

**WG3:KOA-008**  
**DM32.2-2011-00067**

**ISO/IEC JTC 1/SC 32**

Date: 2011-01-03

**IWD 9075-11:2011(E)**

ISO/IEC JTC 1/SC 32/WG 3

The United States of American (ANSI)

## **Information technology — Database languages — SQL —**

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

*Technologies de l'information — Langages de base de données — SQL —  
Partie 11: Information et Definition Schémas (SQL/Schemata)*

Document type: International Standard  
Document subtype: Informal Working Draft (IWD)  
Document stage: (2) IWD = unofficial informal working drafts  
Document language: English

Edited by: Jim Melton (Ed.) and Jörn Bartels (Associate Ed.)



## Copyright notice

This ISO document is a working draft or a committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester.

*ANSI Customer Service Department  
25 West 43rd Street, 4th Floor  
New York, NY 10036  
Tele: 1-212-642-4980  
Fax: 1-212-302-1286  
Email: [storemanager@ansi.org](mailto:storemanager@ansi.org)  
Web: [www.ansi.org](http://www.ansi.org)*

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

# Contents

Page

Foreword.....	ix
Introduction.....	x
<b>1 Scope.....</b>	<b>1</b>
<b>2 Normative references.....</b>	<b>3</b>
2.1 ISO and IEC standards.....	3
<b>3 Definitions, notations, and conventions.....</b>	<b>5</b>
3.1 Conventions.....	5
<b>4 Concepts.....</b>	<b>7</b>
4.1 Introduction to the Definition Schema.....	7
4.2 Introduction to the Information Schema.....	8
<b>5 Information Schema.....</b>	<b>9</b>
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 YES_OR_NO domain.....	15
5.8 ADMINISTRABLE_ROLE_AUTHORIZATIONS view.....	16
5.9 APPLICABLE_ROLES view.....	17
5.10 ASSERTIONS view.....	18
5.11 ATTRIBUTES view.....	19
5.12 CHARACTER_SETS view.....	21
5.13 CHECK_CONSTRAINT_ROUTINE_USAGE view.....	22
5.14 CHECK_CONSTRAINTS view.....	23
5.15 COLLATIONS view.....	24
5.16 COLLATION_CHARACTER_SET_APPLICABILITY view.....	25
5.17 COLUMN_COLUMN_USAGE view.....	26
5.18 COLUMN_DOMAIN_USAGE view.....	27
5.19 COLUMN_PRIVILEGES view.....	28
5.20 COLUMN_UDT_USAGE view.....	29
5.21 COLUMNS view.....	30
5.22 CONSTRAINT_COLUMN_USAGE view.....	33
5.23 CONSTRAINT_PERIOD_USAGE view.....	35
5.24 CONSTRAINT_TABLE_USAGE view.....	37

5.25	DATA_TYPE_PRIVILEGES view.....	39
5.26	DIRECT_SUPERTABLES view.....	41
5.27	DIRECT_SUPERTYPES view.....	42
5.28	DOMAIN_CONSTRAINTS view.....	43
5.29	DOMAINS view.....	44
5.30	ELEMENT_TYPES view.....	46
5.31	ENABLED_ROLES view.....	48
5.32	FIELDS view.....	49
5.33	KEY_COLUMN_USAGE view.....	51
5.34	KEY_PERIOD_USAGE view.....	53
5.35	METHOD_SPECIFICATION_PARAMETERS view.....	54
5.36	METHOD_SPECIFICATIONS view.....	56
5.37	PARAMETERS view.....	59
5.38	PERIODS view.....	61
5.39	REFERENCED_TYPES view.....	63
5.40	REFERENTIAL_CONSTRAINTS view.....	64
5.41	ROLE_COLUMN_GRANTS view.....	65
5.42	ROLE_ROUTINE_GRANTS view.....	66
5.43	ROLE_TABLE_GRANTS view.....	67
5.44	ROLE_TABLE_METHOD_GRANTS view.....	68
5.45	ROLE_USAGE_GRANTS view.....	69
5.46	ROLE_UDT_GRANTS view.....	70
5.47	ROUTINE_COLUMN_USAGE view.....	71
5.48	ROUTINE_PERIOD_USAGE view.....	72
5.49	ROUTINE_PRIVILEGES view.....	73
5.50	ROUTINE_ROUTINE_USAGE view.....	74
5.51	ROUTINE_SEQUENCE_USAGE view.....	75
5.52	ROUTINE_TABLE_USAGE view.....	76
5.53	ROUTINES view.....	77
5.54	SCHEMATA view.....	80
5.55	SEQUENCES view.....	81
5.56	SQL_FEATURES view.....	82
5.57	SQL_IMPLEMENTATION_INFO view.....	83
5.58	SQL_PACKAGES view.....	84
5.59	SQL_PARTS view.....	85
5.60	SQL_SIZING view.....	86
5.61	TABLE_CONSTRAINTS view.....	87
5.62	TABLE_METHOD_PRIVILEGES view.....	88
5.63	TABLE_PRIVILEGES view.....	89
5.64	TABLES view.....	90
5.65	TRANSFORMS view.....	91
5.66	TRANSLATIONS view.....	92
5.67	TRIGGERED_UPDATE_COLUMNS view.....	93
5.68	TRIGGER_COLUMN_USAGE view.....	94

5.69	TRIGGER_PERIOD_USAGE view.....	95
5.70	TRIGGER_ROUTINE_USAGE view.....	96
5.71	TRIGGER_SEQUENCE_USAGE view.....	97
5.72	TRIGGER_TABLE_USAGE view.....	98
5.73	TRIGGERS view.....	99
5.74	UDT_PRIVILEGES view.....	101
5.75	USAGE_PRIVILEGES view.....	102
5.76	USER_DEFINED_TYPES view.....	103
5.77	VIEW_COLUMN_USAGE view.....	105
5.78	VIEW_PERIOD_USAGE view.....	106
5.79	VIEW_ROUTINE_USAGE view.....	107
5.80	VIEW_TABLE_USAGE view.....	108
5.81	VIEWS view.....	109
5.82	Short name views.....	110
<b>6</b>	<b>Definition Schema.....</b>	<b>129</b>
6.1	DEFINITION_SCHEMA Schema.....	129
6.2	EQUAL_KEY_DEGREES assertion.....	130
6.3	KEY_DEGREE_GREATER_THAN_OR_EQUAL_TO_1 assertion.....	131
6.4	UNIQUE_CONSTRAINT_NAME assertion.....	132
6.5	ASSERTIONS base table.....	133
6.6	ATTRIBUTES base table.....	135
6.7	AUTHORIZATIONS base table.....	137
6.8	CATALOG_NAME base table.....	138
6.9	CHARACTER_ENCODING_FORMS base table.....	139
6.10	CHARACTER_REPERTOIRES base table.....	140
6.11	CHARACTER_SETS base table.....	141
6.12	CHECK_COLUMN_USAGE base table.....	143
6.13	CHECK_CONSTRAINT_ROUTINE_USAGE base table.....	144
6.14	CHECK_CONSTRAINTS base table.....	145
6.15	CHECK_PERIOD_USAGE base table.....	146
6.16	CHECK_TABLE_USAGE base table.....	147
6.17	COLLATIONS base table.....	148
6.18	COLLATION_CHARACTER_SET_APPLICABILITY base table.....	149
6.19	COLUMN_COLUMN_USAGE base table.....	150
6.20	COLUMN_PRIVILEGES base table.....	151
6.21	COLUMNS base table.....	153
6.22	DATA_TYPE_DESCRIPTOR base table.....	158
6.23	DIRECT_SUPERTABLES base table.....	168
6.24	DIRECT_SUPERTYPES base table.....	170
6.25	DOMAIN_CONSTRAINTS base table.....	172
6.26	DOMAINS base table.....	174
6.27	ELEMENT_TYPES base table.....	175
6.28	FIELDS base table.....	177

6.29	KEY_COLUMN_USAGE base table. ....	179
6.30	KEY_PERIOD_USAGE base table. ....	181
6.31	METHOD_SPECIFICATION_PARAMETERS base table. ....	183
6.32	METHOD_SPECIFICATIONS base table. ....	185
6.33	PARAMETERS base table. ....	190
6.34	PERIODS base table. ....	193
6.35	REFERENCED_TYPES base table. ....	194
6.36	REFERENTIAL_CONSTRAINTS base table. ....	196
6.37	ROLE_AUTHORIZATION_DESCRIPTORs base table. ....	199
6.38	ROUTINE_COLUMN_USAGE base table. ....	201
6.39	ROUTINE_PERIOD_USAGE base table. ....	202
6.40	ROUTINE_PRIVILEGES base table. ....	203
6.41	ROUTINE_ROUTINE_USAGE base table. ....	205
6.42	ROUTINE_SEQUENCE_USAGE base table. ....	206
6.43	ROUTINE_TABLE_USAGE base table. ....	207
6.44	ROUTINES base table. ....	208
6.45	SCHEMATA base table. ....	215
6.46	SEQUENCES base table. ....	217
6.47	SQL_CONFORMANCE base table. ....	219
6.48	SQL_IMPLEMENTATION_INFO base table. ....	221
6.49	SQL_SIZING base table. ....	222
6.50	TABLE_CONSTRAINTS base table. ....	223
6.51	TABLE_METHOD_PRIVILEGES base table. ....	226
6.52	TABLE_PRIVILEGES base table. ....	228
6.53	TABLES base table. ....	231
6.54	TRANSFORMS base table. ....	235
6.55	TRANSLATIONS base table. ....	237
6.56	TRIGGERED_UPDATE_COLUMNS base table. ....	239
6.57	TRIGGER_COLUMN_USAGE base table. ....	240
6.58	TRIGGER_PERIOD_USAGE base table. ....	241
6.59	TRIGGER_ROUTINE_USAGE base table. ....	242
6.60	TRIGGER_SEQUENCE_USAGE base table. ....	243
6.61	TRIGGER_TABLE_USAGE base table. ....	244
6.62	TRIGGERS base table. ....	245
6.63	USAGE_PRIVILEGES base table. ....	248
6.64	USER_DEFINED_TYPE_PRIVILEGES base table. ....	250
6.65	USER_DEFINED_TYPES base table. ....	252
6.66	VIEW_COLUMN_USAGE base table. ....	256
6.67	VIEW_PERIOD_USAGE base table. ....	257
6.68	VIEW_ROUTINE_USAGE base table. ....	258
6.69	VIEW_TABLE_USAGE base table. ....	259
6.70	VIEWS base table. ....	260
<b>7</b>	<b>Conformance. ....</b>	<b>263</b>

7.1	Claims of conformance to SQL/Schemata. ....	263
7.2	Additional conformance requirements for SQL/Schemata. ....	263
7.3	Implied feature relationships of SQL/Schemata. ....	263
<b>Annex A</b> (informative)	<b>SQL Conformance Summary. ....</b>	<b>265</b>
<b>Annex B</b> (informative)	<b>Implementation-defined elements. ....</b>	<b>287</b>
<b>Annex C</b> (informative)	<b>Implementation-dependent elements. ....</b>	<b>289</b>
<b>Annex D</b> (informative)	<b>Deprecated features. ....</b>	<b>291</b>
<b>Annex E</b> (informative)	<b>Incompatibilities with ISO/IEC 9075:2008. ....</b>	<b>293</b>
<b>Annex F</b> (informative)	<b>SQL feature taxonomy. ....</b>	<b>295</b>
<b>Annex G</b> (informative)	<b>Defect reports not addressed in this edition of this part of ISO/IEC 9075. ...</b>	<b>299</b>
<b>Index. ....</b>		<b>301</b>

Tables

Table	Page
1 Implied feature relationships of SQL/Schemata. . . . .	263
2 Feature taxonomy and definition for mandatory features. . . . .	296
3 Feature taxonomy for optional features. . . . .	297



## Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9075-11 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This third edition of ISO/IEC 9075-11 cancels and replaces the second edition (ISO/IEC 9075-11:2008), which has been technically revised. It also incorporates Technical Corrigendum ISO/IEC 9075-11:2008/Cor.1:2010.

ISO/IEC 9075 consists of the following parts, under the general title *Information technology — Database languages — SQL*:

- 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 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schema (SQL/Schemata)
- Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)
- Part 14: XML-Related Specifications (SQL/XML)

NOTE 1 — The individual parts of multi-part standards are not necessarily published together. New editions of one or more parts may be published without publication of new editions of other parts.

## Introduction

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

- 1) [Clause 1, “Scope”](#), specifies the scope of this part of ISO/IEC 9075.
- 2) [Clause 2, “Normative references”](#), identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) [Clause 3, “Definitions, notations, and conventions”](#), defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) [Clause 4, “Concepts”](#), presents concepts used in the definition of Persistent SQL modules.
- 5) [Clause 5, “Information Schema”](#), defines viewed tables that contain schema information.
- 6) [Clause 6, “Definition Schema”](#), defines base tables on which the viewed tables containing schema information depend.
- 7) [Clause 7, “Conformance”](#), defines the criteria for conformance to this part of ISO/IEC 9075.
- 8) [Annex A, “SQL Conformance Summary”](#), is an informative Annex. It summarizes the conformance requirements of the SQL language.
- 9) [Annex B, “Implementation-defined elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.
- 10) [Annex C, “Implementation-dependent elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.
- 11) [Annex D, “Deprecated features”](#), is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.
- 12) [Annex E, “Incompatibilities with ISO/IEC 9075:2008”](#), is an informative Annex. It lists incompatibilities with the previous version of this part of ISO/IEC 9075.
- 13) [Annex F, “SQL feature taxonomy”](#), is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by an identifier and a short descriptive name. This taxonomy is used to specify conformance.
- 14) [Annex G, “Defect reports not addressed in this edition of this part of ISO/IEC 9075”](#), is an informative Annex. It describes the Defect Reports that were known at the time of publication of this part of this International Standard. Each of these problems is a problem carried forward from the previous edition of ISO/IEC 9075. No new problems have been created in the drafting of this edition of this International Standard.

In the text of this part of ISO/IEC 9075, Clauses 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.

## **Information technology — Database languages — SQL —**

Part 11:

### **Information and Definition Schemas (SQL/Schemata)**

#### **1 Scope**

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

- The structure and integrity constraints of SQL-data.
- The security and authorization specifications relating to SQL-data.
- The features, 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.

*(Blank page)*

## 2 Normative references

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

### 2.1 ISO and IEC standards

[ISO9075-1] ISO/IEC 9075-1:2011, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[ISO9075-2] ISO/IEC 9075-2:2011, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

*(Blank page)*

### 3 Definitions, notations, and conventions

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

#### 3.1 Conventions

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

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

*(Blank page)*



## 4 Concepts

*This Clause modifies [Clause 4](#), “*Concepts*”, in ISO/IEC 9075-2.*

### 4.1 Introduction to the Definition Schema

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

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

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

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 6.69](#), “[VIEW\\_TABLE\\_USAGE base table](#)”, includes the following constraint:

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

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

Because <unqualified schema name>s are prohibited by [SR 10](#)) of [Subclause 5.4](#), “[Names and identifiers](#)”, in [ISO9075-2], from specifying DEFINITION\_SCHEMA, the Definition Schema cannot normally be accessed in an SQL-statement. However, view definitions in the Information Schema assume the existence of the Definition Schema and reference base tables whose <schema name> is DEFINITION\_SCHEMA. They use the Definition Schema to define the content of the Information Schema. Regardless of [SR 14](#)) of [Subclause 5.4](#), “[Names and identifiers](#)”, in [ISO9075-2], the <schema name> DEFINITION\_SCHEMA is never qualified by a <catalog name>. It is implementation-defined whether the DEFINITION\_SCHEMA referenced by an

## 4.1 Introduction to the Definition Schema

INFORMATION\_SCHEMA describes schemas in catalogs other than the catalog in which the INFORMATION\_SCHEMA is located.

## 4.2 Introduction to the Information Schema

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

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

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

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

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

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

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

## 5 Information Schema

*This Clause is modified by [Clause 18](#), “Information Schema”, in ISO/IEC 9075-4.*

*This Clause is modified by [Clause 24](#), “Information Schema”, in ISO/IEC 9075-9.*

*This Clause is modified by [Clause 13](#), “Information Schema”, in ISO/IEC 9075-13.*

*This Clause is modified by [Clause 20](#), “Information Schema”, in ISO/IEC 9075-14.*

### 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_NAME
    FOREIGN KEY ( CATALOG_NAME ) REFERENCES DEFINITION_SCHEMA.CATALOG_NAME
);

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

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

### Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IDENTIFIER.

## 5.6 TIME\_STAMP domain

### Function

Define a domain that contains a timestamp.

### Definition

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

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

### Description

- 1) The domain TIME\_STAMP contains an SQL timestamp value.

### Conformance Rules

- 1) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TIME\_STAMP.



## 5.7 YES\_OR\_NO domain

### Function

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

### Definition

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

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

### Description

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

### Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.YES\_OR\_NO.

## 5.8 ADMINISTRABLE\_ROLE\_AUTHORIZATIONS view

### Function

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

### Definition

```
CREATE VIEW ADMINISTRABLE_ROLE_AUTHORIZATIONS AS
  SELECT GRANTEE, ROLE_NAME, IS_GRANTABLE
  FROM DEFINITION_SCHEMA.ROLE_AUTHORIZATION_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.

## 5.9 APPLICABLE\_ROLES view

### Function

Identifies the applicable roles for the current user.

### Definition

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

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

### Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.APPLICABLE\_ROLES.

## **5.10 ASSERTIONS view**

### **Function**

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

### **Definition**

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

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

### **Conformance Rules**

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

## 5.11 ATTRIBUTES view

*This Subclause is modified by Subclause 24.1, “ATTRIBUTES view”, in ISO/IEC 9075-9.*  
*This Subclause is modified by Subclause 20.3, “ATTRIBUTES view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

**IWD 9075-11:2011(E)**  
**5.11 ATTRIBUTES view**

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.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.12 CHARACTER\_SETS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHARACTER\_SETS.

## 5.13 CHECK\_CONSTRAINT\_ROUTINE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHECK\_CONSTRAINT\_ROUTINE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHECK\_CONSTRAINT\_ROUTINE\_USAGE.



## 5.14 CHECK\_CONSTRAINTS view

### Function

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

### Definition

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

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

### Conformance Rules

*None.*

## **5.15 COLLATIONS view**

### **Function**

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

### **Definition**

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

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

### **Conformance Rules**

- 1) Without Feature F690, “Collation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS.

## 5.16 COLLATION\_CHARACTER\_SET\_APPLICABILITY view

### Function

Identify the character sets to which each collation is applicable.

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATION\_CHARACTER\_SET\_APPLICABILITY.
- 2) Without Feature F690, “Collation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATION\_CHARACTER\_SET\_APPLICABILITY.

## 5.17 COLUMN\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature T175, “Generated columns”, conforming SQL language shall not reference 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.

## 5.18 COLUMN\_DOMAIN\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.
- 2) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.
- 3) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.

## 5.19 COLUMN\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_PRIVILEGES.

## 5.20 COLUMN\_UDT\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_UDT\_USAGE.

## 5.21 COLUMNS view

*This Subclause is modified by Subclause 24.3, “COLUMNS view”, in ISO/IEC 9075-9.  
This Subclause is modified by Subclause 20.4, “COLUMNS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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



```

COALESCE (D1.SCOPE_NAME, D2.SCOPE_NAME) AS SCOPE_NAME,
COALESCE (D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY)
    AS MAXIMUM_CARDINALITY,
COALESCE (D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER)
    AS DTD_IDENTIFIER,
IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
IDENTITY_START, IDENTITY_INCREMENT,
IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE,

IS_GENERATED, GENERATION_EXPRESSION, IS_TIME_PERIOD_VERSION_START,
IS_SYSTEM_TIME_PERIOD_END, SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION,

IS_UPDATABLE,
COALESCE (D1.DECLARED_DATA_TYPE, D2.DECLARED_DATA_TYPE)
    AS DECLARED_DATA_TYPE,
COALESCE (D1.DECLARED_NUMERIC_PRECISION, D2.DECLARED_NUMERIC_PRECISION)
    AS DECLARED_NUMERIC_PRECISION,
COALESCE (D1.DECLARED_NUMERIC_SCALE, D2.DECLARED_NUMERIC_SCALE)
    AS DECLARED_NUMERIC_SCALE
FROM ( ( DEFINITION_SCHEMA.COLUMNS AS C
    LEFT JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
    ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
        'TABLE', C.DTD_IDENTIFIER )
        = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
        D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) ) )
    LEFT JOIN
        DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
    ON ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
        'DOMAIN', C.DTD_IDENTIFIER )
        = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
        D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME ) IN
    ( SELECT CP.TABLE_CATALOG, CP.TABLE_SCHEMA, CP.TABLE_NAME, CP.COLUMN_NAME
    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES AS CP
    WHERE ( CP.GRANTEE IN
        ( 'PUBLIC', CURRENT_USER )
        OR
        CP.GRANTEE IN
        ( SELECT ROLE_NAME
        FROM ENABLED_ROLES ) ) )
AND
    C.TABLE_CATALOG
    = ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA.CATALOG_NAME );

GRANT SELECT ON TABLE COLUMNS
    TO PUBLIC WITH GRANT OPTION;

```

## Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.

**IWD 9075-11:2011(E)**  
**5.21 COLUMNS view**

- 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.
- 4) Without Feature T111, “Updatable joins, unions, and columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.IS\_UPDATABLE.
- 5) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION, and DECLARED\_NUMERIC\_SCALE.
- 6) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference any of the columns IS\_SYSTEM\_TIME\_PERIOD\_START, IS\_SYSTEM\_TIME\_PERIOD\_END, and SYSTEM\_TIME\_PERIOD\_TIMESTAMP\_GENERATION.

## 5.22 CONSTRAINT\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

## **Conformance Rules**

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_COLUMN\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_COLUMN\_USAGE.

## 5.23 CONSTRAINT\_PERIOD\_USAGE view

### Function

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

### Definition

```
CREATE VIEW CONSTRAINT_PERIOD_USAGE AS
  SELECT AC.TABLE_CATALOG, AC.TABLE_SCHEMA, AC.TABLE_NAME,
         AC.PERIOD_NAME, AC.CONSTRAINT_CATALOG, AC.CONSTRAINT_SCHEMA,
         AC.CONSTRAINT_NAME
  FROM ( ( SELECT CPU.TABLE_CATALOG, CPU.TABLE_SCHEMA, CPU.TABLE_NAME, CPU.PERIOD_NAME,
                  CPU.CONSTRAINT_CATALOG, CPU.CONSTRAINT_SCHEMA, CPU.CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.CHECK_PERIOD_USAGE AS CPU )
        UNION
        ( SELECT KPU.TABLE_CATALOG, KPU.TABLE_SCHEMA, KPU.TABLE_NAME, KPU.PERIOD_NAME,
              RC.CONSTRAINT_CATALOG, RC.CONSTRAINT_SCHEMA, RC.CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
              JOIN
                DEFINITION_SCHEMA.KEY_PERIOD_USAGE AS KPU
              ON
                ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
                  RC.UNIQUE_CONSTRAINT_NAME )
                = ( KPU.CONSTRAINT_CATALOG, KPU.CONSTRAINT_SCHEMA,
                    KPU.CONSTRAINT_NAME ) ) )
        UNION
        ( SELECT KPU.TABLE_CATALOG, KPU.TABLE_SCHEMA, KPU.TABLE_NAME, KPU.PERIOD_NAME,
              CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.KEY_PERIOD_USAGE AS KPU
              JOIN
                DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
              USING ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME )
            WHERE TC.CONSTRAINT_TYPE IN
                  ( 'UNIQUE', 'PRIMARY KEY' ) ) ) AS AC (TABLE_CATALOG, TABLE_SCHEMA,
                                                         TABLE_NAME, PERIOD_NAME,
                                                         CONSTRAINT_CATALOG,
                                                         CONSTRAINT_SCHEMA,
                                                         CONSTRAINT_NAME)

  JOIN
    DEFINITION_SCHEMA.SCHEMATA AS S
  ON
    ( ( AC.TABLE_CATALOG, AC.TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
 WHERE ( S.SCHEMA_OWNER = CURRENT_USER
        OR
        S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
 AND AC.CONSTRAINT_CATALOG
    = ( SELECT ISCN.CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );
```

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

## **Conformance Rules**

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.
- 3) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.

## 5.24 CONSTRAINT\_TABLE\_USAGE view

### Function

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

### Definition

```
CREATE VIEW CONSTRAINT_TABLE_USAGE AS
  SELECT AC.TABLE_CATALOG, AC.TABLE_SCHEMA, AC.TABLE_NAME,
         AC.CONSTRAINT_CATALOG, AC.CONSTRAINT_SCHEMA, AC.CONSTRAINT_NAME
  FROM ( ( SELECT CTU.TABLE_CATALOG, CTU.TABLE_SCHEMA, CTU.TABLE_NAME,
                  CTU.CONSTRAINT_CATALOG, CTU.CONSTRAINT_SCHEMA, CTU.CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.CHECK_TABLE_USAGE AS CTU )
        UNION
        ( SELECT TC.TABLE_CATALOG, TC.TABLE_SCHEMA, TC.TABLE_NAME,
                  RC.CONSTRAINT_CATALOG, RC.CONSTRAINT_SCHEMA, RC.CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
            JOIN
              DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
            ON ( RC.UNIQUE_CONSTRAINT_CATALOG, RC.UNIQUE_CONSTRAINT_SCHEMA,
                 RC.UNIQUE_CONSTRAINT_NAME )
              = ( TC.CONSTRAINT_CATALOG, TC.CONSTRAINT_SCHEMA,
                 TC.CONSTRAINT_NAME ) ) )
        UNION
        ( SELECT TC.TABLE_CATALOG, TC.TABLE_SCHEMA, TC.TABLE_NAME,
                  TC.CONSTRAINT_CATALOG, TC.CONSTRAINT_SCHEMA, TC.CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS AS TC
            WHERE TC.CONSTRAINT_TYPE IN
                  ( 'UNIQUE', 'PRIMARY KEY' ) ) ) AS AC (TABLE_CATALOG, TABLE_SCHEMA,
                                                         TABLE_NAME, CONSTRAINT_CATALOG,
                                                         CONSTRAINT_SCHEMA,
                                                         CONSTRAINT_NAME)

  JOIN
    DEFINITION_SCHEMA.SCHEMATA AS S
  ON ( ( AC.TABLE_CATALOG, AC.TABLE_SCHEMA )
      = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
WHERE ( S.SCHEMA_OWNER = CURRENT_USER
      OR
        S.SCHEMA_OWNER IN
          ( SELECT ER.ROLE_NAME
            FROM ENABLED_ROLES AS ER ) )
AND CONSTRAINT_CATALOG
  = ( SELECT ISCN.CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME AS ISCN );

GRANT SELECT ON TABLE CONSTRAINT_TABLE_USAGE
```

TO PUBLIC WITH GRANT OPTION;

## **Conformance Rules**

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE.



## 5.25 DATA\_TYPE\_PRIVILEGES view

### Function

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

### Definition

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

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
```

TO PUBLIC WITH GRANT OPTION;

## **Conformance Rules**

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DATA\_TYPE\_PRIVILEGES.

## 5.26 DIRECT\_SUPERTABLES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S081, “Subtables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DIRECT\_SUPERTABLES.

## 5.27 DIRECT\_SUPERTYPES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DIRECT\_SUPERTYPES.

## 5.28 DOMAIN\_CONSTRAINTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAIN\_CONSTRAINTS.

## 5.29 DOMAINS view

*This Subclause is modified by Subclause 20.5, “DOMAINS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

GRANT SELECT ON TABLE DOMAINS
```

TO PUBLIC WITH GRANT OPTION;

## **Conformance Rules**

- 1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.30 ELEMENT\_TYPES view

*This Subclause is modified by Subclause 20.6, “ELEMENT\_TYPES view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES.



- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.31 ENABLED\_ROLES view

### Function

Identify the enabled roles for the current SQL-session.

### Definition

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

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

### Conformance Rules

- 1) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ENABLED\_ROLES.

## 5.32 FIELDS view

*This Subclause is modified by Subclause 20.7, “FIELDS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS.

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

## 5.33 KEY\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

## **Conformance Rules**

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE.

## 5.34 KEY\_PERIOD\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.
- 3) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.

## 5.35 METHOD\_SPECIFICATION\_PARAMETERS view

*This Subclause is modified by Subclause 20.8, “METHOD\_SPECIFICATION\_PARAMETERS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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



**5.35 METHOD\_SPECIFICATION\_PARAMETERS view**

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

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

**Conformance Rules**

- 1) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATION\_PARAMETERS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATION\_PARAMETERS.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.36 METHOD\_SPECIFICATIONS view

*This Subclause is modified by Subclause 13.3, “METHOD\_SPECIFICATIONS view”, in ISO/IEC 9075-13.*

*This Subclause is modified by Subclause 20.9, “METHOD\_SPECIFICATIONS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

```

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

GRANT SELECT ON TABLE METHOD_SPECIFICATIONS
  TO PUBLIC WITH GRANT OPTION;

```

NOTE 5 — The METHOD\_SPECIFICATIONS view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with “RESULT\_CAST\_” describes the data type specified in the <result cast>, if any, contained in the <method specification>, the other set of columns describes the data type specified in the <returns data type> contained in the <method specification>.

## Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.
- 2) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.
- 3) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.CREATED.

- 4) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.37 PARAMETERS view

*This Subclause is modified by Subclause 18.5, “PARAMETERS view”, in ISO/IEC 9075-4.*

*This Subclause is modified by Subclause 20.10, “PARAMETERS view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

```

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

GRANT SELECT ON TABLE PARAMETERS
TO PUBLIC WITH GRANT OPTION;

```

## Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS.
- 2) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.
- 3) Without Feature T522, “Default values for IN parameters of SQL-invoked procedures”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS.PARAMETER\_DEFAULT.

## 5.38 PERIODS view

### Function

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

### Definition

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

```

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

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

## **Conformance Rules**

- 1) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PERIODS.
- 2) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PERIODS.



## 5.39 REFERENCED\_TYPES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.40 REFERENTIAL\_CONSTRAINTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENTIAL\_CONSTRAINTS.

## 5.41 ROLE\_COLUMN\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_COLUMN\_GRANTS.

## 5.42 ROLE\_ROUTINE\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.
- 3) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.

## 5.43 ROLE\_TABLE\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_GRANTS.

## 5.44 ROLE\_TABLE\_METHOD\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.
- 2) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.
- 3) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.

## 5.45 ROLE\_USAGE\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_USAGE\_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_USAGE\_GRANTS.

## 5.46 ROLE\_UDT\_GRANTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_UDT\_GRANTS.
- 2) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_UDT\_GRANTS.



## 5.47 ROUTINE\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COLUMN\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COLUMN\_USAGE.

## 5.48 ROUTINE\_PERIOD\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- 3) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- 4) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.

## 5.49 ROUTINE\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PRIVILEGES.

## 5.50 ROUTINE\_ROUTINE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_ROUTINE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_ROUTINE\_USAGE.

## 5.51 ROUTINE\_SEQUENCE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.
- 3) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.

## 5.52 ROUTINE\_TABLE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_TABLE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_TABLE\_USAGE.

