a created local temporary table
PARTITIONED	The table being described is a table created from a Union of tables on different SQL-Implementations
MAPPED	The table being described is located at more than one site as determined by rows in the FRAGMENT and REPLICATION tables

- 4) Fragments are numbered in the order of their declaration in the CREATE MAPPED TABLE statement.

DISCUSSION 21 – Is this the right place for this rule?

6.17 USERS base table

Function

The USERS table has one row for each <authorization identifier> referenced in the DDA-information Schema. These are all those <authorization identifier>s that may grant privileges as well as those that may create a schema, or currently own a schema created through a <schema definition>.

Definition

```
CREATE TABLE USERS
(
  USER_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT USERS_PRIMARY_KEY PRIMARY KEY
  CONSTRAINT USERS_CHECK CHECK ( USER_NAME NOT IN ( SELECT ROLE_NAME
FROM ROLES ) )
)
```

Description

- 1) The values of USER_NAME are <authorization identifier>s that are known as users within the DDA-distribution Schema.

6.18 VIEWS base table

Function

The VIEWS table contains one row for each CREATE VIEW Statement in the DDA-distribution Schema. Each row describes the query expression that defines a view.

NOTE 16 – This table effectively contains a representation of the view descriptors.

Definition

```
CREATE TABLE VIEWS
(
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_DEFINITION INFORMATION_SCHEMA.CHARACTER_DATA,
  CHECK_OPTION INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT CHECK_OPTION_NOT_NULL NOT NULL
  CONSTRAINT CHECK_OPTION_CHECK
  CHECK ( CHECK_OPTION IN ( 'CASCADED', 'LOCAL', 'NONE' ) ),
  IS_UPDATABLE INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT IS_UPDATABLE_NOT_NULL NOT NULL
  CONSTRAINT IS_UPDATABLE_CHECK
  CHECK ( IS_UPDATABLE IN ( 'YES', 'NO' ) ),
  CONSTRAINT VIEWS_PRIMARY_KEY
  PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),
  CONSTRAINT VIEWS_TABLE_NAME_CHECK
  ( CATALOG_NAME, SCHEMA_NAME, TABLE_NAME NOT IN ( SELECT CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME FROM PARTITIONED_TABLE ) AND CATALOG_NAME,
  SCHEMA_NAME, TABLE_NAME NOT IN ( SELECT CATALOG_NAME, SCHEMA_NAME,
  TABLE_NAME FROM REPLICATED_TABLE ) )
  CONSTRAINT VIEWS_IS_UPDATABLE_CHECK_OPTION_CHECK
  CHECK ( ( IS_UPDATABLE, CHECK_OPTION ) NOT IN ( VALUES ( 'NO',
  'CASCADED' ), ( 'NO', 'LOCAL' ) ) )
)
```

Description

- 1) The values of TABLE_CATALOG and TABLE_SCHEMA are the catalog name and unqualified schema name, respectively, of the schema in which the viewed table is defined.
- 2) The value of TABLE_NAME is the name of the viewed table.
- 3) Case:
 - a) If the character representation of the <query expression> contained in the <view definition> that defined the view being described can be represented without truncation, then the value of VIEW_DEFINITION is that character representation.
 - b) Otherwise, the value of VIEW_DEFINITION is the null value.

NOTE 17 – Any implicit column references that were contained in the <query expression> associated with the <view definition> are replaced by explicit column references in VIEW_DEFINITION. An

explicit column reference is of form:

[[<catalog name> .<schema name> .]<table name>.<column name> [@<server name>]

- 4) The values of CHECK_OPTION have the following meanings:

CASCADED The <view definition> contains WITH CASCADED CHECK OPTION.

LOCAL The <view definition> contains WITH LOCAL CHECK OPTION.

NONE The <view definition> does not contain WITH CHECK OPTION.

- 5) The values of IS_UPDATABLE have the following meanings:

YES The <view definition> simply contains a <query expression> that is updatable.

NO The <view definition> simply contains a <query expression> that is not updatable.

