

Australian Standard™

**Information technology—Database
languages—SQL**

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



This Australian Standard was prepared by Committee IT-027, Data Management and Interchange. It was approved on behalf of the Council of Standards Australia on 16 May 2005. This Standard was published on 16 June 2005.

The following are represented on Committee IT-027:

Association of Superannuation Funds of Australia
Australia Post
Australia Bankers Association
Australian Bureau of Statistics
Australian Computer Society
Australian Customs Service
Australian Electoral Commission
Australian Electric and Electronic Manufacturers Association
Australian Industry Group
Australian Information Industry Association
Australian Institute of Health and Welfare
Australian Taxation Office
Centrelink
Department of Defence
Department of Immigration, Multicultural and Indigenous Affairs
Health Insurance Commission
Data Management Association Australia

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about Standards can be found by visiting the Standards Web Shop at www.standards.com.au and looking up the relevant Standard in the on-line catalogue.

Alternatively, the printed Catalogue provides information current at 1 January each year, and the monthly magazine, *The Global Standard*, has a full listing of revisions and amendments published each month.

Australian Standards™ and other products and services developed by Standards Australia are published and distributed under contract by SAI Global, which operates the Standards Web Shop.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to the Chief Executive, Standards Australia, GPO Box 5420, Sydney, NSW 2001.

This Standard was issued in draft form for comment as DR 05149.

Australian Standard™

Information technology—Database languages—SQL

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

First published as AS ISO/IEC 9075.11—2005.

COPYRIGHT

© Standards Australia

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Published by Standards Australia GPO Box 5420, Sydney, NSW 2001, Australia

ISBN 0 7337 6770 2

PREFACE

This Standard was prepared by the Standards Australia Committee IT-027, Data Management and Interchange.

This Standard is identical with, and has been reproduced from ISO/IEC 9075-11:2003, *Information technology—Database languages—SQL—Part 11: Information and Definition Schemas (SQL/Schemata)*.

The objective of this Standard is to provide database designers, administrators and developers with an information schema and definition schema that describes: SQL object identifier, structure and integrity constraints and security and authorization specifications relating to SQL-data.

This Standard is Part 11 of AS 9075, *Information technology—Database languages—SQL*, which is published in parts as follows:

- Part 1: Framework (SQL/Framework)
- Part 2: Foundation (SQL/Foundation)
- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM)
- Part 5: Host Language Bindings (SQL/Bindings)
- Part 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schemas (SQL/Schemata) (this Standard)
- Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)
- Part 14: XML-Related Specifications (SQL/XML)

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this part of ISO/IEC 9075’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian Standard</i>	
ISO/IEC		AS	
9075	Information technology—Database languages—SQL	9075	Information technology—Database languages—SQL
9075-1	Part 1: Framework (SQL/Framework)	9075.1	Part 1: Framework (SQL/Framework)
9075-2	Part 2: Foundation (SQL/Foundation)	9075.2	Part 2: Foundation (SQL/Foundation)