## 5.53 ROUTINES view

*This Subclause is modified by Subclause 18.7, “ROUTINES view”, in ISO/IEC 9075-4.  
This Subclause is modified by Subclause 20.11, “ROUTINES view”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

**IWD 9075-11:2011(E)**  
**5.53 ROUTINES view**

```

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

GRANT SELECT ON TABLE ROUTINES
TO PUBLIC WITH GRANT OPTION;

```

NOTE 6 — The ROUTINES view contains two sets of columns that each describe a data type. While the set of columns that are prefixed with “RESULT\_CAST\_” describes the data type specified in the <result cast>, if any, contained in the <SQL-invoked



routine>, the other set of columns describes the data type specified in the <returns data type> contained in the <SQL-invoked routine>.

## **Conformance Rules**

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.
- 2) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.CREATED.
- 3) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.LAST\_ALTERED.
- 4) Without Feature T272, “Enhanced savepoint management”, conforming SQL-language shall not reference INFORMATION\_SCHEMA.ROUTINES.NEW\_SAVEPOINT\_LEVEL.
- 5) Without Feature T322, “Declared data type attributes”, conforming SQL-language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION, DECLARED\_NUMERIC\_SCALE, RESULT\_CAST\_FROM\_DECLARED\_DATA\_TYPE, RESULT\_CAST\_DECLARED\_NUMERIC\_PRECISION and RESULT\_CAST\_DECLARED\_NUMERIC\_SCALE.

## **5.54 SCHEMATA view**

### **Function**

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

### **Definition**

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

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

### **Conformance Rules**

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SCHEMATA.

## 5.55 SEQUENCES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.56 SQL\_FEATURES view

### Function

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

### Definition

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

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

### Conformance Rules

*None.*

## 5.57 SQL\_IMPLEMENTATION\_INFO view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPLEMENTATION\_INFO.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPLEMENTATION\_INFO.

## 5.58 SQL\_PACKAGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_PACKAGES.

## 5.59 SQL\_PARTS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_PARTS.

## **5.60 SQL\_SIZING view**

### **Function**

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

### **Definition**

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

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

### **Conformance Rules**

*None.*



## 5.61 TABLE\_CONSTRAINTS view

### Function

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

### Definition

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

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

### Conformance Rules

*None.*

## 5.62 TABLE\_METHOD\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVILEGES.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVILEGES.

## 5.63 TABLE\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_PRIVILEGES.

## 5.64 TABLES view

### Function

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

### Definition

```
CREATE VIEW TABLES AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_TYPE,
         SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLES.

## 5.65 TRANSFORMS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature S241, “Transform functions”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSFORMS.

## 5.66 TRANSLATIONS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS.
- 2) Without Feature F695, “Translation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS.
- 3) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATION\_SOURCE\_CATALOG, TRANSLATION\_SOURCE\_SCHEMA, or TRANSLATION\_SOURCE\_NAME.

## 5.67 TRIGGERED\_UPDATE\_COLUMNS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_UPDATE\_COLUMNS.
- 2) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_UPDATE\_COLUMNS.

## 5.68 TRIGGER\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_COLUMN\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_COLUMN\_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_COLUMN\_USAGE.



## 5.69 TRIGGER\_PERIOD\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_PERIOD\_USAGE.
- 3) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.
- 4) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.

## 5.70 TRIGGER\_ROUTINE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.

## 5.71 TRIGGER\_SEQUENCE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.
- 3) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.
- 4) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.

## 5.72 TRIGGER\_TABLE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.

## 5.73 TRIGGERS view

### Function

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

### Definition

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

**IWD 9075-11:2011(E)**  
**5.73 TRIGGERS view**

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

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

## Conformance Rules

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.
- 2) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.TRIGGER\_CREATED.
- 3) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.

## 5.74 UDT\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.UDT\_PRIVILEGES.

## 5.75 USAGE\_PRIVILEGES view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USAGE\_PRIVILEGES.



## 5.76 USER\_DEFINED\_TYPES view

*This Subclause is modified by Subclause 13.6, “USER\_DEFINED\_TYPES view”, in ISO/IEC 9075-13.*

### Function

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

### Definition

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

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

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

## **Conformance Rules**

- 1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USER\_DEFINED\_TYPES.
- 2) Without Feature S401, “Distinct types based on array types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USER\_DEFINED\_TYPES.MAXIMUM\_CARDINALITY.
- 3) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.

## 5.77 VIEW\_COLUMN\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_COLUMN\_USAGE.

## 5.78 VIEW\_PERIOD\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- 3) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- 4) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.

## 5.79 VIEW\_ROUTINE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_ROUTINE\_USAGE.

## 5.80 VIEW\_TABLE\_USAGE view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_TABLE\_USAGE.

## 5.81 VIEWS view

### Function

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

### Definition

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

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

### Conformance Rules

- 1) Without Feature T213, “INSTEAD OF triggers”, conforming SQL language shall not reference any of the columns IS\_TRIGGER\_UPDATABLE, IS\_TRIGGER\_DELETABLE, IS\_TRIGGER\_INSERTABLE\_INTO.

## 5.82 Short name views

*This Subclause is modified by Subclause 18.8, “Short name views”, in ISO/IEC 9075-4.  
This Subclause is modified by Subclause 24.14, “Short name views”, in ISO/IEC 9075-9.  
This Subclause is modified by Subclause 13.7, “Short name views”, in ISO/IEC 9075-13.  
This Subclause is modified by Subclause 20.15, “Short name views”, in ISO/IEC 9075-14.*

### Function

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

### Definition

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

GRANT SELECT ON TABLE CATALOG_NAME
    TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE ADMIN_ROLE_AUTHS
    TO PUBLIC WITH GRANT OPTION;

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



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

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

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

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

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

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

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

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

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

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

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

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

```

        COLUMN_NAME,          ORDINAL_POSITION,    COLUMN_DEFAULT,
        IS_NULLABLE,          DATA_TYPE,          CHAR_MAX_LENGTH,
        CHAR_OCTET_LENGTH,    NUMERIC_PRECISION,    NUMERIC_PREC_RADIX,
        NUMERIC_SCALE,        DATETIME_PRECISION,    INTERVAL_TYPE,
        INTERVAL_PRECISION,    CHAR_SET_CATALOG,      CHAR_SET_SCHEMA,
        CHARACTER_SET_NAME,    COLLATION_CATALOG,     COLLATION_SCHEMA,
        COLLATION_NAME,        DOMAIN_CATALOG,         DOMAIN_SCHEMA,
        DOMAIN_NAME,          UDT_CATALOG,           UDT_SCHEMA,
        UDT_NAME,             SCOPE_CATALOG,          SCOPE_SCHEMA,
        SCOPE_NAME,           MAX_CARDINALITY,        DTD_IDENTIFIER,
        IS_SELF_REF,          IS_IDENTITY,            ID_GENERATION,
        ID_START,             ID_INCREMENT,           ID_MAXIMUM,
        ID_MINIMUM,           ID_CYCLE,               IS_GENERATED,

        GENERATION_EXPR,      IS_SYSPER_START,        IS_SYSPER_END,
        SYSPER_TSTMP_GEN,     IS_UPDATABLE,           DECLARED_DATA_TYPE,

        DECLARED_NUM_PREC,    DECLARED_NUM_SCALE) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
        COLUMN_NAME, ORDINAL_POSITION, COLUMN_DEFAULT,
        IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
        CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
        INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
        COLLATION_NAME, DOMAIN_CATALOG, DOMAIN_SCHEMA,
        DOMAIN_NAME, UDT_CATALOG, UDT_SCHEMA,
        UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
        SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
        IS_SELF_REFERENCING, IS_IDENTITY, IDENTITY_GENERATION,
        IDENTITY_START, IDENTITY_INCREMENT, IDENTITY_MAXIMUM,
        IDENTITY_MINIMUM, IDENTITY_CYCLE, IS_GENERATED,

        GENERATION_EXPRESSION, IS_SYSTEM_TIME_PERIOD_START, IS_SYSTEM_TIME_PERIOD_END,
        SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION, IS_UPDATABLE, DECLARED_DATA_TYPE,

        DECLARED_NUMERIC_PRECISION, DECLARED_NUMERIC_SCALE
FROM INFORMATION_SCHEMA.COLUMNS;

GRANT SELECT ON TABLE COLUMNS_S
TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE CONSTR_ROUT_USE_S
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW CONSTR_COL_USAGE
( TABLE_CATALOG,          TABLE_SCHEMA,          TABLE_NAME,
  COLUMN_NAME,             CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,

```

```
        CONSTRAINT_NAME ) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       COLUMN_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
       CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE;
```

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

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

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

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

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

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

```
GRANT SELECT ON TABLE DOMAINS_S
```

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE ELEMENT\_TYPES\_S  
TO PUBLIC WITH GRANT OPTION;

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

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

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

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

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

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

```
        RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,  
        RESULT_CAST_CHAR_MAX_LENGTH, RESULT_CAST_CHAR_OCTET_LENGTH,  
        RESULT_CAST_CHAR_SET_CATALOG, RESULT_CAST_CHAR_SET_SCHEMA,  
        RESULT_CAST_CHAR_SET_NAME,  
        RESULT_CAST_COLLATION_CATALOG, RESULT_CAST_COLLATION_SCHEMA,  
        RESULT_CAST_COLLATION_NAME, RESULT_CAST_NUMERIC_PRECISION,  
        RESULT_CAST_NUMERIC_RADIX, RESULT_CAST_NUMERIC_SCALE,  
        RESULT_CAST_DATETIME_PRECISION, RESULT_CAST_INTERVAL_TYPE,  
        RESULT_CAST_INTERVAL_PRECISION, RESULT_CAST_TYPE_UDT_CATALOG,  
        RESULT_CAST_TYPE_UDT_SCHEMA, RESULT_CAST_TYPE_UDT_NAME,  
        RESULT_CAST_SCOPE_CATALOG, RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,  
        RESULT_CAST_MAX_CARDINALITY, RESULT_CAST_DTD_IDENTIFIER,  
        DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,  
        DECLARED_NUMERIC_SCALE  
FROM INFORMATION_SCHEMA.METHOD_SPECIFICATIONS;
```

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

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

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

```
CREATE VIEW PARAMETERS_S  
( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,  
  ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
```

```

AS_LOCATOR,          PARAMETER_NAME,          FROM_SQL_SPEC_CAT,
FROM_SQL_SPEC_SCH,   FROM_SQL_SPEC_NAME, TO_SQL_SPEC_CAT,
TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME,   DATA_TYPE,
CHAR_MAX_LENGTH,     CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
CHAR_SET_SCHEMA,     CHARACTER_SET_NAME, COLLATION_CATALOG,
COLLATION_SCHEMA,    COLLATION_NAME,      NUMERIC_PRECISION,
NUMERIC_PREC_RADIX,  NUMERIC_SCALE,      DATETIME_PRECISION,
INTERVAL_TYPE,       INTERVAL_PRECISION, UDT_CATALOG,
UDT_SCHEMA,          UDT_NAME,            SCOPE_CATALOG,
SCOPE_SCHEMA,        SCOPE_NAME,          MAX_CARDINALITY,
DTD_IDENTIFIER,      DECLARED_DATA_TYPE, DECLARED_NUM_PREC,
DECLARED_NUM_SCALE,  PARAMETER_DEFAULT) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
ORDINAL_POSITION, PARAMETER_MODE, IS_RESULT,
AS_LOCATOR, PARAMETER_NAME, FROM_SQL_SPECIFIC_CATALOG,
FROM_SQL_SPECIFIC_SCHEMA, FROM_SQL_SPECIFIC_NAME, TO_SQL_SPECIFIC_CATALOG,
TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, DATA_TYPE,
CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
INTERVAL_TYPE, INTERVAL_PRECISION, UDT_CATALOG,
UDT_SCHEMA, UDT_NAME, SCOPE_CATALOG,
SCOPE_SCHEMA, SCOPE_NAME, MAXIMUM_CARDINALITY,
DTD_IDENTIFIER, DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,
DECLARED_NUMERIC_SCALE, PARAMETER_DEFAULT
FROM INFORMATION_SCHEMA.PARAMETERS;

```

```

GRANT SELECT ON TABLE PARAMETERS_S
TO PUBLIC WITH GRANT OPTION;

```

```

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

```

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

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

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

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

```
CREATE VIEW ROLE_ROUT_GRANTS
  ( GRANTOR,          GRANTEE,          SPECIFIC_CATALOG,
    SPECIFIC_SCHEMA,  SPECIFIC_NAME,    ROUTINE_CATALOG,
    ROUTINE_SCHEMA,   ROUTINE_NAME,     PRIVILEGE_TYPE,
    IS_GRANTABLE ) AS
SELECT GRANTOR, GRANTEE, SPECIFIC_CATALOG,
    SPECIFIC_SCHEMA, SPECIFIC_NAME, ROUTINE_CATALOG,
    ROUTINE_SCHEMA, ROUTINE_NAME, PRIVILEGE_TYPE,
    IS_GRANTABLE
FROM INFORMATION_SCHEMA.ROLE_ROUTINE_GRANTS;
```

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

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

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

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

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



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

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

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

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

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

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

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

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

```
CREATE VIEW ROUTINES_S
```

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

```
( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
ROUTINE_TYPE, MODULE_CATALOG, MODULE_SCHEMA,
MODULE_NAME, UDT_CATALOG, UDT_SCHEMA,
UDT_NAME, DATA_TYPE, CHAR_MAX_LENGTH,
CHAR_OCTET_LENGTH, CHAR_SET_CATALOG, CHAR_SET_SCHEMA,
CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
INTERVAL_PRECISION, TYPE_UDT_CATALOG, TYPE_UDT_SCHEMA,
TYPE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
SCOPE_NAME, MAX_CARDINALITY, DTD_IDENTIFIER,
ROUTINE_BODY, ROUTINE_DEFINITION, EXTERNAL_NAME,
EXTERNAL_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
SCH_LEVEL_ROUTINE, MAX_DYN_RESULT_SETS, IS_USER_DEFINED_CAST,
IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE, TO_SQL_SPEC_CAT,
TO_SQL_SPEC_SCHEMA, TO_SQL_SPEC_NAME, AS_LOCATOR,
CREATED, LAST_ALTERED, NEW_SAVEPOINT_LVL,
IS_UDT_DEPENDENT, RC_FROM_DATA_TYPE, RC_AS_LOCATOR,
RC_CHAR_MAX_LENGTH, RC_CHAR_OCT_LENGTH, RC_CHAR_SET_CAT,
RC_CHAR_SET_SCHEMA, RC_CHAR_SET_NAME, RC_COLLATION_CAT,
RC_COLLATION_SCH, RC_COLLATION_NAME, RC_NUM_PREC,
RC_NUMERIC_RADIX, RC_NUMERIC_SCALE, RC_DATETIME_PREC,
RC_INTERVAL_TYPE, RC_INTERVAL_PREC, RC_TYPE_UDT_CAT,
RC_TYPE_UDT_SCHEMA, RC_TYPE_UDT_NAME, RC_SCOPE_CATALOG,
RC_SCOPE_SCHEMA, RC_SCOPE_NAME, RC_MAX_CARDINALITY,
RC_DTD_IDENTIFIER,
DECLARED_DATA_TYPE, DECLARED_NUM_PREC, DECLARED_NUM_SCALE) AS
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
ROUTINE_TYPE, MODULE_CATALOG, MODULE_SCHEMA,
MODULE_NAME, UDT_CATALOG, UDT_SCHEMA,
UDT_NAME, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
INTERVAL_PRECISION, TYPE_UDT_CATALOG, TYPE_UDT_SCHEMA,
TYPE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER,
ROUTINE_BODY, ROUTINE_DEFINITION, EXTERNAL_NAME,
EXTERNAL_LANGUAGE, PARAMETER_STYLE, IS_DETERMINISTIC,
SQL_DATA_ACCESS, IS_NULL_CALL, SQL_PATH,
SCHEMA_LEVEL_ROUTINE, MAX_DYNAMIC_RESULT_SETS, IS_USER_DEFINED_CAST,
IS_IMPLICITLY_INVOCABLE, SECURITY_TYPE, TO_SQL_SPECIFIC_CATALOG,
TO_SQL_SPECIFIC_SCHEMA, TO_SQL_SPECIFIC_NAME, AS_LOCATOR,
CREATED, LAST_ALTERED, NEW_SAVEPOINT_LEVEL,
IS_UDT_DEPENDENT, RESULT_CAST_FROM_DATA_TYPE, RESULT_CAST_AS_LOCATOR,
RESULT_CAST_CHAR_MAX_LENGTH, RESULT_CAST_CHAR_OCTET_LENGTH,
RESULT_CAST_CHAR_SET_CATALOG,
RESULT_CAST_CHAR_SET_SCHEMA, RESULT_CAST_CHARACTER_SET_NAME,
RESULT_CAST_COLLATION_CATALOG,
RESULT_CAST_COLLATION_SCHEMA, RESULT_CAST_COLLATION_NAME,
RESULT_CAST_NUMERIC_PRECISION,
RESULT_CAST_NUMERIC_RADIX, RESULT_CAST_NUMERIC_SCALE,
RESULT_CAST_DATETIME_PRECISION,
```

```
RESULT_CAST_INTERVAL_TYPE, RESULT_CAST_INTERVAL_PRECISION,  
    RESULT_CAST_TYPE_UDT_CATALOG,  
RESULT_CAST_TYPE_UDT_SCHEMA, RESULT_CAST_TYPE_UDT_NAME,  
    RESULT_CAST_SCOPE_CATALOG,  
RESULT_CAST_SCOPE_SCHEMA, RESULT_CAST_SCOPE_NAME,  
    RESULT_CAST_MAX_CARDINALITY,  
RESULT_CAST_DTD_IDENTIFIER,  
DECLARED_DATA_TYPE, DECLARED_NUMERIC_PRECISION,  
DECLARED_NUMERIC_SCALE  
FROM INFORMATION_SCHEMA.ROUTINES;
```

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

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

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

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

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

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

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

## IWD 9075-11:2011(E)

### 5.82 Short name views

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

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

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

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

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

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

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

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

```
CREATE VIEW TRIG_SEQ_USAGE_S
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,    TRIGGER_NAME,
  SEQUENCE_CATALOG,    SEQUENCE_SCHEMA,    SEQUENCE_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
       SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
FROM INFORMATION_SCHEMA.TRIGGER_SEQUENCE_USAGE;

GRANT SELECT ON TABLE TRIG_SEQ_USAGE_S
TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE TRIG_UPDATE_COLS
TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE TRIG_COLUMN_USAGE
TO PUBLIC WITH GRANT OPTION;

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

GRANT SELECT ON TABLE TRIG_PER_USAGE
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW TRIG_TABLE_USAGE
( TRIGGER_CATALOG,    TRIGGER_SCHEMA,    TRIGGER_NAME,
  TABLE_CATALOG,    TABLE_SCHEMA,    TABLE_NAME ) AS
SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
```

**IWD 9075-11:2011(E)**  
**5.82 Short name views**

```
TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
FROM INFORMATION_SCHEMA.TRIGGER_TABLE_USAGE;
```

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

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

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

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

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

```
CREATE VIEW VIEWS_S
```

```
( TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
  VIEW_DEFINITION,      CHECK_OPTION,        IS_UPDATABLE,
  IS_INSERTABLE_INTRO,  IS_TRIG_UPDATABLE,  IS_TRIG_DELETABLE,
  IS_TRIG_INS_INTRO) AS
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
  VIEW_DEFINITION, CHECK_OPTION, IS_UPDATABLE,
  IS_INSERTABLE_INTRO, IS_TRIGGER_UPDATABLE,
  IS_TRIGGER_DELETABLE, IS_TRIGGER_INSERTABLE_INTRO
FROM INFORMATION_SCHEMA.VIEWS;
```

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

## Conformance Rules

- 1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUT\_GRANTS.
- 2) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS\_S.
- 3) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_DOMAINS\_USAGE.
- 4) Without Feature F341, “Usage tables”, conforming SQL language shall not reference the INFORMATION\_SCHEMA.TRIG\_TABLE\_USAGE view.
- 5) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_UPDATE\_COLS.
- 6) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_DOMAIN\_USAGE.
- 7) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONST\_COL\_USAGE.
- 8) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONST\_TABLE\_USAGE.
- 9) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE\_S.
- 10) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COL\_USAGE.
- 11) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_TABLE\_USAGE.
- 12) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_ROUT\_USAGE\_S.
- 13) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTR\_ROUT\_USE\_S.
- 14) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_ROUT\_USAGE\_S.

- 15) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_SEQ\_USAGE\_S.
- 16) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_COLUMN\_USAGE.
- 17) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_SEQ\_USAGE\_S.
- 18) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_COL\_USAGE.
- 19) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPL\_INFO.
- 20) Without Feature F690, “Collation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS\_S.
- 21) Without Feature F695, “Translation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS\_S.
- 22) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATIONS\_S.TRANS\_SRC\_CATALOG, TRANSLATIONS\_S.TRANS\_SRC\_SCHEMA, or TRANSLATIONS\_S.TRANS\_SRC\_NAME.
- 23) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ATTRIBUTES\_S.
- 24) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECS.
- 25) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPEC\_PARAMS.
- 26) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVS.
- 27) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROL\_TAB\_METH\_GRNTS.
- 28) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES\_S.
- 29) Without Feature S091, “Basic array support” or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES\_S.
- 30) Without Feature S401, “Distinct types based on array types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.UDT\_S.MAX\_CARDINALITY.
- 31) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPEC.CREATED.
- 32) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.CREATED.
- 33) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.LAST\_ALTERED.



- 34) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS\_S.CREATED.
- 35) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS\_S.
- 36) Without Feature T111, “Updatable joins, unions, and columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.IS\_UPDATABLE.
- 37) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_COL\_USAGE.
- 38) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.IS\_GENERATED.
- 39) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.GENERATION\_EXPR.
- 40) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_SEQ\_USAGE\_S.
- 41) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES\_S.
- 42) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQ\_USAGE\_S.
- 43) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference any of the columns COLUMNS\_S.IS\_SYSPER\_START, COLUMNS\_S.IS\_SYSPER\_END, and COLUMNS\_S.SYSPER\_TSTMP\_GEN.
- 44) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_UPDATE\_COLS.
- 45) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference the INFORMATION\_SCHEMA.TRIG\_TABLE\_USAGE view.
- 46) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_ROUT\_USAGE\_S.
- 47) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_SEQ\_USAGE\_S.
- 48) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS\_S.
- 49) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_COLUMN\_USAGE.
- 50) Without Feature T213, “INSTEAD OF triggers”, conforming SQL language shall not reference any of VIEWS\_S.IS\_TRIG\_UPDATABLE, VIEWS\_S.IS\_TRIG\_DELETABLE, VIEWS\_S.IS\_TRIG\_INS\_INT.
- 51) Without Feature T272, “Enhanced savepoint management”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.NEW\_SAVEPOINT\_LEVEL.
- 52) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ADMIN\_ROLE\_AUTHS.

- 53) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUT\_GRANTS.
- 54) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TAB\_METH\_GRNTS.
- 55) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ATTRIBUTES\_S.DECLARED\_DATA\_TYPE, ATTRIBUTES\_S.DEC\_NUMERIC\_PREC and ATTRIBUTES\_S.DEC\_NUM\_SCALE.
- 56) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of COLUMNS\_S.DECLARED\_DATA\_TYPE, COLUMNS\_S.DEC\_NUMERIC\_PREC and COLUMNS\_S.DEC\_NUM\_SCALE.
- 57) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of DOMAINS\_S.DECLARED\_DATA\_TYPE, DOMAINS\_S.DEC\_NUMERIC\_PREC and DOMAINS\_S.DEC\_NUM\_SCALE.
- 58) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ELEMENT\_TYPES\_S.DECLARED\_DATA\_TYPE, ELEMENT\_TYPES\_S.DEC\_NUMERIC\_PREC and ELEMENT\_TYPES\_S.DEC\_NUM\_SCALE.
- 59) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of FIELDS\_S.DECLARED\_DATA\_TYPE, FIELDS\_S.DEC\_NUMERIC\_PREC and FIELDS\_S.DEC\_NUM\_SCALE.
- 60) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of METHOD\_SPECS.DECLARED\_DATA\_TYPE, METHOD\_SPECS.DEC\_NUMERIC\_PREC and METHOD\_SPECS.DEC\_NUM\_SCALE.
- 61) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of METHOD\_SPEC\_PARAMS.DECLARED\_DATA\_TYPE, METHOD\_SPEC\_PARAMS.DEC\_NUMERIC\_PREC and METHOD\_SPEC\_PARAMS.DEC\_NUM\_SCALE.
- 62) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of PARAMETERS\_S.DECLARED\_DATA\_TYPE, PARAMETERS\_S.DEC\_NUMERIC\_PREC and PARAMETERS\_S.DEC\_NUM\_SCALE.
- 63) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of REFERENCED\_TYPES\_S.DECLARED\_DATA\_TYPE, REFERENCED\_TYPES\_S.DEC\_NUMERIC\_PREC and REFERENCED\_TYPES\_S.DEC\_NUM\_SCALE.
- 64) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ROUTINES\_S.DECLARED\_DATA\_TYPE, ROUTINES\_S.DEC\_NUMERIC\_PREC and ROUTINES\_S.DEC\_NUM\_SCALE.
- 65) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of SEQUENCES\_S.DECLARED\_DATA\_TYPE, SEQUENCES\_S.DEC\_NUMERIC\_PREC and SEQUENCES\_S.DEC\_NUM\_SCALE.
- 66) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of UDT\_S.DECLARED\_DATA\_TYPE, UDT\_S.DEC\_NUMERIC\_PREC and UDT\_S.DEC\_NUM\_SCALE.
- 67) Without Feature T522, “Default values for IN parameters of SQL-invoked procedures”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS\_S.PARAMETER\_DEFAULT.

## 6 Definition Schema

*This Clause is modified by [Clause 9](#), “Definition Schema”, in ISO/IEC 9075-3.  
This Clause is modified by [Clause 19](#), “Definition Schema”, in ISO/IEC 9075-4.  
This Clause is modified by [Clause 25](#), “Definition Schema”, in ISO/IEC 9075-9.  
This Clause is modified by [Clause 14](#), “Definition Schema”, in ISO/IEC 9075-13.  
This Clause is modified by [Clause 21](#), “Definition Schema”, in ISO/IEC 9075-14.*

### 6.1 DEFINITION\_SCHEMA Schema

#### Function

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

#### Definition

```
CREATE SCHEMA DEFINITION_SCHEMA  
  AUTHORIZATION DEFINITION_SCHEMA
```

#### Description

*None.*

## 6.2 EQUAL\_KEY\_DEGREES assertion

### Function

The assertion EQUAL\_KEY\_DEGREES ensures that every foreign key is of the same degree as the corresponding unique constraint.

### Definition

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

**6.3 KEY\_DEGREE\_GREATER\_THAN\_OR\_EQUAL\_TO\_1 assertion****6.3 KEY\_DEGREE\_GREATER\_THAN\_OR\_EQUAL\_TO\_1 assertion****Function**

The assertion KEY\_DEGREE\_GREATER\_THAN\_OR\_EQUAL\_TO\_1 ensures that every unique or primary key constraint has at least one unique column and that every referential constraint has at least one referencing column.

**Definition**

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

## 6.4 UNIQUE\_CONSTRAINT\_NAME assertion

### Function

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

NOTE 7 — The UNIQUE\_CONSTRAINT\_NAME assertion avoids the need for separate checks on DOMAINS, TABLE\_CONSTRAINTS, and ASSERTIONS.

### Definition

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

## 6.5 ASSERTIONS base table

### Function

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

### Definition

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

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

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

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

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

### Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion being described.
- 2) The values of IS\_DEFERRABLE have the following meanings:

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

3) The values of INITIALLY\_DEFERRED have the following meanings:

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



## 6.6 ATTRIBUTES base table

### Function

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

### Definition

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

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

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

    CONSTRAINT ATTRIBUTES_CHECK_DATA_TYPE
        CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
                  'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
                ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                      OBJECT_TYPE, DTD_IDENTIFIER
                    FROM DATA_TYPE_DESCRIPTOR ) ),

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

)

## Description

- 1) The values of UDT\_CATALOG, UDT\_SCHEMA, and UDT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type containing the attribute being described.
- 2) The value of ATTRIBUTE\_NAME is the name of the attribute being described.
- 3) The values of UDT\_CATALOG, UDT\_SCHEMA, UDT\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the attribute.
- 4) The value of ORDINAL\_POSITION is the ordinal position of the attribute in the user-defined type.
- 5) The value of ATTRIBUTE\_DEFAULT is null if the attribute being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of ATTRIBUTE\_DEFAULT is "TRUNCATED". Otherwise, the value of ATTRIBUTE\_DEFAULT is a character representation of the default value for the column that obeys the rules specified for <default option> in [Subclause 11.5](#), "<default clause>".

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

- 6) The values of IS\_DERIVED\_REFERENCE\_ATTRIBUTE have the following meanings:

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

## 6.7 AUTHORIZATIONS base table

### Function

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

### Definition

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

  CONSTRAINT AUTHORIZATIONS_PRIMARY_KEY
    PRIMARY KEY (AUTHORIZATION_NAME)
)
```

### Description

1) The values of AUTHORIZATION\_TYPE have the following meanings:

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

## 6.8 CATALOG\_NAME base table

### Function

The CATALOG\_NAME table identifies the catalog or catalogs that are described by this Definition Schema.

### Definition

```
CREATE TABLE CATALOG_NAME (  
    CATALOG_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  
    CONSTRAINT CATALOG_NAME_PRIMARY_KEY  
        PRIMARY KEY ( CATALOG_NAME )  
)
```

### Description

- 1) The values of CATALOG\_NAME are the names of the catalogs that are described by this Definition Schema.

## 6.9 CHARACTER\_ENCODING\_FORMS base table

### Function

The CHARACTER\_ENCODING\_FORMS table has one row for each character encoding form descriptor.

### Definition

```
CREATE TABLE CHARACTER_ENCODING_FORMS (  
    CHARACTER_REPERTOIRE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    CHARACTER_ENCODING_FORM_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  
    CONSTRAINT CHARACTER_ENCODING_FORMS_PRIMARY_KEY  
        PRIMARY KEY ( CHARACTER_ENCODING_FORM_NAME, CHARACTER_REPERTOIRE_NAME ),  
  
    CONSTRAINT CHARACTER_ENCODING_FORMS_FOREIGN_KEY_CHARACTER_REPERTOIRES  
        FOREIGN KEY ( CHARACTER_REPERTOIRE_NAME )  
            REFERENCES CHARACTER_REPERTOIRES  
)
```

### Description

- 1) The value of CHARACTER\_ENCODING\_FORM\_NAME is the name of the character encoding form being described.
- 2) The value of CHARACTER\_REPERTOIRE\_NAME is the name of the character repertoire to which this character encoding form applies.
- 3) There is one row in this table for every character encoding form supported by the SQL-implementation.

## 6.10 CHARACTER\_REPERTOIRES base table

### Function

The CHARACTER\_REPERTOIRES table has one row for each character repertoire descriptor.