6.19 VIEW_COLUMN_USAGE base table

Function

The VIEW_COLUMN_USAGE table has one row for each column of a table that is explicitly or implicitly referenced in the <query expression> of the view being described.

Definition

```
CREATE TABLE VIEW_COLUMN_USAGE
(
  VIEW_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT VIEW_COLUMN_USAGE_PRIMARY_KEY
  PRIMARY KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
  TABLE_SERVER_NAME, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
  COLUMN_NAME ),
  CONSTRAINT VIEW_SERVER_NAME_CHECK CHECK ( VIEW_SERVER_NAME IS NULL
  OR VIEW_SERVER_NAME IN (SELECT SERVER_NAME FROM SERVER_LOCATION ) ),
  CONSTRAINT VIEW_COLUMN_USAGE_FOREIGN_KEY_VIEWS
  FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME ) REFERENCES
  VIEWS
)
```

Description

- 1) The values of VIEW_CATALOG, VIEW_SCHEMA, and VIEW_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the view being described.
- 2) The values of TABLE_SERVER_NAME, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the SQL_Server name, catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column of a table that is explicitly or implicitly referenced in the <query expression> of the view being described.

6.20 VIEW_TABLE_USAGE base table

Function

The VIEW_TABLE_USAGE table has one row for each table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of a view.

Definition

```
CREATE TABLE VIEW_TABLE_USAGE
(
  VIEW_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SERVER_NAME INFORMATION_SCHEMA.IDENTIFIER NOT NULL,
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT VIEW_TABLE_USAGE_PRIMARY_KEY
  PRIMARY KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
  TABLE_SERVER_NAME, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),
  CONSTRAINT VIEW_SERVER_NAME_CHECK CHECK ( VIEW_SERVER_NAME IS NULL
  OR VIEW_SERVER_NAME IN (SELECT SERVER_NAME FROM SERVER_LOCATION ) ),
  CONSTRAINT VIEW_TABLE_USAGE_FOREIGN_KEY_VIEWS
  FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME ) REFERENCES
  VIEWS
)
```

Description

- 1) The values of VIEW_CATALOG, VIEW_SCHEMA, and VIEW_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the view being described.
- 2) The values of TABLE_SERVER_NAME, TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the SQL_Server name, catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the view being described.

7.1 General constructs

7 DDA-distribution Statements

7.1 General constructs

The DDA-distribution Schema SQL Definition is defined as an extension and modification of SQL, as defined in IS 9075. Wherever possible the DDA-distribution Schema SQL Definition makes use of statements already defined in IS9075. The proposed additional statements make use of many of the elements that are common to other SQL Statements. This clause defines differences in the general elements.

7.1.1 Expressions

Views may be declared in the DDA-distribution Schema and so the full expression syntax from the recursive expansion of productions of <query expression> is required.

7.2 DDA-distribution Schema Definition

Function

This clause defines the structure of a DDA-distribution Schema definition.

Format

```

<distributed data definition> ::= CREATE DISTRIBUTION SCHEMA
    <schema name clause>
    [ <schema character set or path> ]      -- not core
    [ <schema element>... ]

<schema character set or path> ::= <schema character set specification>
    | <schema path specification>
    | <schema character set specification> <schema path specification>
    | <schema path specification> <schema character set specification>

<schema name clause> ::= <schema name>
    | AUTHORIZATION <schema authorization identifier>
    | <schema name> AUTHORIZATION <schema authorization identifier>

<schema authorization identifier> ::= <authorization identifier>

<schema character set specification> ::= DEFAULT CHARACTER SET
<character set specification>

<schema path specification> ::= <path specification>

<schema element> ::= <distributed schema element>
    | <view definition>
    | <character set definition> -- not core
    | <collation definition> - not core
    | <translation definition> - not core
    | <user-defined type definition> -- constrained
    | <schema routine>
    | <grant statement>
    | <role definition>
    | <grant role statement>

<distributed schema element> ::= <server location definition>
    | <server authentication>
    | <mapped table definition>
    / <partitioned table definition>

```

Syntax Rules

- 1) There must be at least one <server location definition>.