CONTENTS

Page

1	Scope.....	1
2	Normative references.....	3
2.1	JTC1 standards.....	3
3	Definitions, notations, and conventions.....	5
3.1	Conventions.....	5
4	Concepts.....	7
4.1	Introduction to the Definition Schema.....	7
4.2	Introduction to the Information Schema.....	8
5	Information Schema.....	9
5.1	INFORMATION_SCHEMA Schema.....	9
5.2	INFORMATION_SCHEMA_CATALOG_NAME base table.....	10
5.3	CARDINAL_NUMBER domain.....	11
5.4	CHARACTER_DATA domain.....	12
5.5	SQL_IDENTIFIER domain.....	13
5.6	TIME_STAMP domain.....	14
5.7	ADMINISTRABLE_ROLE_AUTHORIZATIONS view.....	15
5.8	APPLICABLE_ROLES view.....	16
5.9	ASSERTIONS view.....	17
5.10	ATTRIBUTES view.....	18
5.11	CHARACTER_SETS view.....	20
5.12	CHECK_CONSTRAINT_ROUTINE_USAGE view.....	21
5.13	CHECK_CONSTRAINTS view.....	22
5.14	COLLATIONS view.....	23
5.15	COLLATION_CHARACTER_SET_APPLICABILITY view.....	24
5.16	COLUMN_COLUMN_USAGE view.....	25
5.17	COLUMN_DOMAIN_USAGE view.....	26
5.18	COLUMN_PRIVILEGES view.....	27
5.19	COLUMN_UDT_USAGE view.....	28
5.20	COLUMNS view.....	29
5.21	CONSTRAINT_COLUMN_USAGE view.....	31
5.22	CONSTRAINT_TABLE_USAGE view.....	33
5.23	DATA_TYPE_PRIVILEGES view.....	35
5.24	DIRECT_SUPERTABLES view.....	37
5.25	DIRECT_SUPERTYPES view.....	38

5.26	DOMAIN_CONSTRAINTS view.....	39
5.27	DOMAINS view.....	40
5.28	ELEMENT_TYPES view.....	42
5.29	ENABLED_ROLES view.....	43
5.30	FIELDS view.....	44
5.31	KEY_COLUMN_USAGE view.....	45
5.32	METHOD_SPECIFICATION_PARAMETERS view.....	47
5.33	METHOD_SPECIFICATIONS view.....	49
5.34	PARAMETERS view.....	51
5.35	REFERENCED_TYPES view.....	53
5.36	REFERENTIAL_CONSTRAINTS view.....	54
5.37	ROLE_COLUMN_GRANTS view.....	55
5.38	ROLE_ROUTINE_GRANTS view.....	56
5.39	ROLE_TABLE_GRANTS view.....	57
5.40	ROLE_TABLE_METHOD_GRANTS view.....	58
5.41	ROLE_USAGE_GRANTS view.....	59
5.42	ROLE_UDT_GRANTS view.....	60
5.43	ROUTINE_COLUMN_USAGE view.....	61
5.44	ROUTINE_PRIVILEGES view.....	62
5.45	ROUTINE_ROUTINE_USAGE view.....	63
5.46	ROUTINE_SEQUENCE_USAGE view.....	64
5.47	ROUTINE_TABLE_USAGE view.....	65
5.48	ROUTINES view.....	66
5.49	SCHEMATA view.....	69
5.50	SEQUENCES view.....	70
5.51	SQL_FEATURES view.....	71
5.52	SQL_IMPLEMENTATION_INFO view.....	72
5.53	SQL_LANGUAGES view.....	73
5.54	SQL_PACKAGES view.....	74
5.55	SQL_PARTS view.....	75
5.56	SQL_SIZING view.....	76
5.57	SQL_SIZING_PROFILES view.....	77
5.58	TABLE_CONSTRAINTS view.....	78
5.59	TABLE_METHOD_PRIVILEGES view.....	79
5.60	TABLE_PRIVILEGES view.....	80
5.61	TABLES view.....	81
5.62	TRANSFORMS view.....	82
5.63	TRANSLATIONS view.....	83
5.64	TRIGGERED_UPDATE_COLUMNS view.....	84
5.65	TRIGGER_COLUMN_USAGE view.....	85
5.66	TRIGGER_ROUTINE_USAGE view.....	86
5.67	TRIGGER_SEQUENCE_USAGE view.....	87