### Definition

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

  CONSTRAINT CHARACTER_REPERTOIRES_PRIMARY_KEY
    PRIMARY KEY ( CHARACTER_REPERTOIRE_NAME ),

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

### Description

- 1) The value of DEFAULT\_COLLATION\_CATALOG, DEFAULT\_COLLATION\_SCHEMA, and DEFAULT\_COLLATION\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the default collation of the character repertoire being described.
- 2) The value of CHARACTER\_REPERTOIRE\_NAME is the name of the character repertoire being described.
- 3) There is one row in this table for every character repertoire supported by the SQL-implementation.

## 6.11 CHARACTER\_SETS base table

### Function

The CHARACTER\_SETS table has one row for each character set descriptor.

### Definition

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

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

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

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

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

)

## Description

- 1) The values of CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, and CHARACTER\_SET\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set being described.
- 2) The value of CHARACTER\_REPERTOIRE is the name of the character repertoire of the character set being described.
- 3) The value of FORM\_OF\_USE is the name of the character encoding form used by the character set being described.
- 4) The values of DEFAULT\_COLLATE\_CATALOG, DEFAULT\_COLLATE\_SCHEMA, and DEFAULT\_COLLATE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the explicit or implicit default collation for the character set.
- 5) There is a row in this table for the character set INFORMATION\_SCHEMA.SQL\_TEXT. In that row:
  - a) CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, and CHARACTER\_SET\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_TEXT', respectively.
  - b) DEFAULT\_COLLATE\_CATALOG, DEFAULT\_COLLATE\_SCHEMA, and DEFAULT\_COLLATE\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_TEXT', respectively.
- 6) There is a row in this table for the character set INFORMATION\_SCHEMA.SQL\_IDENTIFIER. In that row:
  - a) CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, and CHARACTER\_SET\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_IDENTIFIER', respectively.
  - b) DEFAULT\_COLLATE\_CATALOG, DEFAULT\_COLLATE\_SCHEMA, and DEFAULT\_COLLATE\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_IDENTIFIER', respectively.
- 7) There is a row in this table for the character set INFORMATION\_SCHEMA.SQL\_CHARACTER. In that row:
  - a) CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, and CHARACTER\_SET\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_IDENTIFIER', respectively.
  - b) DEFAULT\_COLLATE\_CATALOG, DEFAULT\_COLLATE\_SCHEMA, and DEFAULT\_COLLATE\_NAME are the name of the catalog, 'INFORMATION\_SCHEMA', and 'SQL\_CHARACTER', respectively.



## 6.12 CHECK\_COLUMN\_USAGE base table

### Function

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

### Definition

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

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

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

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

### Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column identified by a <column reference> explicitly or implicitly contained in the <search condition> of the constraint being described.

## 6.13 CHECK\_CONSTRAINT\_ROUTINE\_USAGE base table

### Function

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

### Definition

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

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

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

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

### Description

- 1) The CHECK\_CONSTRAINT\_ROUTINE\_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in an <assertion definition> or in the <check constraint definition> contained in either a <domain constraint> or a <table constraint definition>.
- 2) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the assertion or check constraint being described.
- 3) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

## 6.14 CHECK\_CONSTRAINTS base table

### Function

The CHECK\_CONSTRAINTS table has one row for each domain constraint, table check constraint, and assertion.

### Definition

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

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

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

### Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) Case:
  - a) If the character representation of the <search condition> contained in the <check constraint definition>, <domain constraint>, or <assertion definition> that defined the check constraint being described can be represented without truncation, then the value of CHECK\_CLAUSE is that character representation.
  - b) Otherwise, the value of CHECK\_CLAUSE is the null value.

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

## 6.15 CHECK\_PERIOD\_USAGE base table

### Function

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

### Definition

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

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

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

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

### Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and PERIOD\_NAME are the catalog name, unqualified schema name, qualified identifier, and period name, respectively, of a period identified by a <application time period name> explicitly or implicitly contained in the <search condition> of the constraint being described.

## 6.16 CHECK\_TABLE\_USAGE base table

### Function

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

### Definition

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

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

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

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

### Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of the constraint being described.

## 6.17 COLLATIONS base table

### Function

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

### Definition

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

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

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

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

### Description

- 1) The values of COLLATION\_CATALOG, COLLATION\_SCHEMA, and COLLATION\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation being described.
- 2) The values of PAD\_ATTRIBUTE have the following meanings:

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

- 3) The value of CHARACTER\_REPERTOIRE\_NAME is the name of the character repertoire to which the collation being described is applicable.

## 6.18 COLLATION\_CHARACTER\_SET\_APPLICABILITY base table

### Function

The COLLATION\_CHARACTER\_SET\_APPLICABILITY table has one row for each applicability of a collation to a character set.

### Definition

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

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

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

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

### Description

- 1) The values of COLLATION\_CATALOG, COLLATION\_SCHEMA, and COLLATION\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the collation whose applicability is being described.
- 2) The values of CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, and CHARACTER\_SET\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set to which the collation is applicable.

## 6.19 COLUMN\_COLUMN\_USAGE base table

### Function

The COLUMN\_COLUMN\_USAGE table has one row for each case where a generated column depends on a base column.

### Definition

```
CREATE TABLE COLUMN_COLUMN_USAGE (
    TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DEPENDENT_COLUMN      INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT COLUMN_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      COLUMN_NAME, DEPENDENT_COLUMN ),

    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_1
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      COLUMN_NAME ) REFERENCES COLUMNS,

    CONSTRAINT COLUMN_COLUMN_USAGE_FOREIGN_KEY_COLUMNS_2
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      DEPENDENT_COLUMN ) REFERENCES COLUMNS
)
```

### Description

- 1) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME and DEPENDENT\_COLUMN are the catalog name, unqualified schema name and qualified identifier, respectively, of a generated column *GC*.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME and COLUMN\_NAME are the catalog name, unqualified schema name and qualified identifier, respectively, of a column on which *GC* depends.



## 6.20 COLUMN\_PRIVILEGES base table

### Function

The COLUMN\_PRIVILEGES table has one row for each column privilege descriptor. It effectively contains a representation of the column privilege descriptors.

### Definition

```
CREATE TABLE COLUMN_PRIVILEGES (
  GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG   INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PRIVILEGE_TYPE    INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT COLUMN_PRIVILEGE_TYPE_CHECK
    CHECK ( PRIVILEGE_TYPE IN
      ( 'SELECT', 'INSERT', 'UPDATE', 'REFERENCES' ) ),
  IS_GRANTABLE     INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT COLUMN_PRIVILEGE_IS_GRANTABLE_NOT_NULL
    NOT NULL,

  CONSTRAINT COLUMN_PRIVILEGE_PRIMARY_KEY
    PRIMARY KEY ( GRANTOR, GRANTEE,
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      PRIVILEGE_TYPE, COLUMN_NAME ),

  CONSTRAINT COLUMN_PRIVILEGE_FOREIGN_KEY_COLUMNS
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
      REFERENCES COLUMNS,

  CONSTRAINT COLUMN_PRIVILEGES_GRANTOR_CHECK
    CHECK ( GRANTOR = '_SYSTEM'
      OR
      GRANTOR IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS ) ),

  CONSTRAINT COLUMN_PRIVILEGES_GRANTEE_CHECK
    CHECK ( GRANTEE = 'PUBLIC'
      OR
      GRANTEE IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS ) )
```

)

## Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted column privileges, on the column identified by TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME, to the user or role identified by the value of GRANTEE for the column privilege being described, or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the column privilege being described is granted.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the column on which the privilege being described was granted.
- 4) The values of PRIVILEGE\_TYPE have the following meanings:

SELECT	The user has SELECT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
INSERT	The user has INSERT privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
UPDATE	The user has UPDATE privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.
REFERENCE	The user has REFERENCES privilege on the column identified by TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME.

- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

## 6.21 COLUMNS base table

### Function

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

### Definition

```
CREATE TABLE COLUMNS (
    TABLE_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION              INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT COLUMNS_ORDINAL_POSITION_NOT_NULL
        NOT NULL,
    CONSTRAINT COLUMNS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 ),
    CONSTRAINT COLUMNS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                           FROM COLUMNS
                           GROUP BY
                               TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) ),
    DTD_IDENTIFIER                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_DEFAULT                INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_NULLABLE                   INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT COLUMNS_IS_NULLABLE_NOT_NULL
        NOT NULL,
    IS_SELF_REFERENCING           INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT COLUMNS_IS_SELF_REFERENCING_NOT_NULL
        NOT NULL,
    IS_IDENTITY                   INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT COLUMNS_IS_IDENTITY_NOT_NULL
        NOT NULL,
    IDENTITY_GENERATION           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IDENTITY_GENERATION_CHECK
        CHECK ( IDENTITY_GENERATION IN
                ( 'ALWAYS', 'BY DEFAULT' ) ),
    IDENTITY_START                INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_INCREMENT            INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MAXIMUM              INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_MINIMUM              INFORMATION_SCHEMA.CHARACTER_DATA,
    IDENTITY_CYCLE                INFORMATION_SCHEMA.YES_OR_NO,
    IS_GENERATED                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COLUMNS_IS_GENERATED_NOT_NULL
        NOT NULL,
    CONSTRAINT COLUMNS_IS_GENERATED_CHECK
        CHECK ( IS_GENERATED IN
                ( 'NEVER', 'ALWAYS' ) ),
```

## 6.21 COLUMNS base table

```

GENERATION_EXPRESSION                                INFORMATION_SCHEMA.CHARACTER_DATA,

IS_SYSTEM_TIME_PERIOD_START                          INFORMATION_SCHEMA.YES_OR_NO
CONSTRAINT IS_SYSTEM_TIME_PERIOD_START_NOT_NULL
    NOT NULL,
IS_SYSTEM_TIME_PERIOD_END                            INFORMATION_SCHEMA.YES_OR_NO
CONSTRAINT IS_SYSTEM_TIME_PERIOD_END_NOT_NULL
    NOT NULL,
SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION              INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT COLUMNS_SYSTEM_TIME_PERIOD_TIMESTAMP_GENERATION_CHECK
    CHECK ( SYSTEMTIME_PERIOD_TIMESTAMP_GENERATION IN ( 'ALWAYS' ) ),

IS_UPDATABLE                                          INFORMATION_SCHEMA.YES_OR_NO
CONSTRAINT COLUMNS_IS_UPDATABLE_NOT_NULL
    NOT NULL,

CONSTRAINT COLUMNS_PRIMARY_KEY
    PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

CONSTRAINT COLUMNS_UNIQUE
    UNIQUE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, ORDINAL_POSITION ),

CONSTRAINT COLUMNS_FOREIGN_KEY_TABLES
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TABLES,

CONSTRAINT COLUMNS_CHECK_REFERENCES_DOMAIN
    CHECK ( DOMAIN_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
        OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN
        ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
          FROM DOMAINS ) ),

CONSTRAINT COLUMNS_CHECK_IDENTITY_COMBINATIONS
    CHECK ( ( IS_IDENTITY = 'NO' ) =
        ( ( IDENTITY_GENERATION, IDENTITY_START, IDENTITY_INCREMENT,
            IDENTITY_MAXIMUM, IDENTITY_MINIMUM, IDENTITY_CYCLE ) IS NULL ) ),

CONSTRAINT COLUMNS_CHECK_GENERATION_COMBINATIONS
    CHECK ( ( IS_GENERATED = 'NEVER' ) =
        ( GENERATION_EXPRESSION IS NULL ) ),

CONSTRAINT COLUMNS_CHECK_DATA_TYPE
    CHECK ( DOMAIN_CATALOG NOT IN
        ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
        ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
          IS NOT NULL
        AND
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
          'TABLE', DTD_IDENTIFIER ) NOT IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) )
        OR

```

```

( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
  IS NULL
AND
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    'TABLE', DTD_IDENTIFIER ) IN
  ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
        OBJECT_TYPE, DTD_IDENTIFIER
    FROM DATA_TYPE_DESCRIPTOR ) ) )
)

```

## Description

- 1) Case:
  - a) If a column is described by a column descriptor included in a table descriptor, then the table descriptor and the column descriptor are associated with that column.
  - b) If a column is described by a column descriptor included in a view descriptor, then the view descriptor and the corresponding column descriptor of the table of the <query expression> are associated with that column.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the column being described.
- 3) The value of COLUMN\_NAME is the name of the column being described.
- 4) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the column.
- 5) The values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA, and DOMAIN\_NAME are null if the column being described is not defined using a <domain name>. Otherwise, the values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA, and DOMAIN\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the domain used by the column being described.
- 6) The value of ORDINAL\_POSITION is the ordinal position of the column in the table.
- 7) Let *DC* be the descriptor of the column being described. If *DC* includes a <default option>, then let *DO* be that <default option>. The value of COLUMN\_DEFAULT is
 

Case:

  - a) If *DC* does not include a <default option>, then the null value.
  - b) If CHARACTER\_LENGTH(*DO*) > *ML*, then "TRUNCATED".
  - c) Otherwise, *DO*.
- 8) The values of IS\_NULLABLE have the following meanings:

YES	The column is possibly nullable.
NO	The column is known not nullable.

9) The values of IS\_SELF\_REFERENCING have the following meanings:

YES	The column is a self-referencing column.
NO	The column is not a self-referencing column.

10) The values of IS\_IDENTITY have the following meanings:

YES	The column is an identity column.
NO	The column is not an identity column.

11) The values of IDENTITY\_GENERATION have the following meanings:

ALWAYS	The column is an identity column whose values are always generated.
BY DEFAULT	The column is an identity column whose values are generated by default.
<i>null</i>	The column is not an identity column.

12) The value of IDENTITY\_START is null if the column is not an identity column; otherwise, it is a character representation of the start value of the column being described.

13) The value of IDENTITY\_INCREMENT is null if the column is not an identity column; otherwise, it is a character representation of the increment of the column being described.

14) The value of IDENTITY\_MAXIMUM is null if the column is not an identity column; otherwise, it is a character representation of the maximum value of the column being described.

15) The value of IDENTITY\_MINIMUM is null if the column is not an identity column; otherwise, it is a character representation of the minimum value of the column being described.

16) The value of IDENTITY\_CYCLE is null if the column is not an identity column; otherwise, it is either YES or NO. The values of IDENTITY\_CYCLE have the following meanings:

YES	The cycle option of the column is CYCLE.
NO	The cycle option of the column is NO CYCLE.
<i>null</i>	The column is not an identity column.

17) The values of IS\_GENERATED have the following meanings:

NEVER	The column is not a generated column.
ALWAYS	The column is generated and stored.

18) The value of GENERATION\_EXPRESSION is the text of the <generation expression> specified in the <column definition> when the column identified by COLUMN\_NAME is defined.

19) The values of IS\_UPDATABLE have the following meanings:

YES	The column is updatable.
NO	The column is not updatable.

20) The values of IS\_SYSTEM\_TIME\_PERIOD\_START have the following meanings:

YES	The column is a system-time-period start column.
NO	The column is not a system-time period start column.

21) The values of IS\_SYSTEM\_TIME\_PERIOD\_END have the following meanings:

YES	The column is system-time period end column.
NO	The column is system-time period end column.

22) The values of SYSTEM\_TIME\_PERIOD\_TIMESTAMP\_GENERATION have the following meanings:

ALWAYS	The column is a system-time period start column or a system-time period end column whose values are always generated.
<i>null</i>	The column is not system-time period start column or a system-time period end column.

## 6.22 DATA\_TYPE\_DESCRIPTOR base table

*This Subclause is modified by Subclause 25.2, “DATA\_TYPE\_DESCRIPTOR base table”, in ISO/IEC 9075-9.  
This Subclause is modified by Subclause 21.1, “DATA\_TYPE\_DESCRIPTOR base table”, in ISO/IEC 9075-14.*

### Function

The DATA\_TYPE\_DESCRIPTOR table has one row for each usage of a datatype as identified by ISO/IEC 9075. It effectively contains a representation of the data type descriptors.

### Definition

```
CREATE TABLE DATA_TYPE_DESCRIPTOR (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_OBJECT_TYPE
        CHECK ( OBJECT_TYPE IN
            ( 'TABLE', 'DOMAIN', 'USER-DEFINED TYPE',
              'ROUTINE', 'SEQUENCE' ) ),
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DATA_TYPE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT DATA_TYPE_DESCRIPTOR_OBJECT_DATA_TYPE_NOT_NULL
        NOT NULL,
    CHARACTER_SET_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_SET_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CHARACTER_MAXIMUM_LENGTH INFORMATION_SCHEMA.CARDINAL_NUMBER,
    CHARACTER_OCTET_LENGTH INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COLLATION_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLATION_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    NUMERIC_PRECISION       INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_PRECISION_RADIX INFORMATION_SCHEMA.CARDINAL_NUMBER,
    NUMERIC_SCALE          INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DECLARED_DATA_TYPE     INFORMATION_SCHEMA.CHARACTER_DATA,
    DECLARED_NUMERIC_PRECISION INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DECLARED_NUMERIC_SCALE INFORMATION_SCHEMA.CARDINAL_NUMBER,
    DATETIME_PRECISION     INFORMATION_SCHEMA.CARDINAL_NUMBER,
    INTERVAL_TYPE          INFORMATION_SCHEMA.CHARACTER_DATA,
    INTERVAL_PRECISION     INFORMATION_SCHEMA.CARDINAL_NUMBER,
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SCOPE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MAXIMUM_CARDINALITY    INFORMATION_SCHEMA.CARDINAL_NUMBER,

    CONSTRAINT DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS
        CHECK ( ( DATA_TYPE IN
            ( 'CHARACTER', 'CHARACTER VARYING', 'CHARACTER LARGE OBJECT' ) )
```



**6.22 DATA\_TYPE\_DESCRIPTOR base table**

```

AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME ) IS NOT NULL
AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
      NUMERIC_SCALE ) IS NULL
AND
    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE IN
      ( 'BINARY', 'BINARY VARYING', 'BINARY LARGE OBJECT' )
    AND
      ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
    AND
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, COLLATION_CATALOG,
        COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
      MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE IN
      ( 'SMALLINT', 'INTEGER', 'BIGINT' )
    AND
      ( DECLARED_DATA_TYPE IN
        ( 'SMALLINT', 'INTEGER', 'BIGINT', 'NUMERIC', 'DECIMAL' )
      AND
        ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
      AND
        NUMERIC_PRECISION_RADIX IN
          ( 2, 10 )
      AND
        NUMERIC_PRECISION IS NOT NULL

```

## 6.22 DATA\_TYPE\_DESCRIPTOR base table

```

AND
    NUMERIC_SCALE = 0
AND
    DECLARED_NUMERIC_SCALE IS NULL OR DECLARED_NUMERIC_SCALE = 0
AND
    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE IN
      ( 'NUMERIC', 'DECIMAL' )
    AND
    DECLARED_DATA_TYPE IN
      ( 'SMALLINT', 'INTEGER', 'BIGINT', 'NUMERIC', 'DECIMAL' )
    AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    NUMERIC_PRECISION_RADIX = 10
    AND
    ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
    AND
    DATETIME_PRECISION IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE IN
      ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
    AND
    DECLARED_DATA_TYPE IN
      ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
    AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    NUMERIC_PRECISION IS NOT NULL
    AND
    NUMERIC_PRECISION_RADIX = 2
    AND
    NUMERIC_SCALE IS NULL
    AND

```

## 6.22 DATA\_TYPE\_DESCRIPTOR base table

```

    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE IN
      ( 'DATE', 'TIME', 'TIMESTAMP',
        'TIME WITH TIME ZONE', 'TIMESTAMP WITH TIME ZONE' )
    AND
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NOT NULL
    AND
      ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
      MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'INTERVAL'
    AND
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
        NUMERIC_SCALE ) IS NULL
    AND
      DATETIME_PRECISION IS NOT NULL
    AND
      INTERVAL_TYPE IN
        ( 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'SECOND',
          'YEAR TO MONTH', 'DAY TO HOUR', 'DAY TO MINUTE',
          'DAY TO SECOND', 'HOUR TO MINUTE',
          'HOUR TO SECOND', 'MINUTE TO SECOND' )
    AND
      INTERVAL_PRECISION IS NOT NULL
    AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
    AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND

```

## 6.22 DATA\_TYPE\_DESCRIPTOR base table

```

        MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'BOOLEAN'
    AND
        ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
        ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
          NUMERIC_SCALE ) IS NULL
    AND
        DATETIME_PRECISION IS NULL
    AND
        ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME ) IS NULL
    AND
        ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
        MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'USER-DEFINED'
    AND
        ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
          CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
          COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
        ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
          NUMERIC_SCALE ) IS NULL
    AND
        DATETIME_PRECISION IS NULL
    AND
        ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME ) IS NOT NULL
    AND
        ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
        MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'REF'
    AND
        ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
    AND
        ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME, COLLATION_CATALOG,
          COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
        ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
          NUMERIC_SCALE ) IS NULL
    AND
        DATETIME_PRECISION IS NULL
    AND
        ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL

```

**6.22 DATA\_TYPE\_DESCRIPTOR base table**

```

AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NOT NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'ARRAY'
AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
      COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
      NUMERIC_SCALE ) IS NULL
AND
    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NOT NULL )
OR
    ( DATA_TYPE = 'MULTISET'
AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
      COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
      NUMERIC_SCALE ) IS NULL
AND
    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE = 'ROW'
AND
    ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
      CHARACTER_SET_NAME, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, COLLATION_CATALOG,
      COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
      NUMERIC_SCALE ) IS NULL

```

## 6.22 DATA\_TYPE\_DESCRIPTOR base table

```

AND
    DATETIME_PRECISION IS NULL
AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
AND
    MAXIMUM_CARDINALITY IS NULL )
OR
    ( DATA_TYPE NOT IN
      ( 'CHARACTER', 'CHARACTER VARYING',
        'CHARACTER LARGE OBJECT', 'BINARY',
        'BINARY VARYING', 'BINARY LARGE OBJECT',
        'NUMERIC', 'DECIMAL', 'SMALLINT', 'INTEGER', 'BIGINT',
        'FLOAT', 'REAL', 'DOUBLE PRECISION',
        'DATE', 'TIME', 'TIMESTAMP',
        'INTERVAL', 'BOOLEAN', 'USER-DEFINED',
        'REF', 'ROW', 'ARRAY', 'MULTISET' ) ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_UDT
    CHECK ( USER_DEFINED_TYPE_CATALOG <>
      ANY ( SELECT CATALOG_NAME
        FROM SCHEMATA )
    OR
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IN
        ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME
        FROM USER_DEFINED_TYPES ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_PRIMARY_KEY
    PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
      OBJECT_TYPE, DTD_IDENTIFIER ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_COLLATION_CHARACTER_SET_APPLICABILITY
    CHECK ( CHARACTER_SET_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      COLLATION_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
        COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IN
        ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
          COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
        FROM COLLATION_CHARACTER_SET_APPLICABILITY )
    CONSTRAINT DATA_TYPE_DESCRIPTOR_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA )
      REFERENCES SCHEMATA

```

)

## Description

- 1) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, and OBJECT\_NAME are the fully qualified name of the object (table, domain, SQL-invoked routine, user-defined type, or sequence generator) whose descriptor includes the data type descriptor, and OBJECT\_TYPE is 'TABLE', 'DOMAIN', 'ROUTINE', 'USER-DEFINED TYPE', or 'SEQUENCE', as the case may be.
- 2) The value of DTD\_IDENTIFIER is the implementation-dependent value that uniquely identifies the data type descriptor among all data type descriptors of the schema object identified by OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and OBJECT\_TYPE.
- 3) Let DTN be the data type name specified in the declaration of the data type being described. The value of DATA\_TYPE is

Case:

- a) If the data type being described is a numeric type, then the name of the normal form of DTN.
- b) Otherwise, DTN.

NOTE 10 — The normal form of a numeric data type is defined in [Subclause 6.1](#), “<data type>” of Part 2 of this International Standard.

- 4) The value of DECLARED\_DATA\_TYPE is

Case:

- a) If the SQL-Implementation knows the data type name specified in the declaration of the data type being described, then DTN.
- b) Otherwise, the null value.

NOTE 11 — The declared data type of a numeric item may be unknown if, for example, the SQL-Implementation has been upgraded to support Feature T322 but manages schema items declared before the upgrade was installed.

- 5) Case:

- a) If the data type being described is a character string type, then the values of CHARACTER\_MAXIMUM\_LENGTH and CHARACTER\_OCTET\_LENGTH are, respectively, the length or maximum length in characters and the length or maximum length in octets of the data type being described.
- b) If the data type being described is a binary string type, then the values of CHARACTER\_MAXIMUM\_LENGTH and CHARACTER\_OCTET\_LENGTH are the length or maximum length in octets of the data type being described.
- c) If the data type being described is a reference type, then the values of CHARACTER\_MAXIMUM\_LENGTH and CHARACTER\_OCTET\_LENGTH are the length in octets of the data type being described.
- d) Otherwise, the values of CHARACTER\_MAXIMUM\_LENGTH and CHARACTER\_OCTET\_LENGTH are the null value.

- 6) The values of CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, CHARACTER\_SET\_NAME, COLLATION\_CATALOG, COLLATION\_SCHEMA and COLLATION\_NAME are,

**6.22 DATA\_TYPE\_DESCRIPTOR base table**

respectively, the qualified name of the character set and applicable collation, if any, if it is a character string type, of the data type being described.

- 7) For a numeric type, the values of NUMERIC\_PRECISION, NUMERIC\_PRECISION\_RADIX, and NUMERIC\_SCALE are, respectively, the implemented precision, the radix of the precision and the scale of the data type being described.
- 8) For a numeric type, the values of DECLARED\_NUMERIC\_PRECISION, and DECLARED\_NUMERIC\_SCALE are, respectively,

Case:

- a) If they are known to the SQL-Implementation, the declared precision and declared scale of the data type being described.
- b) Otherwise, the null value.

NOTE 12 — The declared precision or scale of a numeric item may be unknown if, for example, the SQL-Implementation has been upgraded to support Feature T322 but manages schema items declared before the upgrade was installed, or if it was not specified in the declaration of the item.

- 9) For a datetime or interval type, the value of DATETIME\_PRECISION is the fractional seconds precision of the data type being described.
- 10) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA and USER\_DEFINED\_TYPE\_NAME are, the fully qualified name of the user-defined type or the referenced structured type if it is a reference type, if specified, of the data type being described.
- 11) If DATA\_TYPE is 'INTERVAL', then the values of INTERVAL\_TYPE are the value for <interval qualifier> (as specified in Table 27, “Codes used for <interval qualifier>s in Dynamic SQL”, in [ISO9075-2]) for the data type being described; otherwise, INTERVAL\_TYPE is the null value.
- 12) If DATA\_TYPE is 'INTERVAL', then the values of INTERVAL\_PRECISION are the interval leading field precision of the data type being described; otherwise, INTERVAL\_PRECISION is the null value.
- 13) Case:
  - a) If DATA\_TYPE is 'USER-DEFINED', then the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the qualified name of the user-defined type being described.
  - b) If the DATA\_TYPE is 'REF', then the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the qualified name of the referenced structured type of the reference type being described.
  - c) Otherwise, the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the null value.
- 14) If DATA\_TYPE is 'REF', then the values of SCOPE\_CATALOG, SCOPE\_SCHEMA, and SCOPE\_NAME are the fully qualified name of the referenceable table, if any; otherwise, the values of SCOPE\_CATALOG, SCOPE\_SCHEMA, and SCOPE\_NAME are the null value.
- 15) If DATA\_TYPE is the name of some character string type and OBJECT\_SCHEMA is 'INFORMATION\_SCHEMA', then the values for CHARACTER\_MAXIMUM\_LENGTH, CHARACTER\_OCTET\_LENGTH, CHARACTER\_SET\_CATALOG, CHARACTER\_SET\_SCHEMA, CHARACTER\_SET\_NAME, COLLATION\_CATALOG, COLLATION\_SCHEMA, and COLLATION\_NAME are implementation-defined.



**6.22 DATA\_TYPE\_DESCRIPTOR base table**

- 16) If DATA\_TYPE is 'ARRAY', then the value of MAXIMUM\_CARDINALITY is the maximum cardinality of the array type being described. Otherwise, the value of MAXIMUM\_CARDINALITY is the null value.
- 17) If DATA\_TYPE is 'ROW' then the data type being described is a row type.

## 6.23 DIRECT\_SUPERTABLES base table

### Function

The DIRECT\_SUPERTABLES base table contains one row for each direct subtable-supertable relationship.

### Definition

```
CREATE TABLE DIRECT_SUPERTABLES (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT DIRECT_SUPERTABLES_PRIMARY_KEY
        PRIMARY KEY (TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, SUPERTABLE_NAME ),

    CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_TABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT DIRECT_SUPERTABLES_FOREIGN_KEY_SUPERTABLE_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, SUPERTABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_SAME_TABLES
        CHECK ( TABLE_NAME <> SUPERTABLE_NAME ),

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA,
                    SUPERTABLE_NAME, TABLE_NAME ) NOT IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA,
                      TABLE_NAME, SUPERTABLE_NAME
                    FROM DIRECT_SUPERTABLES ) ),

    CONSTRAINT DIRECT_SUPERTABLES_CHECK_NOT_ALSO_INDIRECT
        CHECK (
            NOT EXISTS (
                WITH RECURSIVE SUPER
                ( TYPE, SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME,
                  SUPERTABLE_NAME )
                AS
                ( SELECT 0, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      SUPERTABLE_NAME
                  FROM DIRECT_SUPERTABLES
                UNION
                  SELECT 1, S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME,
                        D.SUPERTABLE_NAME
                  FROM SUPER AS S
                JOIN
                    DIRECT_SUPERTABLES AS D
                ON ( S.SUBTABLE_CATALOG, S.SUBTABLE_SCHEMA, S.SUBTABLE_NAME )
                  = ( D.TABLE_CATALOG, D.TABLE_SCHEMA, D.TABLE_NAME ) )
            )
        )
```

**6.23 DIRECT\_SUPERTABLES base table**

```
SELECT SUBTABLE_CATALOG, SUBTABLE_SCHEMA, SUBTABLE_NAME, SUPERTABLE_NAME
FROM SUPER
WHERE TYPE = 1
INTERSECT
SELECT *
FROM DIRECT_SUPERTABLES ) )
)
```

**Description**

- 1) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the fully qualified name of the subtable.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and SUPERTABLE\_NAME are the fully qualified name of the direct supertable.

## 6.24 DIRECT\_SUPERTYPES base table

### Function

The DIRECT\_SUPERTYPES base table contains one row for each direct subtype-supertype relationship.