General Rules

- 2) A row is inserted into the Distributed Schema Schemata table with the <schema name> defined in the schema name clause.
- 3) <distributed schema element> components are defined in following clauses. Other elements are defined in IS 9075.

7.3 DDA-distribution Schema Elements

This subclause defines the constructs in the DDA-distribution Statements which are specific to the DDA-distribution Schema, or which differ from the equivalent statements in IS 9075.

7.3.1 Server Location Definition

Function

Name the servers contributing to the DDB and provide connection information

Syntax

```
<server location definition> ::=
    CREATE SERVER LOCATION <server name> TRANSPORT MAPPING <transport
name> ADDRESS <address>
    RDA_DESTINATION <rda server name>
    INFORMATION SCHEMA <catalog name>
```

Syntax Rules

- 1) <server name> is unique amongst all the Server Location Definitions in the DDA-distribution Schema.

General Rules

- 1) A row is inserted into the Server Location table of the DDA-distribution Schema with values

7.3.2 Server Authentication

Function

Identify the authentication identifier to be used for a given user for access to data at a server.

Syntax

```
<server authentication> ::= LOGIN [ FOR < authorisation identifier> ]
AT <server name> IS <authorisation identifier>
```

Syntax Rules

- 1) Let *A* be the <authorisation identifier> in the FOR subclause. If there is no FOR subclause then *A* is the user AUTHORITY_ID for this session.
- 2) Let *AA* be the <authorisation identifier> in the IS subclause.
- 3) Let *S* be the SQL-implementation identified by <server name>.
- 4) There is at most one <server authentication> for any *A* and *S*.

General Rules

- 1) A row is inserted into the User Authorisation Table specifying that the authorisation identifier to be used when a connection is made to server *S* for user *A* is *AA*.
- 2) Tables available to user *AA* in *S* are available to user *A* of this schema with the same privileges.

7.3.3 Mapped Table Definition

Function

Creates a description of a mapped table in the DDA-distribution Schema. A mapped table is mapped to tables in the contributing SQL-implementations using partitioning, replication, or both.

Syntax

```
< mapped table definition > ::= CREATE MAPPED TABLE <table name >
<table element list> )
<component list>
<component list> ::= < component>
| <conditional component> {, < conditional component> }...
<conditional component> ::= IF < insert condition> <component>
<component> ::= AT <server table list>
<server table ref list> ::= <server table ref> [ {, <server table ref>
} ... ]
<server table ref> ::= <server name> [ ALIAS <table name> ]
NOTE: there are many ways of defining allocation schemes and one could
make some sort of case for most of them! This one, and the CASE one are
the most obvious.
<table element list> ::=
<left paren> <table element> [ { <comma> <table element> }... ] <right
paren>
<table element> ::=
<column definition>
| <table constraint definition>
```

Rules

- 1) Let *T* be the value of <table name> in < Mapped Table Assertion>.
- 2) *T* is unique among the Views and Mapped tables in this Schema.
- 3) Let *S* be the value of < server name> in a < server table ref> and *ST* be,
 - a) if there is no ALIAS subclause the value *T*,
 - b) else the value of <table name> in the ALIAS subclause.
- 4) <conditional expression> is a boolean expression referencing only columns in *T* and constants.
- 5) For each server *S* in the <server table list> there is a table with name *ST*. Further, for each server *S* the name *ST* is unique amongst the set of names *ST* in the component list.

The tables ST in all the SQL-implementations have identical structure. That is, they have the same column names in the same order and with the same data types.

Table T is defined to have the same structure as each of the tables ST.

Better - define the columns here - then they could be a subset of those in the servers, and in a different order.

- 6) The Tables ST in the SQL-implementations may have columns in a different order and may contain additional columns with NULL values allowed.
- 7) <insert condition> is a boolean expression ranging over column values in the table T.

NOTE: This may involve a function defined in the Distributed Schema and returning fragment number, e.g. (f(row) = 1) if we allow functions - routines - to be defined in the DDA-distribution Schema.