5.68	TRIGGER_TABLE_USAGE view.....	88
5.69	TRIGGERS view.....	89
5.70	UDT_PRIVILEGES view.....	91
5.71	USAGE_PRIVILEGES view.....	92
5.72	USER_DEFINED_TYPES view.....	93
5.73	VIEW_COLUMN_USAGE view.....	95
5.74	VIEW_ROUTINE_USAGE view.....	96
5.75	VIEW_TABLE_USAGE view.....	97
5.76	VIEWS view.....	98
5.77	Short name views.....	99
6	Definition Schema.....	117
6.1	DEFINITION_SCHEMA Schema.....	117
6.2	EQUAL_KEY_DEGREES assertion.....	118
6.3	KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion.....	119
6.4	UNIQUE_CONSTRAINT_NAME assertion.....	120
6.5	ASSERTIONS base table.....	121
6.6	ATTRIBUTES base table.....	123
6.7	AUTHORIZATIONS base table.....	125
6.8	CATALOG_NAMES base table.....	126
6.9	CHARACTER_ENCODING_FORMS base table.....	127
6.10	CHARACTER_REPERTOIRES base table.....	128
6.11	CHARACTER_SETS base table.....	129
6.12	CHECK_COLUMN_USAGE base table.....	131
6.13	CHECK_CONSTRAINT_ROUTINE_USAGE base table.....	132
6.14	CHECK_CONSTRAINTS base table.....	133
6.15	CHECK_TABLE_USAGE base table.....	134
6.16	COLLATIONS base table.....	135
6.17	COLLATION_CHARACTER_SET_APPLICABILITY base table.....	136
6.18	COLUMN_COLUMN_USAGE base table.....	137
6.19	COLUMN_PRIVILEGES base table.....	138
6.20	COLUMNS base table.....	140
6.21	DATA_TYPE_DESCRIPTOR base table.....	144
6.22	DIRECT_SUPERTABLES base table.....	152
6.23	DIRECT_SUPERTYPES base table.....	154
6.24	DOMAIN_CONSTRAINTS base table.....	156
6.25	DOMAINS base table.....	158
6.26	ELEMENT_TYPES base table.....	159
6.27	FIELDS base table.....	161
6.28	KEY_COLUMN_USAGE base table.....	163
6.29	METHOD_SPECIFICATION_PARAMETERS base table.....	166
6.30	METHOD_SPECIFICATIONS base table.....	168
6.31	PARAMETERS base table.....	173

6.32	REFERENCED_TYPES base table.	176
6.33	REFERENTIAL_CONSTRAINTS base table.	178
6.34	ROLE_AUTHORIZATION_DESCRIPTORs base table.	181
6.35	ROUTINE_COLUMN_USAGE base table.	183
6.36	ROUTINE_PRIVILEGES base table.	185
6.37	ROUTINE_ROUTINE_USAGE base table.	187
6.38	ROUTINE_SEQUENCE_USAGE base table.	188
6.39	ROUTINE_TABLE_USAGE base table.	189
6.40	ROUTINES base table.	191
6.41	SCHEMATA base table.	199
6.42	SEQUENCES base table.	201
6.43	SQL_CONFORMANCE base table.	203
6.44	SQL_IMPLEMENTATION_INFO base table.	205
6.45	SQL_LANGUAGES base table.	206
6.46	SQL_SIZING base table.	209
6.47	SQL_SIZING_PROFILES base table.	211
6.48	TABLE_CONSTRAINTS base table.	213
6.49	TABLE_METHOD_PRIVILEGES base table.	216
6.50	TABLE_PRIVILEGES base table.	218
6.51	TABLES base table.	221
6.52	TRANSFORMS base table.	225
6.53	TRANSLATIONS base table.	227
6.54	TRIGGERED_UPDATE_COLUMNS base table.	229
6.55	TRIGGER_COLUMN_USAGE base table.	231
6.56	TRIGGER_ROUTINE_USAGE base table.	233
6.57	TRIGGER_SEQUENCE_USAGE base table.	234
6.58	TRIGGER_TABLE_USAGE base table.	235
6.59	TRIGGERS base table.	237
6.60	USAGE_PRIVILEGES base table.	240
6.61	USER_DEFINED_TYPE_PRIVILEGES base table.	242
6.62	USER_DEFINED_TYPES base table.	244
6.63	VIEW_COLUMN_USAGE base table.	248
6.64	VIEW_ROUTINE_USAGE base table.	249
6.65	VIEW_TABLE_USAGE base table.	250
6.66	VIEWS base table.	251
7	Conformance.	253
7.1	Claims of conformance to SQL/Schemata.	253
7.2	Additional conformance requirements for SQL/Schemata.	253
7.3	Implied feature relationships of SQL/Schemata.	253
Annex A	SQL Conformance Summary.	255
Annex B	Implementation-defined elements.	271