### Definition

```
CREATE TABLE DIRECT_SUPERTYPES (
    USER_DEFINED_TYPE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_CATALOG             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_SCHEMA              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SUPERTYPE_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT DIRECT_SUPERTYPES_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME,
                      SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ),

    CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_1
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                      USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT DIRECT_SUPERTYPES_FOREIGN_KEY_USER_DEFINED_TYPES_2
        FOREIGN KEY ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_SAME_TYPES
        CHECK ( ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME ) <>
              ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) ),

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NO_REFLEXIVITY
        CHECK ( ( SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME,
                  USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME ) NOT IN
              ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                    USER_DEFINED_TYPE_NAME,
                    SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA,
                    SUPERTYPE_NAME
                  FROM DIRECT_SUPERTYPES ) ),

    CONSTRAINT DIRECT_SUPERTYPES_CHECK_NOT_ALSO_INDIRECT
        CHECK (
            NOT EXISTS (
                WITH RECURSIVE SUPER
                ( TYPE, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME,
                  SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME ) AS
                ( SELECT 0, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
```

**6.24 DIRECT\_SUPERTYPES base table**

```

        USER_DEFINED_TYPE_NAME,
        SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
    FROM DIRECT_SUPERTYPES
UNION
    SELECT 1, S.USER_DEFINED_TYPE_CATALOG, S.USER_DEFINED_TYPE_SCHEMA,
        S.USER_DEFINED_TYPE_NAME,
        D.SUPERTYPE_CATALOG, D.SUPERTYPE_SCHEMA, D.SUPERTYPE_NAME
    FROM SUPER AS S
    JOIN
        DIRECT_TPERTYPES AS D
    ON ( D.USER_DEFINED_TYPE_CATALOG, D.USER_DEFINED_TYPE_SCHEMA,
        D.USER_DEFINED_TYPE_NAME ) =
        ( S.SUPERTYPE_CATALOG, S.SUPERTYPE_SCHEMA,
        S.SUPERTYPE_NAME ) )
    SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME,
        SUPERTYPE_CATALOG, SUPERTYPE_SCHEMA, SUPERTYPE_NAME
    FROM SUPER
    WHERE TYPE = 1
INTERSECT
    SELECT *
    FROM DIRECT_SUPERTYPES ) )
)

```

**Description**

- 1) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the fully qualified name of the user-defined type that is a direct subtype.
- 2) The values of SUPERTYPE\_CATALOG, SUPERTYPE\_SCHEMA, and SUPERTYPE\_NAME are the fully qualified name of the user-defined type that is the direct supertype.

## 6.25 DOMAIN\_CONSTRAINTS base table

### Function

The DOMAIN\_CONSTRAINTS table has one row for each domain constraint associated with a domain. It effectively contains a representation of the domain constraint descriptors.

### Definition

```
CREATE TABLE DOMAIN_CONSTRAINTS (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  DOMAIN_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DOMAIN_CATALOG_NOT_NULL
    NOT NULL,
  DOMAIN_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DOMAIN_SCHEMA_NOT_NULL
    NOT NULL,
  DOMAIN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DOMAIN_NAME_NOT_NULL
    NOT NULL,
  IS_DEFERRABLE           INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT IS_DEFERRABLE_NOT_NULL
    NOT NULL,
  INITIALLY_DEFERRED      INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT INITIALLY_DEFERRED_NOT_NULL
    NOT NULL,

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

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

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

  CONSTRAINT DOMAIN_CONSTRAINTS_FOREIGN_KEY_DOMAINS
    FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
      REFERENCES DOMAINS,

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

  CONSTRAINT DOMAIN_CONSTRAINTS_CHECK_SCHEMA_IDENTITY
    CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA )
      = ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) )
```

)

## Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the fully qualified name of the domain constraint.
- 2) The values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA and DOMAIN\_NAME are the fully qualified name of the domain in which the domain constraint is defined.
- 3) The values of IS\_DEFERRABLE have the following meanings:

YES	The domain constraint is deferrable.
NO	The domain constraint is not deferrable.

- 4) The values of INITIALLY\_DEFERRED have the following meanings:

YES	The domain constraint is initially deferred.
NO	The domain constraint is initially immediate.

## 6.26 DOMAINS base table

### Function

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

### Definition

```
CREATE TABLE DOMAINS (
    DOMAIN_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DOMAIN_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DTD_IDENTIFIER_NOT_NULL
        NOT NULL,
    DOMAIN_DEFAULT           INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT DOMAINS_PRIMARY_KEY
        PRIMARY KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ),

    CONSTRAINT DOMAINS_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( DOMAIN_CATALOG, DOMAIN_SCHEMA ) REFERENCES SCHEMATA,

    CONSTRAINT DOMAIN_CHECK_DATA_TYPE
        CHECK ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
            'DOMAIN', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) )
)
```

### Description

- 1) The values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA, and DOMAIN\_NAME are the fully qualified name of the domain.
- 2) The values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA, DOMAIN\_NAME, and DTD\_IDENTIFIER are the values of DOMAIN\_CATALOG, DOMAIN\_SCHEMA, DOMAIN\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the domain.
- 3) The value of DOMAIN\_DEFAULT is null if the domain being described has no explicit default value. If the character representation of the default value cannot be represented without truncation, then the value of DOMAIN\_DEFAULT is “TRUNCATED”. Otherwise, the value of DOMAIN\_DEFAULT is a character representation of the default value for the domain that obeys the rules specified for <default option> in [Subclause 11.5, “<default clause>”](#).

NOTE 13 — “TRUNCATED” is different from other values like CURRENT\_USER or CURRENT\_TIMESTAMP in that it is not an SQL <key word> and does not correspond to a defined value in SQL.



## 6.27 ELEMENT\_TYPES base table

### Function

The ELEMENT\_TYPES table has one row for each collection type. It effectively contains a representation of the element descriptor of the collection type.

### Definition

```
CREATE TABLE ELEMENT_TYPES (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLLECTION_TYPE_IDENTIFIER INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ELEMENT_TYPES_PRIMARY_KEY
        PRIMARY KEY (OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ),

    CONSTRAINT ELEMENT_TYPES_CHECK_COLLECTION_TYPE
        CHECK (
            ( OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, COLLECTION_TYPE_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                  OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR
              WHERE DATA_TYPE IN ( 'ARRAY', 'MULTISET' ) ) ),

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT ELEMENT_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
)
```

### Description

- 1) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and COLLECTION\_TYPE\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the collection type whose element type is being described.
- 2) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME,

**6.27 ELEMENT\_TYPES base table**

OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the element type of the collection type.

- 3) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and ROOT\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the root data type of the element type.

## 6.28 FIELDS base table

### Function

The FIELDS table has one row for each field of each row type. It effectively contains a representation of the field descriptors.

### Definition

```
CREATE TABLE FIELDS (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROW_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FIELD_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION        INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT FIELDS_ORDINAL_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT FIELDS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT FIELDS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                          FROM FIELDS
                          GROUP BY OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                                   OBJECT_TYPE, ROW_IDENTIFIER ) ),
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT FIELDS_PRIMARY_KEY
        PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                      OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME ),

    CONSTRAINT FIELDS_UNIQUE
        UNIQUE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                 OBJECT_TYPE, ROW_IDENTIFIER, ORDINAL_POSITION ),

    CONSTRAINT FIELDS_CHECK_ROW_TYPE
        CHECK (
            ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
              OBJECT_TYPE, ROW_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR
              WHERE DATA_TYPE = 'ROW' ) ),

    CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                      OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT FIELDS_REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
```

**IWD 9075-11:2011(E)**  
**6.28 FIELDS base table**

```
OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )  
REFERENCES DATA_TYPE_DESCRIPTOR  
)
```

## **Description**

- 1) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and ROW\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the row type containing the field being described.
- 2) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and ROOT\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the root data type of the field type.
- 3) The value of FIELD\_NAME is the name of the field being described.
- 4) The value of ORDINAL\_POSITION is the ordinal position of the field in the row type.
- 5) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the field being described.

## 6.29 KEY\_COLUMN\_USAGE base table

### Function

The KEY\_COLUMN\_USAGE table has one or more rows for each row in the TABLE\_CONSTRAINTS table that has a CONSTRAINT\_TYPE of “UNIQUE”, “PRIMARY KEY”, or “FOREIGN KEY”. The rows list the columns that constitute each unique constraint, and the referencing columns in each foreign key constraint.

### Definition

```
CREATE TABLE KEY_COLUMN_USAGE (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_COLUMN_TABLE_CATALOG_NOT_NULL
    NOT NULL,
  TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_COLUMN_TABLE_SCHEMA_NOT_NULL
    NOT NULL,
  TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_COLUMN_TABLE_NAME_NOT_NULL
    NOT NULL,
  COLUMN_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  ORDINAL_POSITION        INFORMATION_SCHEMA.CARDINAL_NUMBER
  CONSTRAINT KEY_COLUMN_ORDINAL_POSITION_NOT_NULL
    NOT NULL
  CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
    CHECK ( ORDINAL_POSITION > 0 )
  CONSTRAINT KEY_COLUMN_USAGE_ORDINAL_POSITION_CONTIGUOUS_CHECK
    CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                      FROM KEY_COLUMN_USAGE
                      GROUP BY CONSTRAINT_CATALOG,
                             CONSTRAINT_SCHEMA,
                             CONSTRAINT_NAME ) ),
  POSITION_IN_UNIQUE_CONSTRAINT INFORMATION_SCHEMA.CARDINAL_NUMBER
  CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_GREATER_THAN_ZERO_CHECK
    CHECK ( POSITION_IN_UNIQUE_CONSTRAINT > 0 ),

  CONSTRAINT KEY_COLUMN_USAGE_UNIQUE_POSITION_IN_UNIQUE_CONSTRAINT
    UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
             CONSTRAINT_NAME, POSITION_IN_UNIQUE_CONSTRAINT ),

  CONSTRAINT KEY_COLUMN_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                 COLUMN_NAME ),

  CONSTRAINT KEY_COLUMN_USAGE_UNIQUE
    UNIQUE ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA,
            CONSTRAINT_NAME, ORDINAL_POSITION ),

  CONSTRAINT KEY_COLUMN_USAGE_FOREIGN_KEY_COLUMNS
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME )
```

**6.29 KEY\_COLUMN\_USAGE base table**

```

REFERENCES COLUMNS,

CONSTRAINT KEY_COLUMN_CONSTRAINT_TYPE_CHECK
CHECK (
    ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
    ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
      FROM TABLE_CONSTRAINTS
      WHERE CONSTRAINT_TYPE IN
        ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) ),

CONSTRAINT KEY_COLUMN_USAGE_POSITION_IN_UNIQUE_CONSTRAINT_CHECK
CHECK ( ( POSITION_IN_UNIQUE_CONSTRAINT IS NULL
        AND
          NOT ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
              ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
                FROM REFERENTIAL_CONSTRAINTS ) )
      OR
        ( POSITION_IN_UNIQUE_CONSTRAINT IS NOT NULL
        AND
          ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
            ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
              FROM REFERENTIAL_CONSTRAINTS )
        AND
          NOT EXISTS ( SELECT 1
                      FROM KEY_COLUMN_USAGE
                      GROUP BY CONSTRAINT_CATALOG,
                              CONSTRAINT_SCHEMA,
                              CONSTRAINT_NAME
                      HAVING COUNT ( POSITION_IN_UNIQUE_CONSTRAINT )
                        <> MAX ( POSITION_IN_UNIQUE_CONSTRAINT ) ) ) )
)

```

**Description**

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME are the catalog name, unqualified schema name, qualified identifier of the table name, and the column name of the column that participates in the unique, primary key, or foreign key constraint being described.
- 3) The value of ORDINAL\_POSITION is the ordinal position of the specific column in the constraint being described. If the constraint described is a key of cardinality 1 (one), then the value of ORDINAL\_POSITION is always 1 (one).
- 4) Case:
  - a) If the constraint being described is a foreign key constraint, then the value of POSITION\_IN\_UNIQUE\_CONSTRAINT is the ordinal position of the referenced column corresponding to the referencing column being described, in the corresponding unique key constraint.
  - b) Otherwise, the value of POSITION\_IN\_UNIQUE\_CONSTRAINT is the null value.

## 6.30 KEY\_PERIOD\_USAGE base table

### Function

The KEY\_PERIOD\_USAGE table has one or more rows for each row in the TABLE\_CONSTRAINTS table that has a CONSTRAINT\_TYPE of “UNIQUE”, “PRIMARY KEY”, or “FOREIGN KEY”. The rows list the period that is referenced in <without overlap specification> in each unique constraint, and the <referencing period specification> in each foreign key constraint.

### Definition

```
CREATE TABLE KEY_PERIOD_USAGE (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_PERIOD_TABLE_CATALOG_NOT_NULL
    NOT NULL,
  TABLE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_PERIOD_TABLE_SCHEMA_NOT_NULL
    NOT NULL,
  TABLE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT KEY_PERIOD_TABLE_NAME_NOT_NULL
    NOT NULL,
  PERIOD_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT KEY_PERIOD_USAGE_PRIMARY_KEY
    PRIMARY KEY ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME,
                  PERIOD_NAME ),

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

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

  CONSTRAINT KEY_PERIOD_CONSTRAINT_TYPE_CHECK
    CHECK (
      ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
      ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
        FROM TABLE_CONSTRAINTS
        WHERE CONSTRAINT_TYPE IN
          ( 'UNIQUE', 'PRIMARY KEY', 'FOREIGN KEY' ) ) ),
```

)

## Description

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and PERIOD\_NAME are the catalog name, unqualified schema name, qualified identifier of the table name, and the period name of the period that participates in the unique, primary key, or foreign key constraint being described.



## 6.31 METHOD\_SPECIFICATION\_PARAMETERS base table

### Function

The METHOD\_SPECIFICATION\_PARAMETERS base table has one row for each SQL parameter of each method specification described in the METHOD\_SPECIFICATIONS base table.

### Definition

```
CREATE TABLE METHOD_SPECIFICATION_PARAMETERS (
  SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
  ORDINAL_POSITION          INFORMATION_SCHEMA.CARDINAL_NUMBER
  CONSTRAINT METHOD_SPECIFICATION_PARAMETER_POSITION_NOT_NULL
    NOT NULL,
  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
    CHECK ( ORDINAL_POSITION > 0 ),
  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
    CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                      FROM METHOD_SPECIFICATION_PARAMETERS
                      GROUP BY SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) ),
  DTD_IDENTIFIER            INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PARAMETER_MODE            INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT METHOD_SPECIFICATION_PARAMETER_MODE_CHECK
    CHECK ( PARAMETER_MODE IN
            ( 'IN' ) ),
  IS_RESULT                 INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_IS_RESULT_NOT_NULL
    NOT NULL,
  AS_LOCATOR                INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_AS_LOCATOR_NOT_NULL
    NOT NULL,
  PARAMETER_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FROM_SQL_SPECIFIC_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FROM_SQL_SPECIFIC_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  FROM_SQL_SPECIFIC_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY
    PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                  ORDINAL_POSITION ),

  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY
    FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
      REFERENCES METHOD_SPECIFICATIONS,

  CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_CHECK_DATA_TYPE
    CHECK (
      ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
        'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
      ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
            OBJECT_TYPE, DTD_IDENTIFIER
        FROM DATA_TYPE_DESCRIPTOR ) )
```

)

## Description

- 1) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method whose parameters are being described.
- 2) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the fully qualified name of the user-defined type with which the SQL-invoked method is associated.
- 3) The value of ORDINAL\_POSITION is the ordinal position of the SQL parameter in the SQL-invoked method.
- 4) The values of PARAMETER\_MODE have the following meanings:

IN	The SQL parameter being described is an input parameter.
----	--

- 5) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, SPECIFIC\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the parameter being described.
- 6) The values of IS\_RESULT have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT parameter of a type-preserving function.

- 7) The values of AS\_LOCATOR have the following meanings:

YES	The parameter is passed AS LOCATOR.
NO	The parameter is not passed AS LOCATOR.

- 8) Case:
  - a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of PARAMETER\_NAME is that <SQL parameter name>.
  - b) Otherwise, the value of PARAMETER\_NAME is the null value.
- 9) FROM\_SQL\_SPECIFIC\_CATALOG, FROM\_SQL\_SPECIFIC\_SCHEMA, and FROM\_SQL\_SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the parameter being described.

## 6.32 METHOD\_SPECIFICATIONS base table

*This Subclause is modified by Subclause 14.3, “METHOD\_SPECIFICATIONS base table”, in ISO/IEC 9075-13.*

### Function

The METHOD\_SPECIFICATIONS base table has one row for each method specification.

### Definition

```
CREATE TABLE METHOD_SPECIFICATIONS (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    METHOD_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_STATIC                  INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT METHOD_SPECIFICATIONS_IS_STATIC_NOT_NULL
        NOT NULL,
    IS_OVERRIDING              INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT METHOD_SPECIFICATIONS_IS_OVERRIDING_NOT_NULL
        NOT NULL,
    IS_CONSTRUCTOR             INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_NOT_NULL
        NOT NULL,
    METHOD_LANGUAGE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_LANGUAGE_CHECK
        CHECK ( METHOD_LANGUAGE IN
            ( 'SQL', 'ADA', 'C',
              'COBOL', 'FORTRAN', 'MUMPS', 'PASCAL', 'PLI' ) ),
    PARAMETER_STYLE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_PARAMETER_STYLE_CHECK
        CHECK ( PARAMETER_STYLE IN
            ( 'SQL', 'GENERAL' ) ),
    DTD_IDENTIFIER             INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_DETERMINISTIC           INFORMATION_SCHEMA.YES_OR_NO,
    SQL_DATA_ACCESS            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_NOT_NULL
        NOT NULL
    CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_CHECK
        CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                      'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
    IS_NULL_CALL               INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT METHOD_SPECIFICATIONS_IS_NULL_CALL_NOT_NULL
        NOT NULL,
    TO_SQL_SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    AS_LOCATOR                 INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT METHOD_SPECIFICATIONS_AS_LOCATOR_NOT_NULL
        NOT NULL,
```

## 6.32 METHOD\_SPECIFICATIONS base table

```

CREATED                                INFORMATION_SCHEMA.TIME_STAMP,
RESULT_CAST_FROM_DTD_IDENTIFIER        INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR                 INFORMATION_SCHEMA.YES_OR_NO,

CONSTRAINT METHOD_SPECIFICATIONS_PRIMARY_KEY
    PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
        REFERENCES SCHEMATA,

CONSTRAINT METHOD_SPECIFICATIONS_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES MATCH FULL,

CONSTRAINT METHOD_SPECIFICATIONS_CHECK_DATA_TYPE
    CHECK (
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
          USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
              OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) ),

CONSTRAINT METHOD_SPECIFICATIONS_COMBINATIONS
    CHECK (
        ( ( METHOD_LANGUAGE = 'SQL'
          AND
            IS_DETERMINISTIC IS NULL )
        OR
        ( METHOD_LANGUAGE <> 'SQL'
          AND
            IS_DETERMINISTIC IS NOT NULL ) ) ),

CONSTRAINT METHOD_SPECIFICATIONS_IS_CONSTRUCTOR_COMBINATION_CHECK
    CHECK ( IS_CONSTRUCTOR = 'NO' OR IS_OVERRIDING = 'NO' ),

CONSTRAINT METHOD_SPECIFICATIONS_SAME_SCHEMA
    CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
        ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),

CONSTRAINT METHOD_SPECIFICATIONS_CHECK_RESULT_CAST
    CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NULL )
    OR
        ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
        AND
          RESULT_CAST_AS_LOCATOR IS NOT NULL
        AND
          ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
            'USER-DEFINED TYPE', DTD_IDENTIFIER ) IN
          ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, RESULT_CAST_FROM_DTD_IDENTIFIER
            FROM DATA_TYPE_DESCRIPTOR )
        )
    )

```

)

## Description

- 1) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked method being described.
- 2) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of the user-defined type with which the SQL-invoked method is associated.
- 3) The values of METHOD\_NAME is the identifier of the method name of the SQL-invoked method being described.

- 4) The values of IS\_STATIC have the following meanings:

YES	The SQL-invoked routine is a static method.
NO	The SQL-invoked routine is not a static method.

- 5) The values of IS\_OVERRIDING have the following meanings:

YES	The SQL-invoked method is an overriding method.
NO	The SQL-invoked method is an original method.

- 6) The values of IS\_CONSTRUCTOR have the following meanings:

YES	The SQL-invoked method is an SQL-invoked constructor method.
NO	The SQL-invoked method is not an SQL-invoked constructor method.

- 7) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, USER\_DEFINED\_TYPE\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the result type of the method.

- 8) The values of IS\_NULL\_CALL have the following meanings:

YES	The SQL-invoked routine is a null-call function.
NO	The SQL-invoked routine is not a null-call function.

- 9) The value of METHOD\_LANGUAGE is the explicit or implicit <language name> contained in the method specification being described.

**6.32 METHOD\_SPECIFICATIONS base table**

10) Case:

- a) If the method being defined specifies LANGUAGE SQL, then the values of IS\_DETERMINISTIC, PARAMETER\_STYLE, TO\_SQL\_SPECIFIC\_CATALOG, TO\_SQL\_SPECIFIC\_SCHEMA, and TO\_SQL\_SPECIFIC\_NAME are the null value.
- b) Otherwise:
  - i) The values of IS\_DETERMINISTIC have the following meanings:

YES	The method is deterministic.
NO	The method is possibly not deterministic.

- ii) The values of PARAMETER\_STYLE have the following meanings:

SQL	The method specification specified PARAMETER STYLE SQL.
GENERAL	The method specification specified PARAMETER STYLE GENERAL.

- iii) TO\_SQL\_SPECIFIC\_CATALOG, TO\_SQL\_SPECIFIC\_SCHEMA, and TO\_SQL\_SPECIFIC\_NAME are catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked method being described.

11) The values of SQL\_DATA\_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

12) The values of AS\_LOCATOR have the following meanings:

YES	The return value is passed AS LOCATOR.
NO	The return value is not passed AS LOCATOR.

13) The value of CREATED is

Case:

- a) If Feature T011, “Timestamp in Information Schema”, is supported and the SQL-implementation knows the value of CURRENT\_TIMESTAMP at the time when the SQL-invoked method specification being described was created, then that value.
- b) Otherwise, the NULL value.

## 14) Case:

- a) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the SQL-invoked method specifies a <result cast>, then the values of RESULT\_CAST\_FROM\_DTD\_IDENTIFIER and RESULT\_CAST\_AS\_LOCATOR are the null value.
- b) Otherwise, SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, SPECIFIC\_NAME, and RESULT\_CAST\_FROM\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the <data type> specified in the <result cast> of the SQL-invoked method being described.

## Case:

- i) If the method specification descriptor of the SQL-invoked method being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of RESULT\_CAST\_AS\_LOCATOR is 'NO'.
- ii) Otherwise, the value of RESULT\_CAST\_AS\_LOCATOR is 'YES'.

## 6.33 PARAMETERS base table

*This Subclause is modified by Subclause 21.2, “PARAMETERS base table”, in ISO/IEC 9075-14.*

### Function

The PARAMETERS table has one row for each SQL parameter of each SQL-invoked routine described in the ROUTINES base table.

### Definition

```
CREATE TABLE PARAMETERS (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDINAL_POSITION         INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT PARAMETERS_POSITION_NOT_NULL
        NOT NULL
    CONSTRAINT PARAMETERS_ORDINAL_POSITION_GREATER_THAN_ZERO_CHECK
        CHECK ( ORDINAL_POSITION > 0 )
    CONSTRAINT PARAMETERS_ORDINAL_POSITION_CONTIGUOUS_CHECK
        CHECK ( 0 = ALL ( SELECT MAX(ORDINAL_POSITION) - COUNT(*)
                           FROM PARAMETERS
                           GROUP BY SPECIFIC_CATALOG,
                                   SPECIFIC_SCHEMA,
                                   SPECIFIC_NAME ) ),
    DTD_IDENTIFIER            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_MODE            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PARAMETER_MODE_NOT_NULL
        NOT NULL
    CONSTRAINT PARAMETER_MODE_CHECK
        CHECK (
            PARAMETER_MODE IN
            ( 'IN', 'OUT', 'INOUT' ) ),
    IS_RESULT                 INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT PARAMETERS_IS_RESULT_NOT_NULL
        NOT NULL,
    AS_LOCATOR                INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT PARAMETERS_AS_LOCATOR_NOT_NULL
        NOT NULL,
    PARAMETER_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    FROM_SQL_SPECIFIC_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_CATALOG   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_SCHEMA    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TO_SQL_SPECIFIC_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PARAMETER_DEFAULT         INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT PARAMETERS_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
                      SPECIFIC_NAME, ORDINAL_POSITION ),
    CONSTRAINT PARAMETERS_UNIQUE_CHECK
```



```

    UNIQUE (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, PARAMETER_NAME),
CONSTRAINT PARAMETERS_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
        REFERENCES SCHEMATA,

CONSTRAINT PARAMETERS_CHECK_DATA_TYPE
    CHECK (
        ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
          SPECIFIC_NAME, 'ROUTINE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
          OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) )
    )

```

## Description

- 1) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine that contains the SQL parameter being described.
- 2) The value of ORDINAL\_POSITION is the ordinal position of the SQL parameter in the SQL-invoked routine.
- 3) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the parameter.
- 4) The values of PARAMETER\_MODE have the following meanings:

IN	The SQL parameter being described is an input parameter.
OUT	The SQL parameter being described is an output parameter.
INOUT	The SQL parameter being described is an input parameter and an output parameter.

- 5) The values of IS\_RESULT have the following meanings:

YES	The parameter is the RESULT parameter of a type-preserving function.
NO	The parameter is not the RESULT parameter of a type-preserving function.

- 6) The values of AS\_LOCATOR have the following meanings:

YES	The parameter is passed AS LOCATOR.
NO	The parameter is not passed AS LOCATOR.

- 7) Case:

**6.33 PARAMETERS base table**

- a) If <SQL parameter name> was specified when the SQL-invoked routine was created, then the value of `PARAMETER_NAME` is that <SQL parameter name>.
  - b) Otherwise, the value of `PARAMETER_NAME` is the null value.
- 8) `FROM_SQL_SPECIFIC_CATALOG`, `FROM_SQL_SPECIFIC_SCHEMA`, and `FROM_SQL_SPECIFIC_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the from-sql routine for the input parameter being described.
- 9) `TO_SQL_SPECIFIC_CATALOG`, `TO_SQL_SPECIFIC_SCHEMA`, and `TO_SQL_SPECIFIC_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the output parameter being described.
- 10) Case:
- a) If <parameter default> was specified when the SQL-invoked routine was created, then the value of `PARAMETER_DEFAULT` is that <parameter default>.
  - b) Otherwise, the value of `PARAMETER_DEFAULT` is the null value.

## 6.34 PERIODS base table

### Function

The PERIODS base table has one row for each period defined for a table. It effectively contains a representation of the period descriptors.

### Definition

```
CREATE TABLE PERIODS (
  TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PERIOD_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  START_COLUMN_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT START_COLUMN_NAME_NOT_NULL NOT NULL,
  END_COLUMN_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT END_COLUMN_NAME_NOT_NULL NOT NULL,

  CONSTRAINT PERIODS_PRIMARY_KEY
    PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME )

  CONSTRAINT PERIODS_FOREIGN_KEY_TABLES
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
      REFERENCES TABLES,

  CONSTRAINT PERIODS_FOREIGN_KEY_COLUMNS_1
    FOREIGN KEY TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, START_COLUMN_NAME )
      REFERENCES COLUMNS,

  CONSTRAINT PERIODS_FOREIGN_KEY_COLUMNS_2
    FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, END_COLUMN_NAME )
      REFERENCES COLUMNS
)
```

### Description

- 1) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table containing the period being described.
- 2) The value of PERIOD\_NAME is the name of the period being described.
- 3) The value of START\_COLUMN\_NAME is the name of the column in the table containing the period being described that acts as the start column of the period.
- 4) The value of END\_COLUMN\_NAME is the name of the column in the table containing the period being described that acts as the end column of the period.

## 6.35 REFERENCED\_TYPES base table

### Function

The REFERENCED\_TYPES table has one row for each reference type. It effectively contains a representation of the referenced type descriptors.

### Definition

```
CREATE TABLE REFERENCED_TYPES (
    OBJECT_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_TYPE_IDENTIFIER INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROOT_DTD_IDENTIFIER     INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT REFERENCED_TYPES_PRIMARY_KEY
        PRIMARY KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ),

    CONSTRAINT REFERENCED_TYPES_CHECK_REFERENCE_TYPE
        CHECK ( ( OBJECT_CATALOG, OBJECT_SCHEMA,
                  OBJECT_NAME, OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR
              WHERE DATA_TYPE = 'REF' ) ),

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR,

    CONSTRAINT REFERENCED_TYPES_FOREIGN_KEY_ROOT_DATA_TYPE_DESCRIPTOR
        FOREIGN KEY ( OBJECT_CATALOG, OBJECT_SCHEMA,
                     OBJECT_NAME, OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
        REFERENCES DATA_TYPE_DESCRIPTOR
)
```

### Description

- 1) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and REFERENCE\_TYPE\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the reference type whose referenced type is being described.
- 2) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME,

**6.35 REFERENCED\_TYPES base table**

OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the referenced type of the reference type.

- 3) The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and ROOT\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, OBJECT\_TYPE, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the root data type of the reference type.

## 6.36 REFERENTIAL\_CONSTRAINTS base table

### Function

The REFERENTIAL\_CONSTRAINTS table has one row for each row in the TABLE\_CONSTRAINTS table that has a CONSTRAINT\_TYPE value of “FOREIGN KEY”.

### Definition

```
CREATE TABLE REFERENTIAL_CONSTRAINTS (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  UNIQUE_CONSTRAINT_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_CATALOG_NOT_NULL
    NOT NULL,
  UNIQUE_CONSTRAINT_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_SCHEMA_NOT_NULL
    NOT NULL,
  UNIQUE_CONSTRAINT_NAME   INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT UNIQUE_CONSTRAINT_NAME_NOT_NULL
    NOT NULL,
  MATCH_OPTION             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_MATCH_OPTION_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_MATCH_OPTION_CHECK
    CHECK ( MATCH_OPTION IN
      ( 'NONE', 'PARTIAL', 'FULL' ) ),
  UPDATE_RULE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_UPDATE_RULE_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_UPDATE_RULE_CHECK
    CHECK ( UPDATE_RULE IN
      ( 'CASCADE',
        'SET NULL',
        'SET DEFAULT',
        'RESTRICT',
        'NO ACTION' ) ),
  DELETE_RULE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENTIAL_DELETE_RULE_NOT_NULL
    NOT NULL
    CONSTRAINT REFERENTIAL_DELETE_RULE_CHECK
    CHECK ( DELETE_RULE IN
      ( 'CASCADE',
        'SET NULL',
        'SET DEFAULT',
        'RESTRICT',
        'NO ACTION' ) ),

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

  CONSTRAINT REFERENTIAL_CONSTRAINTS_CONSTRAINT_TYPE_CHECK
```

**6.36 REFERENTIAL\_CONSTRAINTS base table**

```

CHECK ( ( CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME ) IN
        ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
          FROM TABLE_CONSTRAINTS
          WHERE CONSTRAINT_TYPE = 'FOREIGN KEY' ) ),

CONSTRAINT UNIQUE_CONSTRAINT_CHECK_REFERENCES_UNIQUE_CONSTRAINT
CHECK ( UNIQUE_CONSTRAINT_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
OR
        ( ( UNIQUE_CONSTRAINT_CATALOG, UNIQUE_CONSTRAINT_SCHEMA,
            UNIQUE_CONSTRAINT_NAME ) IN
          ( SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
            FROM TABLE_CONSTRAINTS
            WHERE CONSTRAINT_TYPE IN
              ( 'UNIQUE', 'PRIMARY KEY' ) ) ) )
)

```

**Description**

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described.
- 2) The values of UNIQUE\_CONSTRAINT\_CATALOG, UNIQUE\_CONSTRAINT\_SCHEMA, and UNIQUE\_CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the unique or primary key constraint applied to the referenced column list being described.
- 3) The values of MATCH\_OPTION have the following meanings:

NONE	No <match type> was specified.
PARTIAL	A <match type> of PARTIAL was specified.
FULL	A <match type> of FULL was specified.