General Rules

- 1) Each <conditional component> in the <component list> defines a fragment of the table T, such that the fragments are disjoint. That is, the fragments have no rows in common.
- 2) If <server table ref list> contains more than one <server table ref> then the tables in the list are replicates, each containing the same set of rows.
- 3) When a new row is inserted into the table the <condition>'s are evaluated in the order of the <component list> until one of them evaluates to TRUE. If there is no <condition> the effect is as if a condition was always TRUE. The row is then inserted into each of the tables in the corresponding <server table list>. If no <condition> evaluates to TRUE then an exception is raised.

Alternative - the last <condition> must be unconditional TRUE

- 4) An exception is raised if an attempt to insert a row into a component table fails.
- 5) A row defining table T is inserted into the DDA-distribution Schema Tables table, and
 - a) A row is inserted into the Fragment Table for each <component>
 - b) A row is inserted into the Replicated_Fragment table for each <server table ref>.
 - c) A row is inserted into the Columns table for each column defined in the <table element list>
 - d) *We could have a table to link columns that feature in the <insert condition> to the fragments - is it required?*

8 Exceptions

The DDA rules include rules that when evaluated may result in the raising of an exception.

DISCUSSION 22 – There are currently no DDA-specific conditions. This clause is included as a placemarker.

8.1 Exception codes for DDA-specific Conditions

The SQL conditions that may be raised as a result of the invocation of a DDA-distribution Schema Statement are listed in Table 2 together with the corresponding status codes.

Table 2 – SQLSTATE class and subclass values for DDA-specific conditions

Condition	Class	Subcondition	Subclass
RDA-specific condition	HZ		10001
			10002
			10003
			10004
			10005
			10006
			10007
			10008
			10009
			10010
			10011
			10012
			10013
			10014
			10015
			10016
			10017
			10018
			10019
			10020
			10021
			10022

9 Conformance

An implementation of DDA-services shall claim conformance as defined in this clause.

9.1 DDA Conformance

An implementation that claims conformance to this International Standard shall provide a DDA-information Schema, DDA-distribution Schema and DDA-distribution Statements as defined by this International Standard.

An implementation that claims conformance to this International Standard must also conform to ISO/IEC 9075 (SQL).

DISCUSSION 23 – It is expected that international review may wish to adjust this conformance statement. An alternative would be to permit separate conformance to each of DDA-information Schema, DDA-distribution Schema and DDA-distribution Statements to be made. In this case, the Conformance proforma should have a question for each of these.

9.2 Claims of Conformance

Claims of conformance to this International Standard shall be accompanied by a completed proforma for the information listed in Annex A.

Annex A Conformance Proforma

(normative)

A copy of the proforma in this annex is to be completed for each DDA implementation that claims to conform to this International Standard.

All items in 9.2 to 0 shall be completed and either the items in A.2 or **Error! Reference source not found.** according to the entry in item 0.

A.1. Identification

A.1.1 Conformance statement for: ISO/IEC SSSS:200y (E), Information Technology — Distributed Database Access for SQL.
A.1.2 Date of Statement:

A.2. Supplier Details

A.2.1 Organisation:
A.2.2 Contact Name(s):
A.2.3 Address:
A.2.4 Telephone:
A.2.5 Fax:
A.2.6 Email:
A.2.7 Other contact information:

A.3. Implementation Details

A.3.1	Implementation Name:
A.3.2	Version:
A.3.3	Hardware name:
A.3.4	Hardware version:
A.3.5	Operating system name:
A.3.6	Operating system version:
A.3.7	Special configuration requirements:
A.3.8	Other implementation information:

A.4. DDA Support

A.4.1	Specify a list of amendments implemented (or none):
A.4.2	Specify a list of technical corrigenda implemented (or none):

Annex B Extensions to RDA

(normative)

DISCUSSION 24 – There are no extensions to RDA at present

Annex C Examples of DDA architecture

(informative)

This Annex presents a number of system configurations that conform to this International Standard.

For the purposes of this Annex, the term “remote” is used informally to describe components that are connected by a communications medium, and the term “local” is used informally to describe components that are not remote. For the purposes of this Annex, RMDM diagram conventions are varied to explicitly show multiple instances of a processor for purposes of clarification.