Annex C	Deprecated features.....	273
Annex D	Incompatibilities with ISO/IEC 9075:1999.....	275
Annex E	SQL feature taxonomy.....	277
Index.....		281

TABLES

Page

1	Implied feature relationships of SQL/Schemata.	253
2	Feature taxonomy and definition for mandatory features.	277
3	Feature taxonomy for optional features.	279

INTRODUCTION

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

- 1) [Clause 1, “Scope”](#), specifies the scope of this part of ISO/IEC 9075.
- 2) [Clause 2, “Normative references”](#), identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) [Clause 3, “Definitions, notations, and conventions”](#), defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) [Clause 4, “Concepts”](#), presents concepts used in the definition of Persistent SQL modules.
- 5) [Clause 5, “Information Schema”](#), defines viewed tables that contain schema information.
- 6) [Clause 6, “Definition Schema”](#), defines base tables on which the viewed tables containing schema information depend.
- 7) [Clause 7, “Conformance”](#), defines the criteria for conformance to this part of ISO/IEC 9075.
- 8) [Annex A, “SQL Conformance Summary”](#), is an informative Annex. It summarizes the conformance requirements of the SQL language.
- 9) [Annex B, “Implementation-defined elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.
- 10) [Annex C, “Deprecated features”](#), is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.
- 11) [Annex D, “Incompatibilities with ISO/IEC 9075:1999”](#), is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and ISO/IEC 9075:1999.
- 12) [Annex E, “SQL feature taxonomy”](#), is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance and may be used to develop other profiles involving the SQL language.

In the text of this part of ISO/IEC 9075, Clauses begin a new odd-numbered page, and in [Clause 5, “Information Schema”](#), through [Clause 7, “Conformance”](#), Subclauses begin a new page. Any resulting blank space is not significant.

X

NOTES

AUSTRALIAN STANDARD

Information technology — Database languages — SQL —**Part 11:****Information and Definition Schemas (SQL/Schemata)****1 Scope**

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

- The SQL object identifier of ISO/IEC 9075.
- The structure and integrity constraints of SQL-data.
- The security and authorization specifications relating to SQL-data.
- The features, subfeatures, and packages of ISO/IEC 9075, and the support that each of these has in an SQL-implementation.
- The SQL-implementation information and sizing items of ISO/IEC 9075 and the values supported by an SQL-implementation.

This page intentionally left blank.

2 Normative references

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

2.1 JTC1 standards

[Framework] ISO/IEC 9075-1:2003, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[Foundation] ISO/IEC 9075-2:2003, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

This page intentionally left blank.

3 Definitions, notations, and conventions

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

3.1 Conventions

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

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

This page intentionally left blank.

4 Concepts

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

4.1 Introduction to the Definition Schema

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

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

The Definition Schema allows information to be stored about multiple catalogs. The defined constraints guarantee consistency, not only within a single catalog, but also over all catalogs described by the Definition Schema.