- 4) The values of UPDATE\_RULE have the following meanings for a referential constraint that has an <update rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.
SET DEFAULT	A <referential action> of SET DEFAULT was specified.

- 5) The values of DELETE\_RULE have the following meanings for a referential constraint that has a <delete rule>:

NO ACTION	A <referential action> of NO ACTION was specified.
RESTRICT	A <referential action> of RESTRICT was specified.
CASCADE	A <referential action> of CASCADE was specified.
SET NULL	A <referential action> of SET NULL was specified.
SET DEFAULT	A <referential action> of SET DEFAULT was specified.



## 6.37 ROLE\_AUTHORIZATION\_DESCRIPTORs base table

### Function

Contains a representation of the role authorization descriptors.

### Definition

```
CREATE TABLE ROLE_AUTHORIZATION_DESCRIPTORs (
  ROLE_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTOR                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
  IS_GRANTABLE              INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_IS_GRANTABLE_NOT_NULL
    NOT NULL,

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_PRIMARY_KEY
    PRIMARY KEY ( ROLE_NAME, GRANTEE, GRANTOR ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_CHECK_ROLE_NAME
    CHECK ( ROLE_NAME IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS
        WHERE AUTHORIZATION_TYPE = 'ROLE' ) ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_GRANTOR_CHECK
    CHECK ( GRANTOR = '_SYSTEM'
      OR
        GRANTOR IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS ) ),

  CONSTRAINT ROLE_AUTHORIZATION_DESCRIPTORs_GRANTEE_CHECK
    CHECK ( GRANTEE = 'PUBLIC'
      OR
        GRANTEE IN
      ( SELECT AUTHORIZATION_NAME
        FROM AUTHORIZATIONS ) )
)
```

### Description

- 1) The value of ROLE\_NAME is the <role name> of some <role granted> by the <grant role statement> or the <role name> of a <role definition>.
- 2) The value of GRANTEE is an <authorization identifier>, possibly PUBLIC, or <role name> specified as a <grantee> contained in a <grant role statement>, or the <authorization identifier> of the current SQL-session when the <role definition> is executed.
- 3) The value of GRANTOR is the <authorization identifier> of the user or role who granted the role identified by ROLE\_NAME to the user or role identified by the value of GRANTEE, or "\_SYSTEM" to indicate that

the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

4) The values of IS\_GRANTABLE have the following meanings:

YES	The described role is grantable.
NO	The described role is not grantable.

A role is grantable if the WITH ADMIN OPTION is specified in the <grant role statement> or a <role definition> is executed.

## 6.38 ROUTINE\_COLUMN\_USAGE base table

### Function

The ROUTINE\_COLUMN\_USAGE table has one row for each column identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINE_COLUMN_USAGE (
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
        CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
              OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
                FROM COLUMNS ) ),

    CONSTRAINT ROUTINE_COLUMN_USAGE_FOREIGN_KEY_ROUTINE_TABLE_USAGE
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES ROUTINE_TABLE_USAGE
)
```

### Description

- 1) The ROUTINE\_COLUMN\_USAGE table has one row for each column *COL* of a table *TAB* identified by a column reference or column name contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *SIR* being described.
- 2) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SIR*.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.
- 4) The value of COLUMN\_NAME is the name of the column *COL*.

## 6.39 ROUTINE\_PERIOD\_USAGE base table

### Function

The ROUTINE\_PERIOD\_USAGE table has one row for each period identified in the <SQL routine body> of an SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINE_PERIOD_USAGE (
    SPECIFIC_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PERIOD_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_PERIOD_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

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

    CONSTRAINT ROUTINE_PERIOD_USAGE_FOREIGN_KEY_ROUTINE_TABLE_USAGE
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES ROUTINE_TABLE_USAGE
)
```

### Description

- 1) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine *R* being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME and PERIOD\_NAME are the catalog name, unqualified schema name, and qualified identifier of the table name, and the period name, respectively, of the period that is identified in the <SQL routine body> of *R*.

## 6.40 ROUTINE\_PRIVILEGES base table

### Function

The ROUTINE\_PRIVILEGES table has one row for each execute privilege descriptor for an SQL-invoked routine. It effectively contains a representation of the execute privilege descriptors.

### Definition

```
CREATE TABLE ROUTINE_PRIVILEGES (
  GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SPECIFIC_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PRIVILEGE_TYPE    INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT ROUTINE_PRIVILEGES_TYPE_CHECK
    CHECK ( PRIVILEGE_TYPE IN
      ( 'EXECUTE' ) ),
  IS_GRANTABLE      INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT ROUTINE_PRIVILEGES_GRANTABLE_NOT_NULL
    NOT NULL,

  CONSTRAINT ROUTINE_PRIVILEGES_PRIMARY_KEY
    PRIMARY KEY ( GRANTOR, GRANTEE, SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
      SPECIFIC_NAME, PRIVILEGE_TYPE ),

  CONSTRAINT ROUTINE_PRIVILEGES_FOREIGN_KEY_TABLES
    FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
      REFERENCES ROUTINES,

  CONSTRAINT ROUTINE_PRIVILEGES_GRANTOR_CHECK
    CHECK ( GRANTOR = '_SYSTEM'
      OR
        GRANTOR IN
          ( SELECT AUTHORIZATION_NAME
            FROM AUTHORIZATIONS ) ),

  CONSTRAINT ROUTINE_PRIVILEGES GRANTEE_CHECK
    CHECK ( GRANTEE = 'PUBLIC'
      OR
        GRANTEE IN
          ( SELECT AUTHORIZATION_NAME
            FROM AUTHORIZATIONS ) )
)
```

### Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-invoked routine identified by SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME, to the user or role identified by the value of GRANTEE for the privilege being described,

**6.40 ROUTINE\_PRIVILEGES base table**

or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the privilege being described is granted.
- 3) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine on which the privilege being described has been granted.
- 4) The values of PRIVILEGE\_TYPE have the following meanings:

EXECUTE	The user has EXECUTE privilege on the SQL-invoked routine identified by SPECIFIC_CATALOG, SPECIFIC_SCHEMA, and SPECIFIC_NAME.
---------	---

- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

## 6.41 ROUTINE\_ROUTINE\_USAGE base table

### Function

The ROUTINE\_ROUTINE\_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINE_ROUTINE_USAGE (
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_CHECK_REFERENCES_ROUTINES
        CHECK ( ROUTINE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME ) IN
              ( SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME
                FROM ROUTINES ) ),

    CONSTRAINT ROUTINE_ROUTINE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

### Description

- 1) The ROUTINE\_ROUTINE\_USAGE table has one row for each SQL-invoked routine *R1* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL routine *R2*.
- 2) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R2*.
- 3) The values of ROUTINE\_CATALOG, ROUTINE\_SCHEMA, and ROUTINE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R1*.

## 6.42 ROUTINE\_SEQUENCE\_USAGE base table

### Function

The ROUTINE\_SEQUENCE\_USAGE table has one row for each external sequence generator identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINE_SEQUENCE_USAGE (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_CHECK_REFERENCES_SEQUENCES
        CHECK ( SEQUENCE_CATALOG NOT IN
                ( SELECT CATALOG_NAME
                  FROM SCHEMATA )
        OR
        ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) IN
        ( SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
          FROM SEQUENCES ) ),

    CONSTRAINT ROUTINE_SEQUENCE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

### Description

- 1) The ROUTINE\_SEQUENCE\_USAGE table has one row for each sequence generator *SEQ* identified by a <sequence generator name> contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *R* being described.
- 2) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.
- 3) The values of SEQUENCE\_CATALOG, SEQUENCE\_SCHEMA, and SEQUENCE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SEQ*.



## 6.43 ROUTINE\_TABLE\_USAGE base table

### Function

The ROUTINE\_TABLE\_USAGE table has one row for each table identified in the <SQL routine body> or in the <parameter default> of an SQL parameter of an SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINE_TABLE_USAGE (
    SPECIFIC_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT ROUTINE_TABLE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT ROUTINE_TABLE_USAGE_CHECK_REFERENCES_TABLES
        CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
              OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM TABLES ) ),

    CONSTRAINT ROUTINE_TABLE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
            REFERENCES ROUTINES
)
```

### Description

- 1) The ROUTINE\_TABLE\_USAGE table has one row for each table *TAB* identified by a <table reference> contained in the <SQL routine body> or in the <parameter default> of any SQL parameter of an SQL-invoked routine *SIR* being described.
- 2) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SIR*.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.

## 6.44 ROUTINES base table

*This Subclause is modified by Subclause 19.5, “ROUTINES base table”, in ISO/IEC 9075-4.*

*This Subclause is modified by Subclause 14.5, “ROUTINES base table”, in ISO/IEC 9075-13.*

*This Subclause is modified by Subclause 21.3, “ROUTINES base table”, in ISO/IEC 9075-14.*

### Function

The ROUTINES base table has one row for each SQL-invoked routine.

### Definition

```
CREATE TABLE ROUTINES (
    SPECIFIC_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_CATALOG           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_SCHEMA            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_TYPE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ROUTINE_TYPE_NOT_NULL
    NOT NULL
    CONSTRAINT ROUTINE_TYPE_CHECK
    CHECK ( ROUTINE_TYPE IN
        ( 'PROCEDURE', 'FUNCTION',
          'INSTANCE METHOD', 'STATIC METHOD', 'CONSTRUCTOR METHOD' ) ),
    DTD_IDENTIFIER            INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ROUTINE_BODY              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT ROUTINE_BODY_NOT_NULL
    NOT NULL
    CONSTRAINT ROUTINE_BODY_CHECK
    CHECK ( ROUTINE_BODY IN
        ( 'SQL', 'EXTERNAL' ) ),
    ROUTINE_DEFINITION        INFORMATION_SCHEMA.CHARACTER_DATA,
    EXTERNAL_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    EXTERNAL_LANGUAGE         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT EXTERNAL_LANGUAGE_CHECK
    CHECK ( EXTERNAL_LANGUAGE IN
        ( 'ADA', 'C', 'COBOL',
          'FORTRAN', 'MUMPS', 'PASCAL', 'PLI' ) ),
    PARAMETER_STYLE           INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PARAMETER_STYLE_CHECK
    CHECK ( PARAMETER_STYLE IN
        ( 'SQL', 'GENERAL' ) ),
    IS_DETERMINISTIC          INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT ROUTINES_IS_DETERMINISTIC_NOT_NULL
    NOT NULL,
```

```

SQL_DATA_ACCESS          INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_SQL_DATA_ACCESS_NOT_NULL
    NOT NULL
CONSTRAINT    ROUTINES_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN
        ( 'NO SQL', 'CONTAINS SQL',
          'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
IS_NULL_CALL             INFORMATION_SCHEMA.YES_OR_NO,
SQL_PATH                  INFORMATION_SCHEMA.CHARACTER_DATA,
SCHEMA_LEVEL_ROUTINE     INFORMATION_SCHEMA.YES_OR_NO
CONSTRAINT ROUTINES_SCHEMA_LEVEL_ROUTINE_NOT_NULL
    NOT NULL,
MAX_DYNAMIC_RESULT_SETS  INFORMATION_SCHEMA.CARDINAL_NUMBER,
IS_USER_DEFINED_CAST     INFORMATION_SCHEMA.YES_OR_NO,
IS_IMPLICITLY_INVOCABLE  INFORMATION_SCHEMA.YES_OR_NO,
SECURITY_TYPE            INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT ROUTINES_SECURITY_TYPE_CHECK
    CHECK ( SECURITY_TYPE IN
        ( 'DEFINER', 'INVOKER', 'IMPLEMENTATION DEFINED' ) ),
TO_SQL_SPECIFIC_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
TO_SQL_SPECIFIC_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
AS_LOCATOR               INFORMATION_SCHEMA.YES_OR_NO,
CREATED                  INFORMATION_SCHEMA.TIME_STAMP,
LAST_ALTERED             INFORMATION_SCHEMA.TIME_STAMP,
NEW_SAVEPOINT_LEVEL      INFORMATION_SCHEMA.YES_OR_NO,
IS_UDT_DEPENDENT         INFORMATION_SCHEMA.YES_OR_NO
CONSTRAINT ROUTINES_IS_UDT_DEPENDENT_NOT_NULL
    NOT NULL,
RESULT_CAST_FROM_DTD_IDENTIFIER  INFORMATION_SCHEMA.SQL_IDENTIFIER,
RESULT_CAST_AS_LOCATOR           INFORMATION_SCHEMA.YES_OR_NO,

CONSTRAINT ROUTINES_PRIMARY_KEY
    PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

CONSTRAINT ROUTINES_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( ROUTINE_CATALOG, ROUTINE_SCHEMA )
        REFERENCES SCHEMATA,

CONSTRAINT ROUTINES_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES
            MATCH FULL,

CONSTRAINT ROUTINES_COMBINATIONS
    CHECK ( ( ROUTINE_BODY = 'SQL'
        AND
            ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NULL )
        OR
            ( ROUTINE_BODY = 'EXTERNAL'
        AND
            ( EXTERNAL_NAME, EXTERNAL_LANGUAGE, PARAMETER_STYLE ) IS NOT NULL ) ),

CONSTRAINT ROUTINES_SAME_SCHEMA
    CHECK ( ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
        ( ROUTINE_CATALOG, ROUTINE_SCHEMA )

```

## 6.44 ROUTINES base table

```

OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
  ( MODULE_CATALOG, MODULE_SCHEMA )
OR ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA ) =
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA ) ),

CONSTRAINT ROUTINES_CHECK_RESULT_TYPE
  CHECK ( ( ROUTINE_TYPE = 'PROCEDURE'
    AND
      DTD_IDENTIFIER IS NULL )
    OR
      ( ROUTINE_TYPE <> 'PROCEDURE'
    AND
      ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
        'ROUTINE', DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) ) ),

CONSTRAINT ROUTINES_CHECK_RESULT_CAST
  CHECK ( ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NULL
    AND
      RESULT_CAST_AS_LOCATOR IS NULL )
    OR
      ( RESULT_CAST_FROM_DTD_IDENTIFIER IS NOT NULL
    AND
      RESULT_CAST_AS_LOCATOR IS NOT NULL
    AND
      ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
        'ROUTINE', RESULT_CAST_FROM_DTD_IDENTIFIER ) IN
        ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, DTD_IDENTIFIER
          FROM DATA_TYPE_DESCRIPTOR ) ) )
)

```

## Description

- 1) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine being described.
- 2) The values of ROUTINE\_CATALOG, ROUTINE\_SCHEMA, and ROUTINE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the routine name of the SQL-invoked routine being described.
- 3) 04 The values of MODULE\_CATALOG, MODULE\_SCHEMA, and MODULE\_NAME are the null value.
- 4) Case:
  - a) If the SQL-invoked routine being described was defined as a method of a user-defined type, then the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type name of this user-defined type.
  - b) Otherwise, the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the null value.

5) The values of ROUTINE\_TYPE have the following meanings:

PROCEDURE	The SQL-invoked routine being described is an SQL-invoked procedure.
FUNCTION	The SQL-invoked routine being described is an SQL-invoked function that is not an SQL-invoked method.
INSTANCE METHOD	The SQL-invoked routine being described is an SQL-invoked method that is neither a static SQL-invoked method nor an SQL-invoked constructor method.
STATIC METHOD	The SQL-invoked routine being described is a static SQL-invoked method.
CONSTRUCTOR METHOD	The SQL-invoked routine being described is an SQL-invoked constructor method.

6) If the SQL-invoked routine being described is an SQL-invoked procedure, then DTD\_IDENTIFIER is the null value; otherwise, SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the result type of the SQL-invoked routine being described.

7) The values of ROUTINE\_BODY have the following meanings:

SQL	The SQL-invoked routine being described is an SQL routine.
EXTERNAL	The SQL-invoked routine being described is an external routine.

8) The values of SQL\_DATA\_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

9) The values of IS\_DETERMINISTIC have the following meanings:

YES	DETERMINISTIC was specified when the SQL-invoked routine was defined.
NO	DETERMINISTIC was not specified when the SQL-invoked routine was defined.

10) The values of IS\_NULL\_CALL have the following meanings:

YES	The SQL-invoked routine is a function and returns null if any of its parameters are null.
NO	The SQL-invoked routine is a function and its return value is determined by invoking the routine.
<i>null</i>	The routine being described is a procedure.

11) Case:

- a) If the SQL-invoked routine being described is an SQL routine, and the SQL-invoked routine is not contained in an SQL-server module definition, and the character representation of the <routine body> that defined the SQL-invoked routine can be represented without truncation, then the value of ROUTINE\_DEFINITION is that character representation.
- b) Otherwise, the value of ROUTINE\_DEFINITION is the null value.

12) Case:

- a) If the SQL-invoked routine being described is an external routine, then:
  - i) The value of EXTERNAL\_NAME is the external name of the external routine.
  - ii) The value of EXTERNAL\_LANGUAGE is the language of the external routine.
  - iii) The value of PARAMETER\_STYLE is the SQL parameter passing style of the external routine.
- b) Otherwise, the values of EXTERNAL\_NAME, EXTERNAL\_LANGUAGE and PARAMETER\_STYLE are the null value.

13) Case:

- a) If the routine being described is an SQL routine, then the value of SQL\_PATH is the SQL-path of the routine being described.
- b) Otherwise, the value of SQL\_PATH is the null value.

14) Case:

- a) If the SQL-invoked routine is a schema-level routine, then the value of SCHEMA\_LEVEL\_ROUTINE is 'YES'.
- b) Otherwise, the value of SCHEMA\_LEVEL\_ROUTINE is 'NO'.

15) The value of MAX\_DYNAMIC\_RESULT\_SETS is

Case:

- a) If the routine being described is an SQL-invoked procedure defined by an <SQL-invoked routine> for which <maximum returned result sets> was specified, then the value of <maximum returned result sets>.
- b) Otherwise, 0 (zero).

16) The values of IS\_USER\_DEFINED\_CAST have the following meanings:

YES	The SQL-invoked routine is a function that is a user-defined cast function.
NO	The function SQL-invoked routine is a function that is not a user-defined cast function.
<i>null</i>	The SQL-invoked routine is a procedure.

17) The values of IS\_IMPLICITLY\_INVOCABLE have the following meanings:

YES	The user-defined cast function is implicitly invocable.
NO	The user-defined cast function is not implicitly invocable.
<i>null</i>	The routine is not a user-defined cast function.

18) The values of SECURITY\_TYPE have the following meanings:

DEFINER	The routine has the security characteristic DEFINER.
INVOKER	The routine has the security characteristic INVOKER.
IMPLEMENTATION DEFINED	The external routine has the security characteristic IMPLEMENTATION DEFINED.
<i>null</i>	The SQL-invoked routine is not an external routine and Feature T324, “Explicit security for SQL routines” is not implemented.

19) TO\_SQL\_SPECIFIC\_CATALOG, TO\_SQL\_SPECIFIC\_SCHEMA and TO\_SQL\_SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the to-sql routine for the result type of the SQL-invoked routine being described.

20) The values of AS\_LOCATOR have the following meanings:

YES	The return value of the SQL-invoked routine being described is passed AS LOCATOR.
NO	The return value of the SQL-invoked routine being described is not passed AS LOCATOR.
<i>null</i>	The SQL-invoked routine is a procedure.

21) If Feature T272, “Enhanced savepoint management”, is not implemented, then the value of NEW\_SAVEPOINT\_LEVEL is null; otherwise, the values of NEW\_SAVEPOINT\_LEVEL have the following meanings:

YES	The SQL-invoked routine is an SQL-invoked function or is an SQL-invoked procedure that specifies NEW SAVEPOINT LEVEL.
NO	The SQL-invoked routine is an SQL-invoked procedure that does not specify NEW SAVEPOINT LEVEL or specifies OLD SAVEPOINT LEVEL.

22) The values of IS\_UDT\_DEPENDENT have the following meanings:

YES	The SQL-invoked routine being described is dependent on a user-defined type.
NO	The SQL-invoked routine being described is not dependent on a user-defined type.

23) Case:

- a) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the SQL-invoked routine specifies a <result cast>, then the values of RESULT\_CAST\_FROM\_DTD\_IDENTIFIER and RESULT\_CAST\_AS\_LOCATOR are the null value.
- b) Otherwise, SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, SPECIFIC\_NAME, and RESULT\_CAST\_FROM\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the <data type> specified in the <result cast> of the SQL-invoked routine being described.

Case:

- i) If the routine descriptor of the SQL-invoked routine being described does not include an indication that the <data type> specified in the <result cast> has a locator indication, then the value of RESULT\_CAST\_AS\_LOCATOR is 'NO'.
- ii) Otherwise, the value of RESULT\_CAST\_AS\_LOCATOR is 'YES'.

24) The value of CREATED is

Case:

- a) If Feature T011, “Timestamp in Information Schema”, is supported and the SQL-implementation knows the value of CURRENT\_TIMESTAMP at the time when the SQL-invoked routine being described was created, then that value.
- b) Otherwise, the NULL value.

25) The value of LAST\_ALTERED is

Case:

- a) If Feature T011, “Timestamp in Information Schema”, is supported and the SQL-implementation knows the value of CURRENT\_TIMESTAMP at the time when the SQL-invoked routine being described was last altered, then that value.
- b) Otherwise, the NULL value.

This value is identical to the value of CREATED for SQL-invoked routines that have never been altered.



## 6.45 SCHEMATA base table

### Function

The SCHEMATA table has one row for each schema.

### Definition

```
CREATE TABLE SCHEMATA (
  CATALOG_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_OWNER          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT SCHEMA_OWNER_NOT_NULL
    NOT NULL,
  DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL
    NOT NULL,
  DEFAULT_CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL
    NOT NULL,
  DEFAULT_CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL
    NOT NULL,
  SQL_PATH              INFORMATION_SCHEMA.CHARACTER_DATA,

  CONSTRAINT SCHEMATA_PRIMARY_KEY
    PRIMARY KEY ( CATALOG_NAME, SCHEMA_NAME ),

  CONSTRAINT SCHEMATA_FOREIGN_KEY_AUTHORIZATIONS
    FOREIGN KEY ( SCHEMA_OWNER )
    REFERENCES AUTHORIZATIONS,

  CONSTRAINT SCHEMATA_FOREIGN_KEY_CATALOG_NAME
    FOREIGN KEY ( CATALOG_NAME )
    REFERENCES CATALOG_NAME,

  CONSTRAINT SCHEMATA_CHECK_REFERENCES_CHARACTER_SETS
    CHECK ( DEFAULT_CHARACTER_SET_CATALOG NOT IN
      ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
      ( DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
        DEFAULT_CHARACTER_SET_NAME ) IN
      ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
        CHARACTER_SET_NAME
        FROM CHARACTER_SETS ) )
)
```

### Description

- 1) The value of CATALOG\_NAME is the name of the catalog of the schema described by this row.

**6.45 SCHEMATA base table**

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

## 6.46 SEQUENCES base table

### Function

The SEQUENCES base table has one row for each external sequence generator.

### Definition

```
CREATE TABLE SEQUENCES (
    SEQUENCE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    START_VALUE                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_START_VALUE_NOT_NULL
    NOT NULL,
    MINIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_MINIMUM_VALUE_NOT_NULL
    NOT NULL,
    MAXIMUM_VALUE              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_MAXIMUM_VALUE_NOT_NULL
    NOT NULL,
    INCREMENT                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SEQUENCES_INCREMENT_NOT_NULL
    NOT NULL,
    CYCLE_OPTION               INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT SEQUENCES_CYCLE_OPTION_NOT_NULL
    NOT NULL,

    CONSTRAINT SEQUENCES_PRIMARY_KEY
    PRIMARY KEY (SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME),

    CONSTRAINT SEQUENCES_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA )
    REFERENCES SCHEMATA,

    CONSTRAINT SEQUENCES_CHECK_DATA_TYPE
    CHECK ( ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA,
              SEQUENCE_NAME, 'SEQUENCE', DTD_IDENTIFIER ) IN
            ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                  OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
              FROM DATA_TYPE_DESCRIPTOR ) )
)
```

### Description

- 1) The values of SEQUENCE\_CATALOG, SEQUENCE\_SCHEMA, and SEQUENCE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the sequence generator being described.

**6.46 SEQUENCES base table**

- 2) The values of SEQUENCE\_CATALOG, SEQUENCE\_SCHEMA, SEQUENCE\_NAME, and DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the sequence generator.
- 3) The values of START\_VALUE, MINIMUM\_VALUE, MAXIMUM\_VALUE, and INCREMENT are the character representations of start value, minimum value, maximum value, and increment, respectively, of the sequence generator being described.
- 4) The values of CYCLE\_OPTION have the following meanings:

YES	The cycle option of the sequence generator is CYCLE.
NO	The cycle option of the sequence generator is NO CYCLE.

## 6.47 SQL\_CONFORMANCE base table

### Function

The SQL\_CONFORMANCE base table has one row for each conformance element (part, package, feature, and subfeature) identified by ISO/IEC 9075.

### Definition

```
CREATE TABLE SQL_CONFORMANCE
(
    TYPE                                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_CONFORMANCE_TYPE_CHECK
    CHECK ( TYPE IN ( 'PART', 'FEATURE', 'SUBFEATURE', 'PACKAGE' ) ),
    ID                                  INFORMATION_SCHEMA.CHARACTER_DATA,
    SUB_ID                              INFORMATION_SCHEMA.CHARACTER_DATA,
    NAME                                INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_CONFORMANCE_NAME_NOT_NULL
    NOT NULL,
    SUB_NAME                            INFORMATION_SCHEMA.CHARACTER_DATA,
    IS_SUPPORTED                        INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT SQL_CONFORMANCE_IS_SUPPORTED_NOT_NULL
    NOT NULL,
    IS_VERIFIED_BY                      INFORMATION_SCHEMA.CHARACTER_DATA,
    COMMENTS                            INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_CONFORMANCE_PRIMARY_KEY
    PRIMARY KEY ( TYPE, ID, SUB_ID ),

    CONSTRAINT SQL_CONFORMANCE_CHECK_SUPPORTED_VERIFIED
    CHECK ( IS_SUPPORTED = 'YES'
    OR
    IS_VERIFIED_BY IS NULL )
)
```

### Description

- 1) The SQL\_CONFORMANCE table consists of exactly one row for each SQL part, package, feature, and subfeature defined in ISO/IEC 9075.
- 2) The values of TYPE have the following meanings:

PART	the conformance element described is a Part of ISO/IEC 9075.
FEATURE	the conformance element described is an optional feature of ISO/IEC 9075.
SUBFEA-TURE	the conformance element described is a subfeature of an optional feature of ISO/IEC 9075.

PACKAGE	the conformance element described is a package of features of ISO/IEC 9075.
---------	---

- 3) The ID and NAME columns identify the conformance element described.
- 4) If the conformance element is a subfeature, then the SUB\_ID and SUB\_NAME columns identify the subfeature by the subfeature identifier and name assigned to it. Otherwise, the values of SUB\_ID and SUB\_NAME are each a character string consisting of a single space.
- 5) The IS\_SUPPORTED column is 'YES' if an SQL-implementation fully supports that conformance element described when SQL-data in the identified catalog is accessed through that implementation and is 'NO' if the SQL-implementation does not fully support that conformance element described when accessing SQL-data in that catalog.
- 6) If full support for the conformance element described has been verified by testing, then the IS\_VERIFIED\_BY column shall contain information identifying the conformance test used to verify the conformance claim; otherwise, IS\_VERIFIED\_BY shall be the null value.
- 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 COMMENTS column contains any implementer comments pertinent to the identified SQL part, package, feature, or subfeature.

## 6.48 SQL\_IMPLEMENTATION\_INFO base table

*This Subclause is modified by Subclause 9.1, “SQL\_IMPLEMENTATION\_INFO base table”, in ISO/IEC 9075-3.*

### Function

The SQL\_IMPLEMENTATION\_INFO base table has one row for each SQL-implementation information item defined by ISO/IEC 9075.

### Definition

```
CREATE TABLE SQL_IMPLEMENTATION_INFO (
    IMPLEMENTATION_INFO_ID          INFORMATION_SCHEMA.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,
    COMMENTS                       INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
        PRIMARY KEY ( IMPLEMENTATION_INFO_ID ),

    CONSTRAINT SQL_IMPLEMENTATION_INFO_CHECK_INTEGER_EXCLUDES_CHARACTER
        CHECK ( INTEGER_VALUE IS NULL
            OR
                CHARACTER_VALUE IS NULL )
)
```

### Description

- 1) The SQL\_IMPLEMENTATION\_INFO table consists of exactly one row for each SQL-implementation information item defined in ISO/IEC 9075.
- 2) The IMPLEMENTATION\_INFO\_ID and IMPLEMENTATION\_INFO\_NAME columns identify the SQL-implementation information item by the integer and name assigned to it.
- 3) Depending on the type of information, the value is present in either INTEGER\_VALUE or CHARACTER\_VALUE; the other column is the null value.
- 4) The COMMENTS column is intended for any implementer comments pertinent to the identified item.

## 6.49 SQL\_SIZING base table

*This Subclause is modified by Subclause 9.2, “SQL\_SIZING base table”, in ISO/IEC 9075-3.*

### Function

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

### Definition

```
CREATE TABLE SQL_SIZING (
    SIZING_ID                INFORMATION_SCHEMA.CARDINAL_NUMBER,
    SIZING_NAME              INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL
            NOT NULL,
    SUPPORTED_VALUE          INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COMMENTS                INFORMATION_SCHEMA.CHARACTER_DATA,

    CONSTRAINT SQL_SIZING_PRIMARY_KEY
        PRIMARY KEY (SIZING_ID ),

    CONSTRAINT SQL_SIZING_SIZING_NAME_UNIQUE
        UNIQUE ( SIZING_NAME )
)
```

### Description

- 1) The SQL\_SIZING table shall consist of exactly one row for each SQL sizing item defined in ISO/IEC 9075.
- 2) The SIZING\_ID and SIZING\_NAME columns identify the sizing item by the integer and description assigned to it.
- 3) The values of the SUPPORTED\_VALUE column are:

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

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



## 6.50 TABLE\_CONSTRAINTS base table

### Function

The TABLE\_CONSTRAINTS table has one row for each table constraint associated with a table. It effectively contains a representation of the table constraint descriptors.

### Definition

```
CREATE TABLE TABLE_CONSTRAINTS (
  CONSTRAINT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  CONSTRAINT_TYPE          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT CONSTRAINT_TYPE_NOT_NULL
    NOT NULL
  CONSTRAINT CONSTRAINT_TYPE_CHECK
    CHECK ( CONSTRAINT_TYPE IN
      ( 'UNIQUE', 'PRIMARY KEY',
        'FOREIGN KEY', 'CHECK' ) ),
  TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_CATALOG_NOT_NULL
    NOT NULL,
  TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_SCHEMA_NOT_NULL
    NOT NULL,
  TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TABLE_CONSTRAINTS_TABLE_NAME_NOT_NULL
    NOT NULL,
  IS_DEFERRABLE            INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT TABLE_CONSTRAINTS_IS_DEFERRABLE_NOT_NULL
    NOT NULL,
  INITIALLY_DEFERRED      INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT TABLE_CONSTRAINTS_INITIALLY_DEFERRED_NOT_NULL
    NOT NULL,
  ENFORCED                 INFORMATION_SCHEMA.YES_OR_NO
  CONSTRAINT TABLE_CONSTRAINTS_ENFORCED_NOT_NULL
    NOT NULL,