C.1. Local SQL-servers

DDA-services may be used to combine local SQL-environments and present them as one SQL-environment to a Service User as modelled in Figure 4.

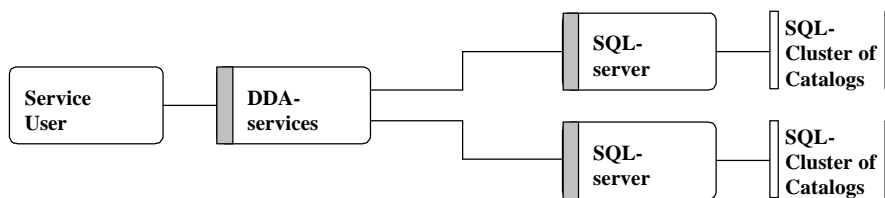


Figure 4 – Local SQL-servers

C.2. Remote SQL-servers

DDA-services may be used to combine SQL-environments which are remote from the Service User, so as to present them as one SQL-environment to a Service User as modelled in Figure 5 which shows two clients both connecting to two remote servers.

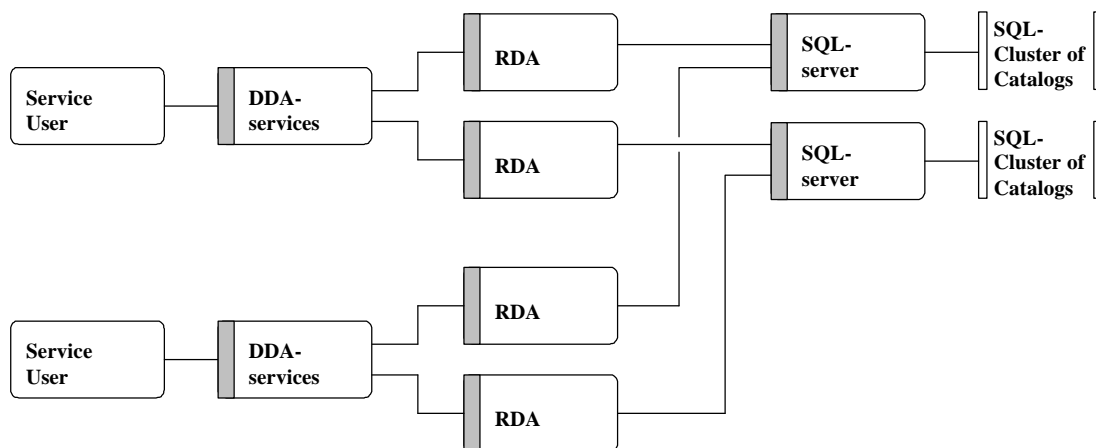


Figure 5 – Remote SQL-servers

C.3. Remote SQL-servers

An SQL-client may connect to a remote SQL-server that is itself a Distributed Database, using DDA-services to combine SQL-environments which may themselves be remote from the DDA-services so as to present them as one SQL-server to the SQL-client as modelled in Figure 6.

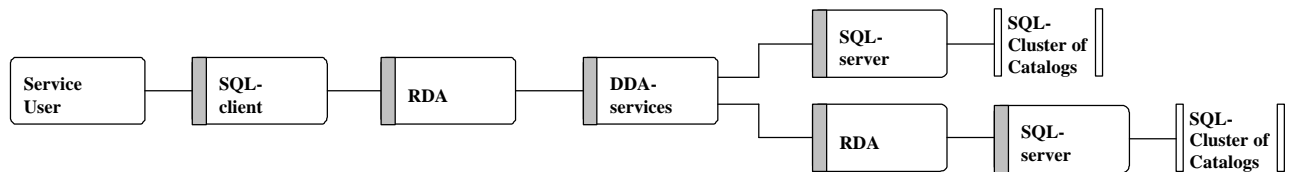


Figure 6 – Remote DDA-services

NOTE 18 – This International Standard does not require that a remote SQL-server that is distributed to contain conforming DDA-services or to use RDA to communicate with its component SQL-servers.