The way in which certain constraints are expressed caters for the possibility that an object is being referenced that exists in a catalog that is outside the purview of the Definition Schema containing the reference in question. For example, the definition of the VIEW_TABLE_USAGE base table in Subclause 6.65, “VIEW_TABLE_USAGE base table”, includes the following constraint:

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

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

Because <unqualified schema name>s are prohibited by SR 10) of Subclause 5.4, “Names and identifiers”, in ISO/IEC 9075-2, from specifying DEFINITION_SCHEMA, the Definition Schema cannot normally be accessed in an SQL-statement. However, view definitions in the Information Schema assume the existence of the Definition Schema and reference base tables whose <schema name> is DEFINITION_SCHEMA. They use the Definition Schema to define the content of the Information Schema. Regardless of SR 14) of Subclause 5.4, “Names and identifiers”, in ISO/IEC 9075-2, the <schema name> DEFINITION_SCHEMA is never qualified by a <catalog name>. It is implementation-defined whether the DEFINITION_SCHEMA referenced by an INFORMATION_SCHEMA describes schemas in catalogs other than the catalog in which the INFORMATION_SCHEMA is located.

4.2 Introduction to the Information Schema

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

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

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

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

NOTE 1 — The Information Schema tables may be supposed to be represented in the Definition Schema in the same way as any other tables, and are hence self-describing.

NOTE 2 — The Information Schema is a definition of the SQL data model, specified as an SQL-schema, in terms of <SQL schema statement>s as defined in ISO/IEC 9075. Constraints defined in this Clause are not actual SQL constraints.

The representation of an <identifier> in the base tables and views of the Information Schema is by a character string corresponding to its <identifier body> (in the case of a <regular identifier>) or its <delimited identifier body> (in the case of a <delimited identifier>). Within this character string, any lower-case letter appearing in a <regular identifier> is replaced by the equivalent upper-case letter, and any <doublequote symbol> appearing in a <delimited identifier body> is replaced by a <double quote>. Where an <actual identifier> has multiple forms that are equal according to the rules of Subclause 8.2, “<comparison predicate>”, in ISO/IEC 9075-2, the form stored is that encountered at definition time.

5 Information Schema

5.1 INFORMATION_SCHEMA Schema

Function

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

Definition

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

Conformance Rules

None.

5.2 INFORMATION_SCHEMA_CATALOG_NAME base table

Function

Identify the catalog that contains the Information Schema.

Definition

```
CREATE TABLE INFORMATION_SCHEMA_CATALOG_NAME
( CATALOG_NAME          SQL_IDENTIFIER,

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_PRIMARY_KEY
    PRIMARY KEY ( CATALOG_NAME ),

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_CHECK
    CHECK ( 1 = ( SELECT COUNT(*)
                  FROM INFORMATION_SCHEMA_CATALOG_NAME ) ),

  CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_FK_CATALOG_NAMES
    FOREIGN KEY ( CATALOG_NAME ) REFERENCES DEFINITION_SCHEMA.CATALOG_NAMES
);

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

Description

- 1) The value of CATALOG_NAME is the name of the catalog in which this Information Schema resides.

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.
- 2) Without Feature F651, “Catalog name qualifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.INFORMATION_SCHEMA_CATALOG_NAME.

5.3 CARDINAL_NUMBER domain

Function

Define a domain that contains a non-negative number.

Definition

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

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

Description

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

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CARDINAL_NUMBER.

5.4 CHARACTER_DATA domain

Function

Define a domain that contains any character data.

Definition

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

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

Description

- 1) This domain specifies any character data.
- 2) *ML* is the implementation-defined maximum length of a variable-length character string.

Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_DATA.

5.5 SQL_IDENTIFIER domain

Function

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

Definition

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

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

Description

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

NOTE 3 — There is no way in SQL to specify a <domain constraint> that would be true for the body of any valid SQL <regular identifier> or <delimited identifier> and false for all other character string values.
- 2) *L* is the implementation-defined maximum length of <identifier body> and <delimited identifier body>.

Conformance Rules

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

5.6 TIME_STAMP domain

Function

Define a domain that contains a timestamp.

Definition

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

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

Description

- 1) The domain TIME_STAMP contains an SQL timestamp value.

Conformance Rules

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

5.7 ADMINISTRABLE_ROLE_AUTHORIZATIONS view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ADMINISTRABLE_ROLE_AUTHORIZATIONS.

5.8 APPLICABLE_ROLES view

Function

Identifies the applicable roles for the current user.

Definition

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

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

Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.APPLICABLE_ROLES.

5.9 ASSERTIONS view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F521, “Assertions”, conforming SQL language shall not reference INFORMATION_SCHEMA.ASSERTIONS.

5.10 ATTRIBUTES view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.

5.11 CHARACTER_SETS view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHARACTER_SETS.

5.12 CHECK_CONSTRAINT_ROUTINE_USAGE view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CHECK_CONSTRAINT_ROUTINE_USAGE.

5.13 CHECK_CONSTRAINTS view

Function

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

Definition

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

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

Conformance Rules

None.

5.14 COLLATIONS view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F691, “Collation and translation”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATIONS.

5.15 COLLATION_CHARACTER_SET_APPLICABILITY view

Function

Identify the character sets to which each collation is applicable.

Definition

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

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

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.
- 2) Without Feature F690, “Collation support ”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLLATION_CHARACTER_SET_APPLICABILITY.

5.16 COLUMN_COLUMN_USAGE view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- 2) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.
- 3) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_COLUMN_USAGE.

5.17 COLUMN_DOMAIN_USAGE view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE.

5.18 COLUMN_PRIVILEGES view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_PRIVILEGES.

5.19 COLUMN_UDT_USAGE view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMN_UDT_USAGE.

5.20 COLUMNS view

Function

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

Definition

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