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

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

  CONSTRAINT TABLE_CONSTRAINTS_CHECK_VIEWS
    CHECK ( TABLE_CATALOG NOT IN
      ( SELECT CATALOG_NAME
        FROM SCHEMATA )
    OR
```

**6.50 TABLE\_CONSTRAINTS base table**

```

( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
  ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
    FROM TABLES
    WHERE TABLE_TYPE <> 'VIEW' ) ) ),

CONSTRAINT UNIQUE_TABLE_PRIMARY_KEY_CHECK
CHECK ( UNIQUE ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLE_CONSTRAINTS
                  WHERE CONSTRAINT_TYPE = 'PRIMARY KEY' ) )
)

```

**Description**

- 1) The values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the constraint being described. If the <table constraint definition> or <add table constraint definition> that defined the constraint did not specify a <constraint name>, then the values of CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, and CONSTRAINT\_NAME are implementation-defined.
- 2) The values of CONSTRAINT\_TYPE have the following meanings:

FOREIGN KEY	The constraint being described is a foreign key constraint.
UNIQUE	The constraint being described is a unique constraint.
PRIMARY KEY	The constraint being described is a primary key constraint.
CHECK	The constraint being described is a check constraint.

- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, the unqualified schema name, and the qualified identifier of the name of the table to which the table constraint being described applies.
- 4) The values of IS\_DEFERRABLE have the following meanings:

YES	The table constraint is deferrable.
NO	The table constraint is not deferrable.

- 5) The values of INITIALLY\_DEFERRED have the following meanings:

YES	The table constraint is initially deferred.
NO	The table constraint is initially immediate.

- 6) The values of ENFORCED have the following meanings:

**6.50 TABLE\_CONSTRAINTS base table**

YES	The table constraint is enforced.
NO	The table constraint is not enforced.

## 6.51 TABLE\_METHOD\_PRIVILEGES base table

### Function

The TABLE\_METHOD\_PRIVILEGES base table has one row for each table/method privilege descriptor. It effectively contains a representation of the table/method privilege descriptors.

### Definition

```
CREATE TABLE TABLE_METHOD_PRIVILEGES (
    GRANTOR                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_GRANTABLE           INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT TABLE_METHOD_PRIVILEGES_IS_GRANTABLE_NOT_NULL
        NOT NULL,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
            SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
            REFERENCES ROUTINES,

    CONSTRAINT TABLE_METHOD_PRIVILEGES_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
            OR
                GRANTOR IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) ),

    CONSTRAINT TABLE_METHOD_PRIVILEGES GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
            OR
                GRANTEE IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) )
```

)

## Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted a table/method privilege on the table identified by TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME, and the method of the identified table's structured type identified by the SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME, to the user or role identified by the value of GRANTEE for the table/method privilege being described, or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the table/method privilege being described is granted.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described was granted.
- 4) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the method on which the privilege being described was granted.
- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

## 6.52 TABLE\_PRIVILEGES base table

### Function

The TABLE\_PRIVILEGES table has one row for each table privilege descriptor. It effectively contains a representation of the table privilege descriptors.

### Definition

```
CREATE TABLE TABLE_PRIVILEGES (
    GRANTOR                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TABLE_PRIVILEGES_TYPE_CHECK
        ( 'SELECT', 'INSERT', 'DELETE', 'UPDATE',
          'TRIGGERS', 'REFERENCES' ) ),
    IS_GRANTABLE          INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT TABLE_PRIVILEGES_GRANTABLE_NOT_NULL
        NOT NULL,
    WITH_HIERARCHY        INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT TABLE_PRIVILEGES_WITH_HIERARCHY_NOT_NULL
        NOT NULL,

    CONSTRAINT TABLE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                      PRIVILEGE_TYPE ),

    CONSTRAINT TABLE_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
            REFERENCES TABLES,

    CONSTRAINT TABLE_PRIVILEGES_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
              OR
                GRANTOR IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) ),

    CONSTRAINT TABLE_PRIVILEGES GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
              OR
                GRANTEE IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) )
```

)

## Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted table privileges, on the table identified by TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME, to the user or role identified by the value of GRANTEE for the table privilege being described, or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the table privilege being described is granted.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the table on which the privilege being described has been granted.
- 4) The values of PRIVILEGE\_TYPE have the following meanings:

SELECT	The user has SELECT privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
DELETE	The user has DELETE privileges on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.
INSERT	The user will automatically be granted INSERT privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
UPDATE	The user will automatically be granted UPDATE privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
REFER- ENCES	The user will automatically be granted REFERENCES privileges on any columns that may be added to the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME in the future.
TRIGGER	The user has TRIGGER privilege on the table identified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME.

- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

- 6) The values of WITH\_HIERARCHY have the following meanings:

YES	The privilege being described was granted WITH HIERARCHY OPTION and is thus grantable.
NO	The privilege being described was not granted WITH HIERARCHY OPTION and is thus not grantable.



## 6.53 TABLES base table

*This Subclause is modified by Subclause 25.12, “TABLES base table”, in ISO/IEC 9075-9.*

### Function

The TABLES table contains one row for each table including views. It effectively contains a representation of the table descriptors.

### Definition

```
CREATE TABLE TABLES (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_TYPE             INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TABLE_TYPE_NOT_NULL
        NOT NULL,
    CONSTRAINT TABLE_TYPE_CHECK
        CHECK ( TABLE_TYPE IN
            ( 'BASE TABLE', 'VIEW', 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY',
              'SYSTEM VERSIONED' ) ),
    SELF_REFERENCING_COLUMN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_GENERATION     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT REFERENCE_GENERATION_CHECK
        CHECK ( REFERENCE_GENERATION IN
            ( 'SYSTEM GENERATED', 'USER GENERATED', 'DERIVED' ) ),
    USER_DEFINED_TYPE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_INSERTABLE_INTO        INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT IS_INSERTABLE_INTO_NOT_NULL
        NOT NULL,
    IS_TYPED                  INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT IS_TYPED_NOT_NULL
        NOT NULL,
    COMMIT_ACTION            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT COMMIT_ACTIONCHECK
        CHECK ( COMMIT_ACTION IN
            ( 'DELETE', 'PRESERVE' ) ),

    CONSTRAINT TABLES_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT TABLES_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( TABLE_CATALOG, TABLE_SCHEMA )
            REFERENCES SCHEMATA,

    CONSTRAINT TABLES_CHECK_TABLE_IN_COLUMNS
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM COLUMNS ) ),
```

## IWD 9075-11:2011(E)

### 6.53 TABLES base table

```
CONSTRAINT TABLES_FOREIGN_KEY_USER_DEFINED_TYPES
    FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME )
    REFERENCES USER_DEFINED_TYPES MATCH FULL,

CONSTRAINT TABLES_TYPED_TABLE_CHECK
    CHECK ( ( IS_TYPED = 'YES'
              AND
              ( ( USER_DEFINED_TYPE_CATALOG,
                  USER_DEFINED_TYPE_SCHEMA,
                  USER_DEFINED_TYPE_NAME,
                  SELF_REFERENCING_COLUMN_NAME,
                  REFERENCE_GENERATION ) IS NOT NULL
              AND
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                SELF_REFERENCING_COLUMN_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                  COLUMN_NAME
                FROM COLUMNS
                WHERE IS_SELF_REFERENCING = 'YES' ) ) )
    OR
    ( IS_TYPED = 'NO'
      AND
      ( USER_DEFINED_TYPE_CATALOG,
        USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME,
        SELF_REFERENCING_COLUMN_NAME,
        REFERENCE_GENERATION ) IS NULL ) ),

CONSTRAINT TABLES_SELF_REFERENCING_COLUMN_CHECK
    CHECK ( ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NULL
    OR ( SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION ) IS NOT NULL ),

CONSTRAINT TABLES_TEMPORARY_TABLE_CHECK
    CHECK ( ( TABLE_TYPE IN ( 'GLOBAL TEMPORARY', 'LOCAL TEMPORARY' ) )
    = ( COMMIT_ACTION IS NOT NULL ) ),

CONSTRAINT TABLES_CHECK_NOT_VIEW
    CHECK ( NOT EXISTS (
        SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM TABLES
        WHERE TABLE_TYPE = 'VIEW'
    EXCEPT
        SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM VIEWS ) )
)
```

## Description

- 1) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the fully qualified name of the table.
- 2) The values of TABLE\_TYPE have the following meanings:

BASE TABLE	The table being described is a persistent base table.
VIEW	The table being described is a viewed table.
GLOBAL TEMPORARY	The table being described is a global temporary table.
LOCAL TEMPORARY	The table being described is a created local temporary table.
SYSTEM VERSIONED	The table being described is a system-versioned table.

3) The value of SELF\_REFERENCING\_COLUMN\_NAME is the name of the self-referencing column of the table, if the table is a typed table. Otherwise, the value of SELF\_REFERENCING\_COLUMN\_NAME is the null value.

4) The values of COMMIT\_ACTION have the following meanings:

DELETE	A <table commit action> of DELETE was specified.
PRESERVE	A <table commit action> of PRESERVE was specified.
<i>null</i>	The table being described is not a temporary table.

5) The values of REFERENCE\_GENERATION have the following meanings:

SYSTEM GENERATED	The values of the self-referencing column of the table are generated by the SQL-server.
USER GENERATED	The values of the self-referencing column of the table are generated by the user.
DERIVED	The values of the self-referencing column of the table are generated from columns of the table.
<i>null</i>	The table being described does not have a self-referencing column.

6) If the table being described is a table of a structured type *TY*, then the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the fully qualified name of *TY*; otherwise, the values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the null value.

7) The values of IS\_INSERTABLE\_INTO have the following meanings:

a) If the SQL-implementation supports Feature T111, “Updatable joins, unions, and columns”, then:

YES	The table being described is insertable-into.
NO	The table being described is not insertable-into.

**IWD 9075-11:2011(E)**  
**6.53 TABLES base table**

b) Otherwise:

YES	The table being described is insertable-into and simply updatable.
NO	The table being described is not insertable-into or not simply updatable.

8) The values of IS\_TYPED have the following meanings:

YES	The table being described is a typed table.
NO	The table being described is not a typed table.

## 6.54 TRANSFORMS base table

### Function

The TRANSFORMS base table has one row for each transform.

### Definition

```
CREATE TABLE TRANSFORMS (
    USER_DEFINED_TYPE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GROUP_NAME                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRANSFORM_TYPE                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRANSFORM_TYPE_NOT_NULL
        NOT NULL
    CONSTRAINT TRANSFORM_TYPE_CHECK
        CHECK ( TRANSFORM_TYPE IN
            ( 'TO SQL', 'FROM SQL' ) ),

    CONSTRAINT TRANSFORMS_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME, GROUP_NAME, TRANSFORM_TYPE ),

    CONSTRAINT TRANSFORMS_TYPES_FOREIGN_KEY
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME )
            REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT TRANSFORMS_ROUTINES_FOREIGN_KEY
        FOREIGN KEY (SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME)
            REFERENCES ROUTINES
)
```

### Description

- 1) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, and USER\_DEFINED\_TYPE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the user-defined type for which the transform being described applies.
- 2) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-invoked routine that acts as the transform function for the transform being described. The value of GROUP\_NAME is the identifier that acts as the name of a transform group.
- 3) The values of TRANSFORM\_TYPE have the following meanings:

TO SQL	The transform being described identifies a to-sql function
FROM SQL	The transform being described identifies a from-sql function

## 6.55 TRANSLATIONS base table

### Function

The TRANSLATIONS table has one row for each character transliteration descriptor.

### Definition

```
CREATE TABLE TRANSLATIONS (
  TRANSLATION_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TRANSLATION_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TRANSLATION_NAME        INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SOURCE_CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_CATALOG_NOT_NULL
  NOT NULL,
  SOURCE_CHARACTER_SET_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_SCHEMA_NOT_NULL
  NOT NULL,
  SOURCE_CHARACTER_SET_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_SOURCE_CHARACTER_SET_NAME_NOT_NULL
  NOT NULL,
  TARGET_CHARACTER_SET_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_CATALOG_NOT_NULL
  NOT NULL,
  TARGET_CHARACTER_SET_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_SCHEMA_NOT_NULL
  NOT NULL,
  TARGET_CHARACTER_SET_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TARGET_CHARACTER_SET_NAME_NOT_NULL
  NOT NULL,
  TRANSLATION_SOURCE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_CATALOG_NOT_NULL
  NOT NULL,
  TRANSLATION_SOURCE_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_SCHEMA_NOT_NULL
  NOT NULL,
  TRANSLATION_SOURCE_NAME      INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT TRANSLATIONS_TRANSLATION_SOURCE_NAME_NOT_NULL
  NOT NULL,

  CONSTRAINT TRANSLATIONS_PRIMARY_KEY
  PRIMARY KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME ),

  CONSTRAINT TRANSLATIONS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( TRANSLATION_CATALOG, TRANSLATION_SCHEMA )
  REFERENCES SCHEMATA,

  CONSTRAINT TRANSLATIONS_FOREIGN_KEY_ROUTINES
  FOREIGN KEY ( TRANSLATION_SOURCE_CATALOG, TRANSLATION_SOURCE_SCHEMA,
    TRANSLATION_SOURCE_NAME )
  REFERENCES ROUTINES,

  CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_SOURCE
```

**6.55 TRANSLATIONS base table**

```

CHECK ( SOURCE_CHARACTER_SET_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
OR
        ( SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
          SOURCE_CHARACTER_SET_NAME ) IN
        ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME
          FROM CHARACTER_SETS ) ),

CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_TARGET
CHECK ( TARGET_CHARACTER_SET_CATALOG NOT IN
        ( SELECT CATALOG_NAME
          FROM SCHEMATA )
OR
        ( TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
          TARGET_CHARACTER_SET_NAME ) IN
        ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
          CHARACTER_SET_NAME
          FROM CHARACTER_SETS ) )
)

```

**Description**

- 1) The values of TRANSLATION\_CATALOG, TRANSLATION\_SCHEMA, and TRANSLATION\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the transliteration being described.
- 2) The values of SOURCE\_CHARACTER\_SET\_CATALOG, SOURCE\_CHARACTER\_SET\_SCHEMA, and SOURCE\_CHARACTER\_SET\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set specified as the source for the transliteration.
- 3) The values of TARGET\_CHARACTER\_SET\_CATALOG, TARGET\_CHARACTER\_SET\_SCHEMA, and TARGET\_CHARACTER\_SET\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set specified as the target for the transliteration.
- 4) The values of TRANSLATION\_SOURCE\_CATALOG, TRANSLATION\_SOURCE\_SCHEMA, and TRANSLATION\_SOURCE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine used for the transliteration.



## 6.56 TRIGGERED\_UPDATE\_COLUMNS base table

### Function

The TRIGGERED\_UPDATE\_COLUMNS base table has one row for each column identified by a <column name> in a <trigger column list> of a <trigger definition>.

### Definition

```
CREATE TABLE TRIGGERED_UPDATE_COLUMNS (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    EVENT_OBJECT_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT EVENT_OBJECT_CATALOG_NOT_NULL
    NOT NULL,
    EVENT_OBJECT_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT EVENT_OBJECT_SCHEMA_NOT_NULL
    NOT NULL,
    EVENT_OBJECT_TABLE        INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT EVENT_OBJECT_TABLE_NOT_NULL
    NOT NULL,
    EVENT_OBJECT_COLUMN       INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_PRIMARY_KEY
    PRIMARY KEY
    ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME, EVENT_OBJECT_COLUMN ),

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_EVENT_MANIPULATION_CHECK
    CHECK ( ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME ) IN
    ( SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME
      FROM TRIGGERS
      WHERE EVENT_MANIPULATION = 'UPDATE' ) ),

    CONSTRAINT TRIGGERED_UPDATE_COLUMNS_FOREIGN_KEY_COLUMNS
    FOREIGN KEY ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA,
                  EVENT_OBJECT_TABLE, EVENT_OBJECT_COLUMN )
    REFERENCES COLUMNS
)
```

### Description

- 1) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, schema name, and trigger name of the trigger being described.
- 2) The values of EVENT\_OBJECT\_CATALOG, EVENT\_OBJECT\_SCHEMA, and EVENT\_OBJECT\_TABLE are the catalog name, schema name, and table name of the table containing the column being described. The TRIGGERED\_UPDATE\_COLUMNS base table has one row for each column contained in an explicitly specified <trigger column list> of a trigger whose trigger event is UPDATE.

## 6.57 TRIGGER\_COLUMN\_USAGE base table

### Function

The TRIGGER\_COLUMN\_USAGE base table has one row for each explicitly or implicitly identified column of a table referenced in the <trigger definition> of the trigger being described.

### Definition

```
CREATE TABLE TRIGGER_COLUMN_USAGE (
    TRIGGER_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_COLUMN_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

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

    CONSTRAINT TRIGGER_COLUMN_USAGE_FOREIGN_KEY_TRIGGER_TABLE_USAGE
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TRIGGER_TABLE_USAGE
)
```

### Description

- 1) The TRIGGER\_COLUMN\_USAGE base table has one row for each column *COL* of a table *TAB* identified by a column reference or column name contained in the <trigger definition> of a trigger *TR* being described.
- 2) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TR*.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.
- 4) The value of COLUMN\_NAME is the name of the column *COL*.

## 6.58 TRIGGER\_PERIOD\_USAGE base table

### Function

The TRIGGER\_PERIOD\_USAGE base table has one row for each period identified in the <trigger definition> of the trigger being described.

### Definition

```
CREATE TABLE TRIGGER_PERIOD_USAGE (
    TRIGGER_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PERIOD_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_PERIOD_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

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

    CONSTRAINT TRIGGER_PERIOD_USAGE_FOREIGN_KEY_TRIGGER_TABLE_USAGE
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
        REFERENCES TRIGGER_TABLE_USAGE
)
```

### Description

- 1) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the trigger *TR* being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME and PERIOD\_NAME are the catalog name, unqualified schema name, and qualified identifier of the table name, and the period name, respectively, of the period that is identified in the <trigger definition> of *TR*.

## 6.59 TRIGGER\_ROUTINE\_USAGE base table

### Function

The TRIGGER\_ROUTINE\_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <trigger definition>.

### Definition

```
CREATE TABLE TRIGGER_ROUTINE_USAGE (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

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

    CONSTRAINT TRIGGER_ROUTINE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

### Description

- 1) The TRIGGER\_ROUTINE\_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <trigger definition>.
- 2) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the trigger being described.
- 3) The values of SPECIFIC\_CATALOG, SPECIFIC\_SCHEMA, and SPECIFIC\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

## 6.60 TRIGGER\_SEQUENCE\_USAGE base table

### Function

The TRIGGER\_SEQUENCE\_USAGE table has one row for each external sequence generator identified in a trigger.

### Definition

```
CREATE TABLE TRIGGER_SEQUENCE_USAGE (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SEQUENCE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                     SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ),

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_CHECK_REFERENCES_SEQUENCES
        CHECK ( SEQUENCE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME ) IN
              ( SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME
                FROM SEQUENCES ) ),

    CONSTRAINT TRIGGER_SEQUENCE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

### Description

- 1) The TRIGGER\_SEQUENCE\_USAGE table has one row for each sequence generator *SEQ* identified by a <sequence generator name> contained in the <triggered action> of a trigger *TR* being described.
- 2) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TR*.
- 3) The values of SEQUENCE\_CATALOG, SEQUENCE\_SCHEMA, and SEQUENCE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *SEQ*.

## 6.61 TRIGGER\_TABLE\_USAGE base table

### Function

The TRIGGER\_TABLE\_USAGE base table has one row for each table identified by a <table name> contained in the <search condition> of a <triggered action> or referenced in a <triggered SQL statement> of a <trigger definition>.

### Definition

```
CREATE TABLE TRIGGER_TABLE_USAGE      (
    TRIGGER_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT TRIGGER_TABLE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
                      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT TRIGGER_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 ) ),

    CONSTRAINT TRIGGER_TABLE_USAGE_FOREIGN_KEY_TRIGGERS
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME )
        REFERENCES TRIGGERS
)
```

### Description

- 1) The TRIGGER\_TABLE\_USAGE base table has one row for each table *TAB* identified by a <table reference> contained in the <trigger definition> of a trigger *TR* being described..
- 2) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TR*.
- 3) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of *TAB*.

## 6.62 TRIGGERS base table

### Function

The TRIGGERS base table has one row for each trigger. It effectively contains a representation of the trigger descriptors.

### Definition

```
CREATE TABLE TRIGGERS (
    TRIGGER_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TRIGGER_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    EVENT_MANIPULATION       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_EVENT_MANIPULATION_CHECK
        CHECK ( EVENT_MANIPULATION IN
            ( 'INSERT', 'DELETE', 'UPDATE' ) ),
    EVENT_OBJECT_CATALOG     INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_CATALOG_NOT_NULL
        NOT NULL,
    EVENT_OBJECT_SCHEMA      INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_SCHEMA_NOT_NULL
        NOT NULL,
    EVENT_OBJECT_TABLE       INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TRIGGERS_EVENT_OBJECT_TABLE_NOT_NULL
        NOT NULL,
    ACTION_ORDER             INFORMATION_SCHEMA.CARDINAL_NUMBER
    CONSTRAINT TRIGGERS_ACTION_ORDER_NOT_NULL
        NOT NULL,
    ACTION_CONDITION         INFORMATION_SCHEMA.CHARACTER_DATA,
    ACTION_STATEMENT         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_STATEMENT_NOT_NULL
        NOT NULL,
    ACTION_ORIENTATION       INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_ORIENTATION_CHECK
        CHECK ( ACTION_ORIENTATION IN
            ( 'ROW', 'STATEMENT' ) ),
    ACTION_TIMING            INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT TRIGGERS_ACTION_TIMING_CHECK
        CHECK ( ACTION_TIMING IN
            ( 'BEFORE', 'AFTER', 'INSTEAD OF' ) ),
    ACTION_REFERENCE_OLD_TABLE INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_NEW_TABLE INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_OLD_ROW  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ACTION_REFERENCE_NEW_ROW  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CREATED                  INFORMATION_SCHEMA.TIME_STAMP,

    CONSTRAINT TRIGGERS_PRIMARY_KEY
        PRIMARY KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME ),

    CONSTRAINT TRIGGERS_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
            REFERENCES SCHEMATA,
```

```

CONSTRAINT TRIGGERS_REFERENCES_TABLES
CHECK ( EVENT_OBJECT_CATALOG <>
      ANY ( SELECT CATALOG_NAME
            FROM SCHEMATA )
      OR
      ( EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE ) IN
      ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
        FROM TABLES ) )
)

```

## Description

- 1) The values of TRIGGER\_CATALOG, TRIGGER\_SCHEMA, and TRIGGER\_NAME are the fully qualified name of the trigger being described.
- 2) The values of EVENT\_MANIPULATION have the following meaning:

INSERT	The <trigger event> is INSERT.
DELETE	The <trigger event> is DELETE.
UPDATE	The <trigger event> is UPDATE.

- 3) The values of EVENT\_OBJECT\_CATALOG, EVENT\_OBJECT\_SCHEMA, and EVENT\_OBJECT\_TABLE are the fully qualified name of the subject table of the trigger being described.
- 4) The values of ACTION\_TIMING have the following meaning:

BEFORE	The <trigger action time> is BEFORE.
AFTER	The <trigger action time> is AFTER.
INSTEAD OF	The <trigger action time> is INSTEAD OF.

- 5) The value of ACTION\_REFERENCE\_OLD\_TABLE is the <old transition table name> of the trigger being described.
- 6) The value of ACTION\_REFERENCE\_NEW\_TABLE is the <new transition table name> of the trigger being described.
- 7) The value of ACTION\_REFERENCE\_OLD\_ROW is the <old transition variable name> of the trigger being described.
- 8) The value of ACTION\_REFERENCE\_NEW\_ROW is the <new transition variable name> of the trigger being described.
- 9) The value of ACTION\_ORDER is the ordinal position of the trigger in the list of triggers with the same EVENT\_OBJECT\_CATALOG, EVENT\_OBJECT\_SCHEMA, EVENT\_OBJECT\_TABLE, EVENT\_MANIPULATION, CONDITION\_TIMING, and ACTION\_ORIENTATION.



- 10) The value of ACTION\_CONDITION is a character representation of the <search condition> in the <triggered action> of the trigger being described.
- 11) ACTION\_STATEMENT is a character representation of the <triggered SQL statement> in the <triggered action> of the trigger being described.
- 12) The values of ACTION\_ORIENTATION have the following meanings:

ROW	The <triggered action> specifies FOR EACH ROW.
STATE- MENT	The <triggered action> specifies FOR EACH STATEMENT.

- 13) The value of CREATED is

Case:

- a) If Feature T011, “Timestamp in Information Schema”, is supported and the SQL-implementation knows the value of CURRENT\_TIMESTAMP at the time when the trigger being described was created, then that value.
- b) Otherwise, the NULL value.

## 6.63 USAGE\_PRIVILEGES base table

*This Subclause is modified by Subclause 25.13, “USAGE\_PRIVILEGES base table”, in ISO/IEC 9075-9.*

*This Subclause is modified by Subclause 14.7, “USAGE\_PRIVILEGES base table”, in ISO/IEC 9075-13.*

*This Subclause is modified by Subclause 21.4, “USAGE\_PRIVILEGES base table”, in ISO/IEC 9075-14.*

### Function

The USAGE\_PRIVILEGES table has one row for each usage privilege descriptor. It effectively contains a representation of the usage privilege descriptors.

### Definition

```
CREATE TABLE USAGE_PRIVILEGES (
    GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
    OBJECT_TYPE     INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USAGE_PRIVILEGES_OBJECT_TYPE_CHECK
        CHECK ( OBJECT_TYPE IN
            ( 'DOMAIN', 'CHARACTER SET',
              'COLLATION', 'TRANSLATION', 'SEQUENCE' ) ),
    IS_GRANTABLE    INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT USAGE_PRIVILEGES_IS_GRANTABLE_NOT_NULL

    NOT NULL,

    CONSTRAINT USAGE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, OBJECT_CATALOG, OBJECT_SCHEMA,
                      OBJECT_NAME, OBJECT_TYPE ),

    CONSTRAINT USAGE_PRIVILEGES_CHECK_REFERENCES_OBJECT
        CHECK ( ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE ) IN
            ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN'
              FROM DOMAINS
            UNION
              SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
                'CHARACTER SET'
              FROM CHARACTER_SETS
            UNION
              SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, 'COLLATION'
              FROM COLLATIONS
            UNION
              SELECT TRANSLATION_CATALOG, TRANSLATION_SCHEMA, TRANSLATION_NAME,
                'TRANSLATION'
              FROM TRANSLATIONS
            UNION
              SELECT SEQUENCE_CATALOG, SEQUENCE_SCHEMA, SEQUENCE_NAME,
                'SEQUENCE'
              FROM SEQUENCES ) ),
```

```

CONSTRAINT USAGE_PRIVILEGES_GRANTOR_CHECK
    CHECK ( GRANTOR = '_SYSTEM'
        OR
            GRANTOR IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) ),

CONSTRAINT USAGE_PRIVILEGES GRANTEE_CHECK
    CHECK ( GRANTEE = 'PUBLIC'
        OR
            GRANTEE IN
            ( SELECT AUTHORIZATION_NAME
              FROM AUTHORIZATIONS ) )
)

```

## Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted usage privileges, on the object of the type identified by OBJECT\_TYPE that is identified by OBJECT\_CATALOG, OBJECT\_SCHEMA, and OBJECT\_NAME, to the user or role identified by the value of GRANTEE for the usage privilege being described, or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.
- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the usage privilege being described is granted.
- 3) 0914 The values of OBJECT\_CATALOG, OBJECT\_SCHEMA, and OBJECT\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the object to which the privilege applies.
- 4) 13 The values of OBJECT\_TYPE have the following meanings:

DOMAIN	The object to which the privilege applies is a domain.
CHARACTER SET	The object to which the privilege applies is a character set.
COLLATION	The object to which the privilege applies is a collation.
TRANSLATION	The object to which the privilege applies is a transliteration.
SEQUENCE	The object to which the privilege applies is a sequence generator.

- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

## 6.64 USER\_DEFINED\_TYPE\_PRIVILEGES base table

### Function

The USER\_DEFINED\_TYPE\_PRIVILEGES table has one row for each user-defined type privilege descriptor. It effectively contains a representation of the privilege descriptors.

### Definition

```
CREATE TABLE USER_DEFINED_TYPE_PRIVILEGES (
    GRANTOR                                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE                                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE                         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT PRIVILEGE_TYPE_CHECK
        CHECK ( PRIVILEGE_TYPE = 'TYPE USAGE' ),
    IS_GRANTABLE                           INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_IS_GRANTABLE_NOT_NULL
        NOT NULL,

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY (GRANTOR, GRANTEE,
                     USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                     USER_DEFINED_TYPE_NAME, PRIVILEGE_TYPE ),

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_FOREIGN_KEY_USER_DEFINED_TYPE
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                     USER_DEFINED_TYPE_NAME )
        REFERENCES USER_DEFINED_TYPES,

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES_GRANTOR_CHECK
        CHECK ( GRANTOR = '_SYSTEM'
              OR
                GRANTOR IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) ),

    CONSTRAINT USER_DEFINED_TYPE_PRIVILEGES GRANTEE_CHECK
        CHECK ( GRANTEE = 'PUBLIC'
              OR
                GRANTEE IN
                ( SELECT AUTHORIZATION_NAME
                  FROM AUTHORIZATIONS ) )
)
```

### Description

- 1) The value of GRANTOR is the <authorization identifier> of the user or role who granted access privileges on the TYPE USAGE privilege being described to the user or role identified by the value of GRANTEE,

**6.64 USER\_DEFINED\_TYPE\_PRIVILEGES base table**

or "\_SYSTEM" to indicate that the privileges were granted to the authorization identifier of the creator of the object on which the privileges were granted.

- 2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the user-defined type privilege being described is granted.
- 3) The value of PRIVILEGE\_TYPE has the following meaning:

TYPE USAGE	The user has TYPE USAGE privilege on this user-defined type.
---------------	--

- 4) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable

## 6.65 USER\_DEFINED\_TYPES base table

This Subclause is modified by *Subclause 14.8, “USER\_DEFINED\_TYPES base table”*, in ISO/IEC 9075-13.

### Function

The USER\_DEFINED\_TYPES table has one row for each user-defined type.

### Definition