```

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

GRANT SELECT ON TABLE COLUMNS
TO PUBLIC WITH GRANT OPTION;

```

Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.
- 2) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.IS_GENERATED.
- 3) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.GENERATION_EXPRESSION.

5.21 CONSTRAINT_COLUMN_USAGE view

Function

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

Definition

```
CREATE VIEW CONSTRAINT_COLUMN_USAGE AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
         CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
  FROM ( ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
           FROM DEFINITION_SCHEMA.CHECK_COLUMN_USAGE )
        UNION
        ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME, PK.COLUMN_NAME,
                 FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA, FK.CONSTRAINT_NAME
          FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
            JOIN
              DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS PK
            ON
              ( FK.UNIQUE_CONSTRAINT_CATALOG, FK.UNIQUE_CONSTRAINT_SCHEMA,
                FK.UNIQUE_CONSTRAINT_NAME )
              = ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                 PK.CONSTRAINT_NAME ) )
        UNION
        ( SELECT KC.TABLE_CATALOG, KC.TABLE_SCHEMA, KC.TABLE_NAME, KC.COLUMN_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS KC
            JOIN
              DEFINITION_SCHEMA.TABLE_CONSTRAINTS
            USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
            WHERE CONSTRAINT_TYPE IN
                  ( 'UNIQUE', 'PRIMARY KEY' ) ) )
  JOIN
    DEFINITION_SCHEMA.SCHEMATA
  ON
    ( ( TABLE_CATALOG, TABLE_SCHEMA )
      = ( CATALOG_NAME, SCHEMA_NAME ) )
 WHERE ( SCHEMA_OWNER = CURRENT_USER
        OR
          SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
  AND CONSTRAINT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE CONSTRAINT_COLUMN_USAGE
```

TO PUBLIC WITH GRANT OPTION;

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE.

5.22 CONSTRAINT_TABLE_USAGE view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE.

5.23 DATA_TYPE_PRIVILEGES view

Function

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

Definition

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

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
```

TO PUBLIC WITH GRANT OPTION;

Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES.

5.24 DIRECT_SUPERTABLES view

Function

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

Definition

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

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

Conformance Rules

- 1) Without Feature S081, “Subtables”, conforming SQL language shall not reference INFORMATION_SCHEMA.DIRECT_SUPERTABLES.

5.25 DIRECT_SUPERTYPES view

Function

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

Definition

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

  AND
    USER_DEFINED_TYPE_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA.CATALOG_NAME );

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

Conformance Rules

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