```
CREATE TABLE USER_DEFINED_TYPES (
    USER_DEFINED_TYPE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    USER_DEFINED_TYPE_CATEGORY      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_USER_DEFINED_TYPE_CATEGORY_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_USER_DEFINED_TYPE_CATEGORY_CHECK
        CHECK ( USER_DEFINED_TYPE_CATEGORY IN
            ( 'STRUCTURED', 'DISTINCT' ) ),
    SOURCE_DTD_IDENTIFIER           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    IS_INSTANTIABLE                INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT USER_DEFINED_TYPES_IS_INSTANTIABLE_NOT_NULL
        NOT NULL,
    IS_FINAL                       INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT USER_DEFINED_TYPES_IS_FINAL_NOT_NULL
        NOT NULL,
    ORDERING_FORM                  INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_NOT_NULL
        NOT NULL
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_CHECK
        CHECK ( ORDERING_FORM IN
            ( 'NONE', 'FULL', 'EQUALS' ) ),
    ORDERING_CATEGORY              INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_ORDERING_CATEGORY_CHECK
        CHECK ( ORDERING_CATEGORY IN
            ( 'RELATIVE', 'MAP', 'STATE' ) ),
    ORDERING_ROUTINE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDERING_ROUTINE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    ORDERING_ROUTINE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    REFERENCE_TYPE                 INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT USER_DEFINED_TYPES_REFERENCE_TYPE_CHECK
        CHECK ( REFERENCE_TYPE IN
            ( 'SYSTEM GENERATED', 'USER GENERATED', 'DERIVED' ) ),
    REF_DTD_IDENTIFIER             INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT USER_DEFINED_TYPES_PRIMARY_KEY
        PRIMARY KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
            USER_DEFINED_TYPE_NAME ),

    CONSTRAINT USER_DEFINED_TYPES_FOREIGN_KEY_SCHEMATA
        FOREIGN KEY ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA )
            REFERENCES SCHEMATA,
```

```

CONSTRAINT USER_DEFINED_TYPES_FOREIGN_KEY_ROUTINES
    FOREIGN KEY ( ORDERING_ROUTINE_CATALOG, ORDERING_ROUTINE_SCHEMA,
                  ORDERING_ROUTINE_NAME )
    REFERENCES ROUTINES,

CONSTRAINT USER_DEFINED_TYPES_CHECK_SOURCE_TYPE
    CHECK ( ( USER_DEFINED_TYPE_CATEGORY = 'STRUCTURED'
              AND
              SOURCE_DTD_IDENTIFIER IS NULL )
            OR
            ( USER_DEFINED_TYPE_CATEGORY = 'DISTINCT'
              AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, SOURCE_DTD_IDENTIFIER ) IS NOT NULL
              AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', SOURCE_DTD_IDENTIFIER )
              IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR
                  WHERE DATA_TYPE NOT IN
                      ( 'ROW', 'REFERENCE', 'USER-DEFINED' ) ) ) ),

CONSTRAINT USER_DEFINED_TYPES_CHECK_USER_GENERATED_REFERENCE_TYPE
    CHECK ( ( REFERENCE_TYPE <> 'USER GENERATED'
              AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                REF_DTD_IDENTIFIER ) NOT IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR ) )
            OR
            ( REFERENCE_TYPE = 'USER GENERATED'
              AND
              ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                REF_DTD_IDENTIFIER ) IN
              ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
                    OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
                  FROM DATA_TYPE_DESCRIPTOR
                  WHERE DATA_TYPE IN
                      ( 'CHARACTER', 'INTEGER' ) ) ) )
    )

```

## Description

- 1) The values of USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA and USER\_DEFINED\_TYPE\_NAME are the fully qualified name of the user-defined type being described.
- 2) The values of USER\_DEFINED\_TYPE\_CATEGORY have the following meanings:

## 6.65 USER\_DEFINED\_TYPES base table

STRUC-TURED	The user-defined type is a structured type.
DISTINCT	The user-defined type is a distinct type.

3) If USER\_DEFINED\_TYPE\_CATEGORY is 'STRUCTURED', then the value of SOURCE\_DTD\_IDENTIFIER is the null value; otherwise, USER\_DEFINED\_TYPE\_CATALOG, USER\_DEFINED\_TYPE\_SCHEMA, USER\_DEFINED\_TYPE\_NAME, and SOURCE\_DTD\_IDENTIFIER are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER, respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the source type of the distinct type.

4) The values of IS\_INSTANTIABLE have the following meanings:

YES	The user-defined type is instantiable.
NO	The user-defined type is not instantiable.

5) The values of IS\_FINAL have the following meanings:

YES	The user-defined type cannot have subtypes.
NO	The user-defined type can have subtypes.

6) The values of ORDERING\_FORM have the following meanings:

NONE	Two values of this type may not be compared.
FULL	Two values of this type may be compared for equality or relative order.
EQUALS	Two values of this type may be compared for equality only.

7) 13 The values of ORDERING\_CATEGORY have the following meanings:

RELATIVE	Two values of this type can be compared with a relative routine.
MAP	Two values of this type may be compared with a map routine.
STATE	Two values of this type may be compared with a state routine.

8) The values of ORDER\_ROUTINE\_CATALOG, ORDER\_ROUTINE\_SCHEMA, and ORDER\_ROUTINE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of the SQL-invoked routine used for ordering the user-defined type.

9) The values of REFERENCE\_TYPE have the following meanings:



SYSTEM GENER- ATED	REF values for tables of this structured type are system generated.
USER GEN- ERATED	REF values for tables of this structured type are user generated of the data type specified by USER GENERATED TYPE.
DERIVED	REF values for tables of this structured type are derived from the columns corresponding to the specified attributes.

- 10) If the value of REFERENCE\_TYPE is not 'USER GENERATED', then the value of REF\_DTD\_IDENTIFI-  
 FIER is the null value; otherwise, USER\_DEFINED\_TYPE\_CATALOG,  
 USER\_DEFINED\_TYPE\_SCHEMA, USER\_DEFINED\_TYPE\_NAME, and REF\_DTD\_IDENTIFIER  
 are the values of OBJECT\_CATALOG, OBJECT\_SCHEMA, OBJECT\_NAME, and DTD\_IDENTIFIER,  
 respectively, of the row in DATA\_TYPE\_DESCRIPTOR that describes the data type of the user-generated  
 REF values of the structured type.

## 6.66 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 original <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_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_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

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

  CONSTRAINT VIEW_COLUMN_USAGE_FOREIGN_KEY_VIEW_TABLE_USAGE
    FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
    REFERENCES VIEW_TABLE_USAGE
)
```

### 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\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME are the 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 original <query expression> of the view being described.

## 6.67 VIEW\_PERIOD\_USAGE base table

### Function

The VIEW\_PERIOD\_USAGE table has one row for each period used in the <view definition> of the view being described.

### Definition

```
CREATE TABLE VIEW_PERIOD_USAGE (
  VIEW_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  VIEW_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG    INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA     INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PERIOD_NAME       INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT VIEW_PERIOD_USAGE_PRIMARY_KEY
    PRIMARY KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, PERIOD_NAME ),

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

  CONSTRAINT VIEW_PERIOD_USAGE_FOREIGN_KEY_VIEW_TABLE_USAGE
    FOREIGN KEY ( VIEW_CATALOG, VIEW_SCHEMA, VIEW_NAME,
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
    REFERENCES VIEW_TABLE_USAGE
)
```

### 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 *V* being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and PERIOD\_NAME are the catalog name, unqualified schema name, qualified identifier, and period name, respectively, of a period that is referenced in the original <query expression> of *V*.

## 6.68 VIEW\_ROUTINE\_USAGE base table

### Function

The VIEW\_ROUTINE\_USAGE table has one row for each SQL-invoked routine identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in a <view definition>.

### Definition

```
CREATE TABLE VIEW_ROUTINE_USAGE (
    TABLE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    SPECIFIC_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER,

    CONSTRAINT VIEW_ROUTINE_USAGE_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                     SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ),

    CONSTRAINT VIEW_ROUTINE_USAGE_CHECK_REFERENCES_VIEWS
        CHECK ( TABLE_CATALOG NOT IN
              ( SELECT CATALOG_NAME
                FROM SCHEMATA )
        OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
              ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                FROM VIEWS ) ),

    CONSTRAINT VIEW_ROUTINE_USAGE_FOREIGN_KEY_ROUTINES
        FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
        REFERENCES ROUTINES
)
```

### Description

- 1) The VIEW\_ROUTINE\_USAGE table has one row for each SQL-invoked routine *R* identified as the subject routine of either a <routine invocation>, a <method reference>, a <method invocation>, or a <static method invocation> contained in the <query expression> of the <view definition> of the view being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the viewed table being described.
- 3) The values of ROUTINE\_CATALOG, ROUTINE\_SCHEMA, and ROUTINE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the specific name of *R*.

## 6.69 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 original <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_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_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

  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 ) ),

  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\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table identified by a <table name> simply contained in a <table reference> that is contained in the original <query expression> of the view being described.

## 6.70 VIEWS base table

### Function

The VIEWS table contains one row for each row in the TABLES table with a TABLE\_TYPE of 'VIEW'. Each row describes the query expression that defines a view. The 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.YES_OR_NO
    CONSTRAINT IS_UPDATABLE_NOT_NULL
        NOT NULL,
    IS_TRIGGER_UPDATABLE   INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT IS_TRIGGER_UPDATABLE_NOT_NULL
        NOT NULL,
    IS_TRIGGER_DELETABLE   INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT IS_TRIGGER_DELETABLE_NOT_NULL
        NOT NULL,
    IS_TRIGGER_INSERTABLE_INTO INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT IS_TRIGGER_INSERTABLE_INTO_NOT_NULL
        NOT NULL,

    CONSTRAINT VIEWS_PRIMARY_KEY
        PRIMARY KEY ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

    CONSTRAINT VIEWS_IN_TABLES_CHECK
        CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
            ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES
              WHERE TABLE_TYPE = 'VIEW' ) ),

    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, TABLE\_SCHEMA, and TABLE\_NAME are the fully qualified name of the viewed table.
- 2) Case:
  - a) If the character representation of the user-specified <query expression> contained in the corresponding view descriptor 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 14 — Any implicit column references that were contained in the original <query expression> associated with the <view definition> are replaced by explicit column references in VIEW\_DEFINITION.

- 3) The values of CHECK\_OPTION have the following meanings:

CASCADED	The corresponding view descriptor indicates that the view has the CHECK OPTION that is to be applied as CASCADED.
LOCAL	The corresponding view descriptor indicates that the view has the CHECK OPTION that is to be applied as LOCAL.
NONE	The corresponding view descriptor indicates that the view does not have the CHECK OPTION.

- 4) The values of IS\_UPDATABLE have the following meanings:

YES	The view is effectively updatable.
NO	The view is not effectively updatable.

- 5) The values of IS\_TRIGGER\_UPDATABLE have the following meanings:

YES	An update INSTEAD OF trigger is defined on the view.
NO	An update INSTEAD OF trigger is not defined on the view.

- 6) The values of IS\_TRIGGER\_DELETABLE have the following meanings:

YES	A delete INSTEAD OF trigger is defined on the view.
NO	A delete INSTEAD OF trigger is not defined on the view.

- 7) The values of IS\_TRIGGER\_INSERTABLE\_INTO have the following meanings:

**IWD 9075-11:2011(E)**  
**6.70 VIEWS base table**

YES	An insert INSTEAD OF trigger is defined on the view.
NO	An insert INSTEAD OF trigger is not defined on the view.



## 7 Conformance

### 7.1 Claims of conformance to SQL/Schemata

No requirements in addition to the requirements of [ISO9075-1], [Clause 8](#), “Conformance”, are required for a claim of conformance to this part of ISO/IEC 9075.

### 7.2 Additional conformance requirements for SQL/Schemata

A claim of conformance that includes a claim of conformance to a feature in this part of ISO/IEC 9075 shall also include a claim of conformance to the same feature, if present, in [ISO9075-2].

### 7.3 Implied feature relationships of SQL/Schemata

**Table 1 — Implied feature relationships of SQL/Schemata**

Feature ID	Feature Name	Implied Feature ID	Implied Feature Description
T011	Timestamp in Information Schema	F411	Time zone specification
T011	Timestamp in Information Schema	F555	Enhanced seconds precision

*(Blank page)*

## Annex A (informative)

### SQL Conformance Summary

*This Annex modifies Annex A, “SQL Conformance Summary”, in ISO/IEC 9075-2.*

The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

- 1) Specifications for Feature F231, “Privilege tables”:
  - a) Subclause 5.19, “COLUMN\_PRIVILEGES view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_PRIVILEGES.
  - b) Subclause 5.25, “DATA\_TYPE\_PRIVILEGES view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DATA\_TYPE\_PRIVILEGES.
  - c) Subclause 5.41, “ROLE\_COLUMN\_GRANTS view”:
    - i) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_COLUMN\_GRANTS.
  - d) Subclause 5.42, “ROLE\_ROUTINE\_GRANTS view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.
  - e) Subclause 5.43, “ROLE\_TABLE\_GRANTS view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_GRANTS.
  - f) Subclause 5.46, “ROLE\_UDT\_GRANTS view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_UDT\_GRANTS.
  - g) Subclause 5.49, “ROUTINE\_PRIVILEGES view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PRIVILEGES.
  - h) Subclause 5.63, “TABLE\_PRIVILEGES view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_PRIVILEGES.
  - i) Subclause 5.74, “UDT\_PRIVILEGES view”:

- i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.UDT\_PRIVILEGES.
  - j) Subclause 5.75, “USAGE\_PRIVILEGES view”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USAGE\_PRIVILEGES.
  - k) Subclause 5.82, “Short name views”:
    - i) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUT\_GRANTS.
- 2) Specifications for Feature F251, “Domain support”:
  - a) Subclause 5.3, “CARDINAL\_NUMBER domain”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CARDINAL\_NUMBER.
  - b) Subclause 5.4, “CHARACTER\_DATA domain”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHARACTER\_DATA.
  - c) Subclause 5.5, “SQL\_IDENTIFIER domain”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IDENTIFIER.
  - d) Subclause 5.6, “TIME\_STAMP domain”:
    - i) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TIME\_STAMP.
  - e) Subclause 5.7, “YES\_OR\_NO domain”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.YES\_OR\_NO.
  - f) Subclause 5.18, “COLUMN\_DOMAIN\_USAGE view”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.
  - g) Subclause 5.28, “DOMAIN\_CONSTRAINTS view”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAIN\_CONSTRAINTS.
  - h) Subclause 5.29, “DOMAINS view”:
    - i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS.
  - i) Subclause 5.82, “Short name views”:

- i) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS\_S.
- ii) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_DOMAINS\_USAGE.

3) Specifications for Feature F341, “Usage tables”:

- a) Subclause 5.13, “CHECK\_CONSTRAINT\_ROUTINE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHECK\_CONSTRAINT\_ROUTINE\_USAGE.
- b) Subclause 5.17, “COLUMN\_COLUMN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_COLUMN\_USAGE.
- c) Subclause 5.18, “COLUMN\_DOMAIN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.
- d) Subclause 5.20, “COLUMN\_UDT\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_UDT\_USAGE.
- e) Subclause 5.22, “CONSTRAINT\_COLUMN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_COLUMN\_USAGE.
- f) Subclause 5.23, “CONSTRAINT\_PERIOD\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.
- g) Subclause 5.24, “CONSTRAINT\_TABLE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE.
- h) Subclause 5.33, “KEY\_COLUMN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE.
- i) Subclause 5.34, “KEY\_PERIOD\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.
- j) Subclause 5.45, “ROLE\_USAGE\_GRANTS view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_USAGE\_GRANTS.
- k) Subclause 5.47, “ROUTINE\_COLUMN\_USAGE view”:

- i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COLUMN\_USAGE.
- l) Subclause 5.48, “ROUTINE\_PERIOD\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- m) Subclause 5.50, “ROUTINE\_ROUTINE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_ROUTINE\_USAGE.
- n) Subclause 5.51, “ROUTINE\_SEQUENCE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.
- o) Subclause 5.52, “ROUTINE\_TABLE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_TABLE\_USAGE.
- p) Subclause 5.68, “TRIGGER\_COLUMN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_COLUMN\_USAGE.
- q) Subclause 5.69, “TRIGGER\_PERIOD\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.
- r) Subclause 5.70, “TRIGGER\_ROUTINE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.
- s) Subclause 5.71, “TRIGGER\_SEQUENCE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.
- t) Subclause 5.72, “TRIGGER\_TABLE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.
- u) Subclause 5.77, “VIEW\_COLUMN\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_COLUMN\_USAGE.
- v) Subclause 5.78, “VIEW\_PERIOD\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- w) Subclause 5.79, “VIEW\_ROUTINE\_USAGE view”:

- i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_ROUTINE\_USAGE.
- x) Subclause 5.80, “VIEW\_TABLE\_USAGE view”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_TABLE\_USAGE.
- y) Subclause 5.82, “Short name views”:
  - i) Without Feature F341, “Usage tables”, conforming SQL language shall not reference the INFORMATION\_SCHEMA.TRIG\_TABLE\_USAGE view.
  - ii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_UPDATE\_COLS.
  - iii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_DOMAIN\_USAGE.
  - iv) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONST\_COL\_USAGE.
  - v) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONST\_TABLE\_USAGE.
  - vi) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE\_S.
  - vii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COL\_USAGE.
  - viii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_TABLE\_USAGE.
  - ix) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_ROUT\_USAGE\_S.
  - x) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTR\_ROUT\_USE\_S.
  - xi) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_ROUT\_USAGE\_S.
  - xii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_SEQ\_USAGE\_S.
  - xiii) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_COLUMN\_USAGE.
  - xiv) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_SEQ\_USAGE\_S.
  - xv) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_COL\_USAGE.
- 4) Specifications for Feature F391, “Long identifiers”:
  - a) Subclause 5.2, “INFORMATION\_SCHEMA\_CATALOG\_NAME base table”:

- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.INFORMATION\_SCHEMA\_CATALOG\_NAME.
- b) Subclause 5.8, “ADMINISTRABLE\_ROLE\_AUTHORIZATIONS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ADMINISTRABLE\_ROLE\_AUTHORIZATIONS.
- c) Subclause 5.11, “ATTRIBUTES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ATTRIBUTES.
- d) Subclause 5.12, “CHARACTER\_SETS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHARACTER\_SETS.
- e) Subclause 5.13, “CHECK\_CONSTRAINT\_ROUTINE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CHECK\_CONSTRAINT\_ROUTINE\_USAGE.
- f) Subclause 5.15, “COLLATIONS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS.
- g) Subclause 5.16, “COLLATION\_CHARACTER\_SET\_APPLICABILITY view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATION\_CHARACTER\_SET\_APPLICABILITY.
- h) Subclause 5.17, “COLUMN\_COLUMN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_COLUMN\_USAGE.
- i) Subclause 5.18, “COLUMN\_DOMAIN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_DOMAIN\_USAGE.
- j) Subclause 5.21, “COLUMNS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.
- k) Subclause 5.22, “CONSTRAINT\_COLUMN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_COLUMN\_USAGE.
- l) Subclause 5.23, “CONSTRAINT\_PERIOD\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.
- m) Subclause 5.24, “CONSTRAINT\_TABLE\_USAGE view”:



- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_TABLE\_USAGE.
- n) Subclause 5.29, “DOMAINS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DOMAINS.
- o) Subclause 5.30, “ELEMENT\_TYPES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES.
- p) Subclause 5.32, “FIELDS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS.
- q) Subclause 5.33, “KEY\_COLUMN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE.
- r) Subclause 5.34, “KEY\_PERIOD\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.
- s) Subclause 5.35, “METHOD\_SPECIFICATION\_PARAMETERS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATION\_PARAMETERS.
- t) Subclause 5.36, “METHOD\_SPECIFICATIONS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.
- u) Subclause 5.37, “PARAMETERS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS.
- v) Subclause 5.39, “REFERENCED\_TYPES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES.
- w) Subclause 5.40, “REFERENTIAL\_CONSTRAINTS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENTIAL\_CONSTRAINTS.
- x) Subclause 5.42, “ROLE\_ROUTINE\_GRANTS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.
- y) Subclause 5.44, “ROLE\_TABLE\_METHOD\_GRANTS view”:

- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.
- z) Subclause 5.47, “ROUTINE\_COLUMN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_COLUMN\_USAGE.
- aa) Subclause 5.48, “ROUTINE\_PERIOD\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- ab) Subclause 5.50, “ROUTINE\_ROUTINE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_ROUTINE\_USAGE.
- ac) Subclause 5.51, “ROUTINE\_SEQUENCE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.
- ad) Subclause 5.52, “ROUTINE\_TABLE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_TABLE\_USAGE.
- ae) Subclause 5.53, “ROUTINES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.
- af) Subclause 5.54, “SCHEMATA view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SCHEMATA.
- ag) Subclause 5.55, “SEQUENCES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES.
- ah) Subclause 5.57, “SQL\_IMPLEMENTATION\_INFO view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPLEMENTATION\_INFO.
- ai) Subclause 5.62, “TABLE\_METHOD\_PRIVILEGES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVILEGES.
- aj) Subclause 5.64, “TABLES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLES.
- ak) Subclause 5.66, “TRANSLATIONS view”:

- i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS.
- al) Subclause 5.67, “TRIGGERED\_UPDATE\_COLUMNS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_UPDATE\_COLUMNS.
- am) Subclause 5.68, “TRIGGER\_COLUMN\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_COLUMN\_USAGE.
- an) Subclause 5.69, “TRIGGER\_PERIOD\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_PERIOD\_USAGE.
- ao) Subclause 5.70, “TRIGGER\_ROUTINE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.
- ap) Subclause 5.71, “TRIGGER\_SEQUENCE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.
- aq) Subclause 5.72, “TRIGGER\_TABLE\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.
- ar) Subclause 5.73, “TRIGGERS view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.
- as) Subclause 5.76, “USER\_DEFINED\_TYPES view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USER\_DEFINED\_TYPES.
- at) Subclause 5.78, “VIEW\_PERIOD\_USAGE view”:
  - i) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- 5) Specifications for Feature F502, “Enhanced documentation tables”:
  - a) Subclause 5.57, “SQL\_IMPLEMENTATION\_INFO view”:
    - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPLEMENTATION\_INFO.
  - b) Subclause 5.58, “SQL\_PACKAGES view”:
    - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_PACKAGES.

- c) Subclause 5.59, “SQL\_PARTS view”:
  - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_PARTS.
- d) Subclause 5.82, “Short name views”:
  - i) Without Feature F502, “Enhanced documentation tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SQL\_IMPL\_INFO.
- 6) Specifications for Feature F521, “Assertions”:
  - a) Subclause 5.10, “ASSERTIONS view”:
    - i) Without Feature F521, “Assertions”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ASSERTIONS.
- 7) Specifications for Feature F651, “Catalog name qualifiers”:
  - a) Subclause 5.2, “INFORMATION\_SCHEMA\_CATALOG\_NAME base table”:
    - i) Without Feature F651, “Catalog name qualifiers”, conforming SQL language shall not reference INFORMATION\_SCHEMA.INFORMATION\_SCHEMA\_CATALOG\_NAME.
- 8) Specifications for Feature F690, “Collation support ”:
  - a) Subclause 5.15, “COLLATIONS view”:
    - i) Without Feature F690, “Collation support ”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS.
  - b) Subclause 5.16, “COLLATION\_CHARACTER\_SET\_APPLICABILITY view”:
    - i) Without Feature F690, “Collation support ”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATION\_CHARACTER\_SET\_APPLICABILITY.
  - c) Subclause 5.82, “Short name views”:
    - i) Without Feature F690, “Collation support ”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLLATIONS\_S.
- 9) Specifications for Feature F695, “Translation support”:
  - a) Subclause 5.66, “TRANSLATIONS view”:
    - i) Without Feature F695, “Translation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS.
  - b) Subclause 5.82, “Short name views”:
    - i) Without Feature F695, “Translation support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSLATIONS\_S.
- 10) Specifications for Feature F696, “Additional translation documentation”:
  - a) Subclause 5.66, “TRANSLATIONS view”:

- i) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATION\_SOURCE\_CATALOG, TRANSLATION\_SOURCE\_SCHEMA, or TRANSLATION\_SOURCE\_NAME.
- b) Subclause 5.82, “Short name views”:
  - i) Without Feature F696, “Additional translation documentation”, conforming SQL language shall not reference TRANSLATIONS\_S.TRANS\_SRC\_CATALOG, TRANSLATIONS\_S.TRANS\_SRC\_SCHEMA, or TRANSLATIONS\_S.TRANS\_SRC\_NAME.

11) Specifications for Feature S023, “Basic structured types”:

- a) Subclause 5.11, “ATTRIBUTES view”:
  - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ATTRIBUTES.
- b) Subclause 5.35, “METHOD\_SPECIFICATION\_PARAMETERS view”:
  - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATION\_PARAMETERS.
- c) Subclause 5.36, “METHOD\_SPECIFICATIONS view”:
  - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.
- d) Subclause 5.82, “Short name views”:
  - i) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ATTRIBUTES\_S.
  - ii) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECS.
  - iii) Without Feature S023, “Basic structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPEC\_PARAMS.

12) Specifications for Feature S024, “Enhanced structured types”:

- a) Subclause 5.27, “DIRECT\_SUPERTYPES view”:
  - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DIRECT\_SUPERTYPES.
- b) Subclause 5.44, “ROLE\_TABLE\_METHOD\_GRANTS view”:
  - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.
- c) Subclause 5.62, “TABLE\_METHOD\_PRIVILEGES view”:
  - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVILEGES.
- d) Subclause 5.82, “Short name views”:
  - i) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TABLE\_METHOD\_PRIVS.

- ii) Without Feature S024, “Enhanced structured types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROL\_TAB\_METH\_GRNTS.

13) Specifications for Feature S041, “Basic reference types”:

a) Subclause 5.39, “REFERENCED\_TYPES view”:

- i) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES.

b) Subclause 5.82, “Short name views”:

- i) Without Feature S041, “Basic reference types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.REFERENCED\_TYPES\_S.

14) Specifications for Feature S081, “Subtables”:

a) Subclause 5.26, “DIRECT\_SUPERTABLES view”:

- i) Without Feature S081, “Subtables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.DIRECT\_SUPERTABLES.

15) Specifications for Feature S091, “Basic array support”:

a) Subclause 5.30, “ELEMENT\_TYPES view”:

- i) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES.

b) Subclause 5.82, “Short name views”:

- i) Without Feature S091, “Basic array support” or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES\_S.

16) Specifications for Feature S241, “Transform functions”:

a) Subclause 5.65, “TRANSFORMS view”:

- i) Without Feature S241, “Transform functions”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRANSFORMS.

17) Specifications for Feature S271, “Basic multiset support”:

a) Subclause 5.30, “ELEMENT\_TYPES view”:

- i) Without Feature S091, “Basic array support”, or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES.

b) Subclause 5.82, “Short name views”:

- i) Without Feature S091, “Basic array support” or Feature S271, “Basic multiset support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ELEMENT\_TYPES\_S.

18) Specifications for Feature S401, “Distinct types based on array types”:

a) Subclause 5.76, “USER\_DEFINED\_TYPES view”:

- i) Without Feature S401, “Distinct types based on array types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.USER\_DEFINED\_TYPES.MAXIMUM\_CARDINALITY.
- b) Subclause 5.82, “Short name views”:
  - i) Without Feature S401, “Distinct types based on array types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.UDT\_S.MAX\_CARDINALITY.

19) Specifications for Feature T011, “Timestamp in Information Schema”:

- a) Subclause 5.6, “TIME\_STAMP domain”:
  - i) Without Feature F251, “Domain support”, and Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TIME\_STAMP.
- b) Subclause 5.36, “METHOD\_SPECIFICATIONS view”:
  - i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPECIFICATIONS.CREATED.
- c) Subclause 5.53, “ROUTINES view”:
  - i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.CREATED.
  - ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES.LAST\_ALTERED.
- d) Subclause 5.73, “TRIGGERS view”:
  - i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.TRIGGER\_CREATED.
- e) Subclause 5.82, “Short name views”:
  - i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.METHOD\_SPEC.CREATED.
  - ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.CREATED.
  - iii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.LAST\_ALTERED.
  - iv) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS\_S.CREATED.

20) Specifications for Feature T051, “Row types”:

- a) Subclause 5.32, “FIELDS view”:
  - i) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS.
- b) Subclause 5.82, “Short name views”:



- i) Without Feature T051, “Row types”, conforming SQL language shall not reference INFORMATION\_SCHEMA.FIELDS\_S.

21) Specifications for Feature T111, “Updatable joins, unions, and columns”:

a) Subclause 5.21, “COLUMNS view”:

- i) Without Feature T111, “Updatable joins, unions, and columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.IS\_UPDATABLE.

b) Subclause 5.82, “Short name views”:

- i) Without Feature T111, “Updatable joins, unions, and columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.IS\_UPDATABLE.

22) Specifications for Feature T175, “Generated columns”:

a) Subclause 5.17, “COLUMN\_COLUMN\_USAGE view”:

- i) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMN\_COLUMN\_USAGE.

b) Subclause 5.21, “COLUMNS view”:

- i) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.IS\_GENERATED.
- ii) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS.GENERATION\_EXPRESSION.

c) Subclause 5.82, “Short name views”:

- i) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COL\_COL\_USAGE.
- ii) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.IS\_GENERATED
- iii) Without Feature T175, “Generated columns”, conforming SQL language shall not reference INFORMATION\_SCHEMA.COLUMNS\_S.GENERATION\_EXPR.

23) Specifications for Feature T176, “Sequence generator support”:

a) Subclause 5.51, “ROUTINE\_SEQUENCE\_USAGE view”:

- i) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_SEQUENCE\_USAGE.

b) Subclause 5.55, “SEQUENCES view”:

- i) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES.

c) Subclause 5.71, “TRIGGER\_SEQUENCE\_USAGE view”:

- i) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.

d) Subclause 5.82, “Short name views”:



- i) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUT\_SEQ\_USAGE\_S.
- ii) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.SEQUENCES\_S.
- iii) Without Feature T176, “Sequence generator support”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQ\_USAGE\_S.

24) Specifications for Feature T180, “System-versioned tables”:

- a) Subclause 5.21, “COLUMNS view”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference any of the columns IS\_SYSTEM\_TIME\_PERIOD\_START, IS\_SYSTEM\_TIME\_PERIOD\_END, and SYSTEM\_TIME\_PERIOD\_TIMESTAMP\_GENERATION.
- b) Subclause 5.38, “PERIODS view”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PERIODS.
- c) Subclause 5.48, “ROUTINE\_PERIOD\_USAGE view”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.
- d) Subclause 5.69, “TRIGGER\_PERIOD\_USAGE view”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.
- e) Subclause 5.78, “VIEW\_PERIOD\_USAGE view”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.
- f) Subclause 5.82, “Short name views”:
  - i) Without Feature T180, “System-versioned tables”, conforming SQL language shall not reference any of the columns COLUMNS\_S.IS\_SYSPER\_START, COLUMNS\_S.IS\_SYSPER\_END, and COLUMNS\_S.SYSPER\_TSTMP\_GEN.

25) Specifications for Feature T181, “Application-time period tables”:

- a) Subclause 5.23, “CONSTRAINT\_PERIOD\_USAGE view”:
  - i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.CONSTRAINT\_PERIOD\_USAGE.
- b) Subclause 5.34, “KEY\_PERIOD\_USAGE view”:
  - i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.KEY\_PERIOD\_USAGE.
- c) Subclause 5.38, “PERIODS view”:
  - i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PERIODS.

d) Subclause 5.48, “ROUTINE\_PERIOD\_USAGE view”:

- i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINE\_PERIOD\_USAGE.

e) Subclause 5.69, “TRIGGER\_PERIOD\_USAGE view”:

- i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_PERIOD\_USAGE.

f) Subclause 5.78, “VIEW\_PERIOD\_USAGE view”:

- i) Without Feature T181, “Application-time period tables”, conforming SQL language shall not reference INFORMATION\_SCHEMA.VIEW\_PERIOD\_USAGE.

26) Specifications for Feature T211, “Basic trigger capability”:

a) Subclause 5.67, “TRIGGERED\_UPDATE\_COLUMNS view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERED\_UPDATE\_COLUMNS.

b) Subclause 5.68, “TRIGGER\_COLUMN\_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_COLUMN\_USAGE.

c) Subclause 5.70, “TRIGGER\_ROUTINE\_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_ROUTINE\_USAGE.

d) Subclause 5.71, “TRIGGER\_SEQUENCE\_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_SEQUENCE\_USAGE.

e) Subclause 5.72, “TRIGGER\_TABLE\_USAGE view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGER\_TABLE\_USAGE.

f) Subclause 5.73, “TRIGGERS view”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS.

g) Subclause 5.82, “Short name views”:

- i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_UPDATE\_COLS
- ii) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference the INFORMATION\_SCHEMA.TRIG\_TABLE\_USAGE view.
- iii) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_ROUT\_USAGE\_S.

- iv) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_SEQ\_USAGE\_S.
- v) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIGGERS\_S.
- vi) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not reference INFORMATION\_SCHEMA.TRIG\_COLUMN\_USAGE.

27) Specifications for Feature T213, “INSTEAD OF triggers”:

- a) Subclause 5.81, “VIEWS view”:
  - i) Without Feature T213, “INSTEAD OF triggers”, conforming SQL language shall not reference any of the columns IS\_TRIGGER\_UPDATABLE, IS\_TRIGGER\_DELETABLE, IS\_TRIGGER\_INSERTABLE\_INTO.
- b) Subclause 5.82, “Short name views”:
  - i) Without Feature T213, “INSTEAD OF triggers”, conforming SQL language shall not reference any of VIEWS\_S.IS\_TRIG\_UPDATABLE, VIEWS\_S.IS\_TRIG\_DELETABLE, VIEWS\_S.IS\_TRIG\_INS\_INTO.

28) Specifications for Feature T272, “Enhanced savepoint management”:

- a) Subclause 5.53, “ROUTINES view”:
  - i) Without Feature T272, “Enhanced savepoint management”, conforming SQL-language shall not reference INFORMATION\_SCHEMA.ROUTINES.NEW\_SAVEPOINT\_LEVEL.
- b) Subclause 5.82, “Short name views”:
  - i) Without Feature T272, “Enhanced savepoint management”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROUTINES\_S.NEW\_SAVEPOINT\_LEVEL.

29) Specifications for Feature T322, “Declared data type attributes”:

- a) Subclause 5.11, “ATTRIBUTES view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.
- b) Subclause 5.21, “COLUMNS view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION, and DECLARED\_NUMERIC\_SCALE.
- c) Subclause 5.29, “DOMAINS view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns DECLARED\_DATA\_TYPE, DECLARED\_NUMERIC\_PRECISION and DECLARED\_NUMERIC\_SCALE.
- d) Subclause 5.30, “ELEMENT\_TYPES view”:

- i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- e) Subclause 5.32, “`FIELDS` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- f) Subclause 5.35, “`METHOD_SPECIFICATION_PARAMETERS` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- g) Subclause 5.36, “`METHOD_SPECIFICATIONS` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- h) Subclause 5.37, “`PARAMETERS` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- i) Subclause 5.39, “`REFERENCED_TYPES` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- j) Subclause 5.53, “`ROUTINES` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL-language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION`, `DECLARED_NUMERIC_SCALE`, `RESULT_CAST_FROM_DECLARED_DATA_TYPE`, `RESULT_CAST_DECLARED_NUMERIC_PRECISION` and `RESULT_CAST_DECLARED_NUMERIC_SCALE`.
- k) Subclause 5.55, “`SEQUENCES` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- l) Subclause 5.76, “`USER_DEFINED_TYPES` view”:
  - i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of the columns `DECLARED_DATA_TYPE`, `DECLARED_NUMERIC_PRECISION` and `DECLARED_NUMERIC_SCALE`.
- m) Subclause 5.82, “Short name views”:

- i) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ATTRIBUTES\_S.DECLARED\_DATA\_TYPE, ATTRIBUTES\_S.DEC\_NUMERIC\_PREC and ATTRIBUTES\_S.DEC\_NUM\_SCALE.
- ii) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of COLUMNS\_S.DECLARED\_DATA\_TYPE, COLUMNS\_S.DEC\_NUMERIC\_PREC and COLUMNS\_S.DEC\_NUM\_SCALE.
- iii) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of DOMAINS\_S.DECLARED\_DATA\_TYPE, DOMAINS\_S.DEC\_NUMERIC\_PREC and DOMAINS\_S.DEC\_NUM\_SCALE.
- iv) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ELEMENT\_TYPES\_S.DECLARED\_DATA\_TYPE, ELEMENT\_TYPES\_S.DEC\_NUMERIC\_PREC and ELEMENT\_TYPES\_S.DEC\_NUM\_SCALE.
- v) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of FIELDS\_S.DECLARED\_DATA\_TYPE, FIELDS\_S.DEC\_NUMERIC\_PREC and FIELDS\_S.DEC\_NUM\_SCALE.
- vi) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of METHOD\_SPECS.DECLARED\_DATA\_TYPE, METHOD\_SPECS.DEC\_NUMERIC\_PREC and METHOD\_SPECS.DEC\_NUM\_SCALE.
- vii) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of METHOD\_SPEC\_PARAMS.DECLARED\_DATA\_TYPE, METHOD\_SPEC\_PARAMS.DEC\_NUMERIC\_PREC and METHOD\_SPEC\_PARAMS.DEC\_NUM\_SCALE.
- viii) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of PARAMETERS\_S.DECLARED\_DATA\_TYPE, PARAMETERS\_S.DEC\_NUMERIC\_PREC and PARAMETERS\_S.DEC\_NUM\_SCALE.
- ix) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of REFERENCED\_TYPES\_S.DECLARED\_DATA\_TYPE, REFERENCED\_TYPES\_S.DEC\_NUMERIC\_PREC and REFERENCED\_TYPES\_S.DEC\_NUM\_SCALE.
- x) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of ROUTINES\_S.DECLARED\_DATA\_TYPE, ROUTINES\_S.DEC\_NUMERIC\_PREC and ROUTINES\_S.DEC\_NUM\_SCALE.
- xi) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of SEQUENCES\_S.DECLARED\_DATA\_TYPE, SEQUENCES\_S.DEC\_NUMERIC\_PREC and SEQUENCES\_S.DEC\_NUM\_SCALE.
- xii) Without Feature T322, “Declared data type attributes”, conforming SQL language shall not reference any of UDT\_S.DECLARED\_DATA\_TYPE, UDT\_S.DEC\_NUMERIC\_PREC and UDT\_S.DEC\_NUM\_SCALE.

30) Specifications for Feature T331, “Basic roles”:

- a) [Subclause 5.8, “ADMINISTRABLE\\_ROLE\\_AUTHORIZATIONS view”](#):
  - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ADMINISTRABLE\_ROLE\_AUTHORIZATIONS.

- b) Subclause 5.9, “APPLICABLE\_ROLES view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.APPLICABLE\_ROLES.
  - c) Subclause 5.31, “ENABLED\_ROLES view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ENABLED\_ROLES.
  - d) Subclause 5.41, “ROLE\_COLUMN\_GRANTS view”:
    - i) Without Feature F231, “Privilege tables”, and Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_COLUMN\_GRANTS.
  - e) Subclause 5.42, “ROLE\_ROUTINE\_GRANTS view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUTINE\_GRANTS.
  - f) Subclause 5.43, “ROLE\_TABLE\_GRANTS view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_GRANTS.
  - g) Subclause 5.44, “ROLE\_TABLE\_METHOD\_GRANTS view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TABLE\_METHOD\_GRANTS.
  - h) Subclause 5.45, “ROLE\_USAGE\_GRANTS view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_USAGE\_GRANTS.
  - i) Subclause 5.46, “ROLE\_UDT\_GRANTS view”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_UDT\_GRANTS.
  - j) Subclause 5.82, “Short name views”:
    - i) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ADMIN\_ROLE\_AUTHS.
    - ii) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_ROUT\_GRANTS.
    - iii) Without Feature T331, “Basic roles”, conforming SQL language shall not reference INFORMATION\_SCHEMA.ROLE\_TAB\_METH\_GRNTS.
- 31) Specifications for Feature T522, “Default values for IN parameters of SQL-invoked procedures”:
- a) Subclause 5.37, “PARAMETERS view”:
    - i) Without Feature T522, “Default values for IN parameters of SQL-invoked procedures”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS.PARAMETER\_DEFAULT.

b) Subclause 5.82, “Short name views”:

- i) Without Feature T522, “Default values for IN parameters of SQL-invoked procedures”, conforming SQL language shall not reference INFORMATION\_SCHEMA.PARAMETERS\_S.PARAMETER\_DEFAULT.

*(Blank page)*



## **Annex B**

(informative)

### **Implementation-defined elements**

*This Annex modifies Annex B, “Implementation-defined elements”, in ISO/IEC 9075-2.*

This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-defined.

*None.*

*(Blank page)*

## **Annex C**

(informative)

### **Implementation-dependent elements**

*This Annex modifies Annex C, “Implementation-dependent elements”, in ISO/IEC 9075-2.*

This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-dependent.

*None.*

*(Blank page)*

## **Annex D** (informative)

### **Deprecated features**

*This Annex modifies Annex D, “Deprecated features”, in ISO/IEC 9075-2.*

It is intended that the following features will be removed at a later date from a revised version of this part of ISO/IEC 9075:

*None.*

*(Blank page)*

## **Annex E**

(informative)

### **Incompatibilities with ISO/IEC 9075:2008**

*This Annex modifies Annex E, “Incompatibilities with ISO/IEC 9075:2008”, in ISO/IEC 9075-2.*

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075-11:2008.

Except as specified in this Annex, features and capabilities of Database Language SQL are compatible with ISO/IEC 9075-11:2008.

- 1) The SQL\_SIZING\_PROFILES base table, the SQL\_SIZING\_PROFS view and the SQL\_SIZING\_PROFILES view have been removed from this edition of ISO/IEC 9075-11 because the support of SQL Profiles is entirely removed.
- 2) In ISO/IEC 9075-11:2008, the INFORMATION\_SCHEMA contained the view COLLATIONS\_S. This view has been removed, because it was deprecated in ISO/IEC 9075:2008.
- 3) In ISO/IEC 9075-11:2008, the INFORMATION\_SCHEMA views METHOD\_SPECIFICATIONS and METHOD\_SPECS contained the column LAST\_ALTERED. This column is removed, because it was deprecated in ISO/IEC 9075:2008.
- 4) In ISO/IEC 9075-2:2008, ISO/IEC 9075-4:2008, and ISO/IEC 9075-11:2008, Feature Codes T322 and T332 were not used consistently; they identified three different Features in various parts of ISO/IEC 9075. In this edition of ISO/IEC 9075, three Feature Codes (T322, T332, and T341) have been used consistently to identify those three Features.

*(Blank page)*



## Annex F (informative)

### SQL feature taxonomy

*This Annex modifies Annex F, “SQL feature taxonomy”, in ISO/IEC 9075-2.*

This Annex describes a taxonomy of features defined in this part of ISO/IEC 9075.

Table 2, “Feature taxonomy and definition for mandatory features”, contains a taxonomy of the features of the SQL language in Core SQL that are specified in this part of ISO/IEC 9075. Table 3, “Feature taxonomy for optional features”, contains a taxonomy of the features of the SQL language not in Core SQL that are specified in this part of ISO/IEC 9075.

Table 39, “Feature taxonomy for optional features”, in [ISO9075-2] contains a taxonomy of the features of the SQL language not in Core SQL that are specified in Table 3, “Feature taxonomy for optional features”, of this part of ISO/IEC 9075 and in [ISO9075-2].

In these tables, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The “Feature ID” column of Table 2, “Feature taxonomy and definition for mandatory features”, and of Table 3, “Feature taxonomy for optional features”, specifies the formal identification of each feature and each subfeature contained in the table.

The “Feature Name” column of Table 2, “Feature taxonomy and definition for mandatory features”, and or Table 3, “Feature taxonomy for optional features”, contains a brief description of the feature or subfeature associated with the Feature ID value.

The “Feature Description” column of Table 2, “Feature taxonomy and definition for mandatory features”, provides the only definition of the mandatory features of this part of ISO/IEC 9075. This definition consists of indications of specific language elements supported in each feature, subject to the constraints of all Syntax Rules, Access Rules, and Conformance Rules.

Table 3, “Feature taxonomy for optional features”, does not provide definitions of the features; the definition of those features is found in the Conformance Rules that are further summarized in Annex A, “SQL Conformance Summary”.

Table 2 — Feature taxonomy and definition for mandatory features

	Feature ID	Feature Name	Feature Description
1	<b>F021</b>	<b>Basic information schema</b>	— Subclause 5.1, “ <b>INFORMATION_SCHEMA Schema</b> ”: (Support of the COLUMNS, TABLES, VIEWS, TABLE_CONSTRAINTS, REFERENTIAL_CONSTRAINTS, and CHECK_CONSTRAINTS views in the INFORMATION_SCHEMA)
2	F021-01	COLUMNS view	— Subclause 5.21, “COLUMNS view”
3	F021-02	TABLES view	— Subclause 5.64, “TABLES view”
4	F021-03	VIEWS view	— Subclause 5.81, “VIEWS view”
5	F021-04	TABLE_CONSTRAINTS view	— Subclause 5.61, “TABLE_CONSTRAINTS view”
6	F021-05	REFERENTIAL_CONSTRAINTS view	— Subclause 5.40, “REFERENTIAL_CONSTRAINTS view”
7	F021-06	CHECK_CONSTRAINTS view	— Subclause 5.14, “CHECK_CONSTRAINTS view”
8	<b>F501</b>	<b>Features and conformance views</b>	— Clause 5, “Information Schema”: SQL_FEATURES, SQL_SIZING, and SQL_LANGUAGE views
9	F501-01	SQL_FEATURES view	— Subclause 5.56, “SQL_FEATURES view”
10	F501-02	SQL_SIZING view	— Subclause 5.60, “SQL_SIZING view”
11	<b>S011</b>	<b>Distinct data types</b>	— Subclause 11.51, “<user-defined type definition>”: When <representation> is <predefined type>
12	S011-01	USER_DEFINED_TYPES view	— Subclause 5.76, “USER_DEFINED_TYPES view”

	Feature ID	Feature Name	Feature Description
13	T321	Basic SQL-invoked routines	<p>— Subclause 11.60, “&lt;SQL-invoked routine&gt;” — If Feature T041, “Basic LOB data type support”, is supported, then the &lt;locator indication&gt; clause shall also be supported</p> <p>NOTE 15 — “Routine” is the collective term for functions, methods, and procedures. This feature requires a conforming SQL-implementation to support both user-defined functions and user-defined procedures. An SQL-implementation that conforms to Core SQL shall support at least one language for writing routines; that language may be SQL. If the language is SQL, then the basic specification capability in Core SQL is the ability to specify a one-statement routine. Support for overloaded functions and procedures is not part of Core SQL.</p>
14	T321-06	ROUTINES view	— Subclause 5.53, “ROUTINES view”
15	T321-07	PARAMETERS view	— Subclause 5.37, “PARAMETERS view”

Table 3 — Feature taxonomy for optional features

	Feature ID	Feature Name
1	F231	Privilege tables
2	F341	Usage tables
3	F502	Enhanced documentation tables
4	F696	Additional translation documentation
5	T011	Timestamp in Information Schema
6	T322	Declared data type attributes

*(Blank page)*

## **Annex G**

(informative)

### **Defect reports not addressed in this edition of this part of ISO/IEC 9075**

Each entry in this Annex describes a reported defect in the previous edition of this part of ISO/IEC 9075 that remains in this edition.

*None.*

*(Blank page)*

## Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

### — A —

ADMIN • 16, 200  
 ALL • 132, 135, 153, 177, 179, 183, 190  
 AND • 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 41, 42, 43, 44, 51, 54, 57, 59, 60, 61, 62, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 130, 131, 154, 155, 159, 160, 161, 162, 163, 164, 180, 186, 209, 210, 232, 253  
 ANY • 164, 201, 207, 246  
 AS • 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 35, 37, 39, 41, 42, 43, 44, 46, 48, 49, 51, 53, 54, 56, 57, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 130, 132, 168, 170, 171  
 ASSERTION • 130, 131, 132  
 ATTRIBUTES • 19, 39, 111, 135  
 AUTHORIZATION • 9, 129

### — B —

BY • 130, 132, 135, 153, 177, 179, 180, 183, 190

### — C —

CASE • 59, 61, 77, 99, 109  
 CHARACTER • 12, 13, 15  
 CHARACTER\_LENGTH • 155  
 CHECK • 7, 10, 11, 15, 130, 131, 132, 133, 135, 137, 141, 143, 144, 145, 146, 147, 148, 149, 151, 153, 154, 158, 164, 168, 170, 172, 174, 175, 177, 179, 180, 181, 183, 185, 186, 190, 191, 194, 196, 197, 199, 201, 202, 203, 205, 206, 207, 208, 209, 210, 215, 217, 219, 221, 223, 224, 226, 228, 231, 232, 235, 238, 239, 240, 241, 242, 243, 244, 245, 246, 248, 249, 250, 252, 253, 256, 257, 258, 259, 260

COALESCE • 30, 31

CONSTRAINT • 7, 10, 11, 15, 133, 135, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 151, 153, 154, 158, 164, 168, 170, 172, 174, 175, 177, 179, 180, 181, 183, 185, 186, 190, 191, 193, 194, 196, 197, 199, 201, 202, 203, 205, 206, 207, 208, 209, 210, 215, 217, 219, 221, 222, 223, 224, 226, 228, 231, 232, 235, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 248, 249, 250, 252, 253, 256, 257, 258, 259, 260

COUNT • 10, 51, 130, 132, 135, 153, 177, 179, 180, 183, 190

CREATE • 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 33, 35, 37, 39, 41, 42, 43, 44, 46, 48, 49, 51, 53, 54, 56, 59, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 129, 130, 131, 132, 133, 135, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 151, 153, 158, 168, 170, 172, 174, 175, 177, 179, 181, 183, 185, 190, 193, 194, 196, 199, 201, 202, 203, 205, 206, 207, 208, 215, 217, 219, 221, 222, 223, 226, 228, 231, 235, 237, 239, 240, 241, 242, 243, 244, 245, 248, 250, 252, 256, 257, 258, 259, 260

CURRENT\_ROLE • 48

CURRENT\_TIMESTAMP • 136, 174, 189, 214, 247

CURRENT\_USER • 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 41, 42, 43, 44, 51, 54, 57, 60, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 136, 174

### — D —

DISTINCT • 19, 30, 43, 44, 46, 49, 63, 64, 130

DOMAIN • 11, 12, 13, 14, 15

### — E —

ELSE • 59, 61, 77, 99, 109

## IWD 9075-11:2011(E)

END • 59, 61, 77, 99, 109

EXCEPT • 232

EXISTS • 59, 61, 77, 99, 109, 130, 131, 168, 170, 180, 232

### — F —

Feature F231, “Privilege tables” • 28, 40, 65, 66, 67, 70, 73, 89, 101, 102, 125, 265, 266, 284

Feature F251, “Domain support” • 11, 12, 13, 14, 15, 27, 43, 45, 125, 266, 267, 277

Feature F341, “Usage tables” • 22, 26, 27, 29, 34, 36, 38, 52, 53, 69, 71, 72, 74, 75, 76, 94, 95, 96, 97, 98, 105, 106, 107, 108, 125, 126, 267, 268, 269

Feature F391, “Long identifiers” • 8, 10, 16, 20, 21, 22, 24, 25, 26, 27, 31, 34, 36, 38, 45, 46, 49, 52, 53, 55, 57, 60, 63, 64, 66, 68, 71, 72, 74, 75, 76, 79, 80, 81, 83, 88, 90, 92, 93, 94, 95, 96, 97, 98, 100, 104, 106, 110, 269, 270, 271, 272, 273

Feature F502, “Enhanced documentation tables” • 83, 84, 85, 126, 273, 274

Feature F521, “Assertions” • 18, 274

Feature F651, “Catalog name qualifiers” • 10, 274

Feature F690, “Collation support” • 24, 25, 126, 274

Feature F695, “Translation support” • 92, 126, 274

Feature F696, “Additional translation documentation” • 92, 126, 274, 275

FOREIGN • 10, 133, 139, 140, 141, 143, 144, 146, 147, 148, 149, 150, 151, 154, 164, 168, 170, 172, 174, 175, 177, 179, 181, 183, 186, 191, 193, 194, 196, 201, 202, 203, 205, 206, 207, 209, 215, 217, 226, 228, 231, 232, 235, 237, 239, 240, 241, 242, 243, 244, 245, 250, 252, 253, 256, 257, 258, 259

FROM • 7, 10, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 39, 41, 42, 43, 44, 46, 48, 49, 51, 53, 54, 55, 57, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 130, 131, 132, 135, 141, 143, 144, 145, 146, 147, 149, 151, 153, 154, 155, 164, 168, 169, 170, 171, 174, 175, 177, 179, 180, 181, 183, 186, 190, 191, 194, 197, 199, 201, 202, 203, 205, 206, 207, 210, 215, 217, 223, 224, 226, 228, 231, 232, 238, 239, 240, 241, 242, 243, 244, 246, 248, 249, 250, 253, 256, 257, 258, 259, 260

FULL • 131, 186, 209, 232

### — G —

GRANT • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80,

81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125

GROUP • 130, 132, 135, 153, 177, 179, 180, 183, 190

### — H —

HAVING • 180

### — I —

IN • 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 41, 42, 43, 44, 46, 49, 51, 54, 57, 59, 60, 61, 62, 63, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 131, 133, 135, 137, 141, 143, 144, 145, 146, 147, 148, 149, 151, 153, 154, 155, 158, 159, 160, 161, 164, 168, 170, 172, 174, 175, 177, 180, 181, 183, 185, 186, 190, 191, 194, 196, 197, 199, 201, 202, 203, 205, 206, 207, 208, 209, 210, 215, 217, 219, 223, 224, 226, 228, 231, 232, 235, 238, 239, 240, 241, 242, 243, 244, 245, 246, 248, 249, 250, 252, 253, 256, 257, 258, 259, 260

INCREMENT • 218

INTEGER • 11

INTERSECT • 169, 171

IS • 27, 39, 60, 78, 131, 154, 155, 159, 160, 161, 162, 163, 164, 180, 186, 209, 210, 219, 221, 232, 253

### — J —

JOIN • 17, 18, 19, 22, 23, 25, 26, 27, 29, 31, 33, 35, 37, 41, 42, 44, 46, 48, 49, 51, 53, 54, 57, 59, 63, 64, 66, 71, 72, 73, 74, 75, 76, 78, 81, 94, 95, 96, 97, 98, 103, 105, 106, 107, 108, 109, 131, 168, 171

### — K —

KEY • 10, 133, 135, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 151, 154, 164, 168, 170, 172, 174, 175, 177, 179, 181, 183, 186, 190, 191, 193, 194, 196, 199, 201, 202, 203, 205, 206, 207, 209, 215, 217, 219, 221, 222, 223, 226, 228, 231, 232, 235, 237, 239, 240, 241, 242, 243, 244, 245, 248, 250, 252, 253, 256, 257, 258, 259, 260

### — L —

LANGUAGE • 188

LEFT • 19, 31, 57, 59, 64, 78, 103

### — M —

MATCH • 186, 209, 232

MAX • 51, 132, 135, 153, 177, 179, 180, 183, 190

### — N —



NO • 156

NOT • 7, 27, 39, 130, 131, 133, 135, 137, 140, 141, 143, 144, 146, 147, 148, 149, 151, 153, 154, 158, 159, 160, 161, 162, 163, 164, 168, 170, 172, 174, 177, 179, 180, 181, 183, 185, 186, 190, 193, 196, 197, 199, 202, 203, 205, 206, 208, 209, 210, 215, 217, 219, 221, 222, 223, 226, 228, 231, 232, 235, 237, 238, 239, 240, 241, 242, 243, 244, 245, 248, 250, 252, 253, 256, 257, 258, 259, 260

NULL • 27, 39, 59, 60, 61, 77, 78, 99, 109, 131, 133, 135, 137, 140, 141, 148, 151, 153, 154, 155, 158, 159, 160, 161, 162, 163, 164, 172, 174, 177, 179, 180, 181, 183, 185, 186, 190, 193, 196, 199, 203, 208, 209, 210, 215, 217, 219, 221, 222, 223, 226, 228, 231, 232, 235, 237, 239, 245, 248, 250, 252, 253, 260

## — O —

ON • 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35, 36, 37, 39, 41, 42, 43, 44, 46, 48, 49, 52, 53, 54, 55, 57, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 168, 171

OPTION • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 38, 40, 41, 42, 43, 45, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 200

OR • 7, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 41, 42, 43, 44, 51, 54, 57, 59, 60, 61, 62, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 141, 143, 144, 146, 147, 149, 151, 154, 159, 160, 161, 162, 163, 164, 180, 181, 186, 197, 199, 201, 202, 203, 205, 206, 207, 209, 210, 215, 219, 221, 223, 226, 228, 232, 238, 240, 241, 242, 243, 244, 246, 249, 250, 253, 256, 257, 258, 259

OUTER • 131

## — P —

PARAMETER\_MODE • 184, 191

PRIMARY • 10, 133, 135, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 151, 154, 164, 168, 170, 172, 174, 175, 177, 179, 181, 183, 186, 190, 193, 194, 196, 199, 201, 202, 203, 205, 206, 207, 209, 215, 217, 219, 221, 222, 223, 226, 228, 231, 235, 237, 239, 240, 241, 242, 243, 244, 245, 248, 250, 252, 256, 257, 258, 259, 260

PUBLIC • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 38, 40, 41, 42, 43, 45, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 152, 199, 204, 227, 229, 249, 251

## — R —

RECURSIVE • 17, 48, 168, 170

REF • 255

REFERENCES • 10, 133, 139, 140, 141, 143, 144, 146, 147, 148, 149, 150, 151, 154, 164, 168, 170, 172, 174, 175, 177, 178, 180, 181, 183, 186, 191, 193, 194, 201, 202, 203, 205, 206, 207, 209, 215, 217, 226, 228, 231, 232, 235, 237, 239, 240, 241, 242, 243, 244, 245, 250, 252, 253, 256, 257, 258, 259

## — S —

Feature S023, “Basic structured types” • 20, 55, 57, 126, 275

Feature S024, “Enhanced structured types” • 42, 68, 88, 126, 275, 276

Feature S041, “Basic reference types” • 63, 126, 276

Feature S081, “Subtables” • 41, 276

Feature S091, “Basic array support” • 46, 126, 276

Feature S241, “Transform functions” • 91, 276

Feature S271, “Basic multiset support” • 46, 126, 276

Feature S401, “Distinct types based on array types” • 104, 126, 276, 277

SCHEMA • 9, 129

SELECT • 7, 8, 10, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 39, 41, 42, 43, 44, 46, 48, 49, 51, 52, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 130, 131, 132, 135, 141, 143, 144, 145, 146, 147, 149, 151, 153, 154, 155, 164, 168, 169, 170, 171, 174, 175, 177, 179, 180, 181, 183, 186, 190, 191, 194, 197, 199, 201, 202, 203, 205, 206, 207, 210, 215, 217, 223, 224, 226, 228, 231, 232, 238, 239, 240, 241, 242, 243, 244, 246, 248, 249, 250, 253, 256, 257, 258, 259, 260

SET • 12, 13, 15

SQL • 188, 211, 212, 236

## — T —

Feature T011, “Timestamp in Information Schema” • 14, 57, 79, 100, 126, 127, 189, 214, 247, 266, 277

Feature T051, “Row types” • 49, 127, 277, 278

## IWD 9075-11:2011(E)

Feature T111, “Updatable joins, unions, and columns” • 32, 127, 233, 278  
Feature T175, “Generated columns” • 26, 32, 127, 278  
Feature T176, “Sequence generator support” • 75, 81, 97, 127, 278, 279  
Feature T180, “System-versioned tables” • 32, 62, 72, 95, 106, 127, 279  
Feature T181, “Application-time period tables” • 36, 53, 62, 72, 95, 106, 279, 280  
Feature T211, “Basic trigger capability” • 93, 94, 96, 97, 98, 100, 127, 280, 281  
Feature T213, “INSTEAD OF triggers” • 109, 127, 281  
Feature T272, “Enhanced savepoint management” • 79, 127, 213, 281  
Feature T322, “Declared data type attributes” • 20, 32, 45, 47, 50, 55, 58, 60, 63, 79, 81, 104, 128, 281, 282, 283  
Feature T331, “Basic roles” • 16, 17, 48, 65, 66, 67, 68, 69, 70, 127, 128, 265, 283, 284  
Feature T522, “Default values for IN parameters of SQL-invoked procedures” • 60, 128, 284, 285  
TABLE • 10, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 37, 39, 41, 42, 43, 44, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 133, 135, 137, 138, 139, 140, 141, 143, 144, 145, 146, 147, 148, 149, 150, 151, 153, 158, 168, 170, 172, 174, 175, 177, 179, 181, 183, 185, 190, 194, 196, 199, 201, 203, 205, 206, 207, 208, 215, 217, 219, 221, 222, 223, 226, 228, 231, 235, 237, 239, 240, 241, 242, 243, 244, 245, 248, 250, 252, 256, 257, 258, 259, 260  
THEN • 59, 61, 77, 99, 109  
TIME • 14  
TIMESTAMP • 14  
TO • 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 38, 40, 41, 42, 43, 45, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125  
TRANSFORMS • 91, 235, 276  
TYPE • 250

## — U —

UNION • 17, 33, 35, 37, 39, 48, 61, 87, 90, 100, 132, 145, 168, 171, 248  
UNIQUE • 135, 154, 177, 179, 181, 191, 222, 224  
UPDATE • 93, 239  
USAGE • 8, 11, 12, 13, 14, 15, 69, 102, 250

USING • 27, 33, 35, 46, 49, 51, 53, 54, 60, 63, 64, 66, 71, 72, 73, 76, 109, 131

## — V —

VALUE • 11, 15  
VALUES • 48, 133, 172, 223, 260  
VARYING • 12, 13, 15  
VIEW • 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 33, 35, 37, 39, 41, 42, 43, 44, 46, 48, 49, 51, 53, 54, 56, 59, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124

## — W —

WHEN • 59, 61, 77, 99, 109  
WHERE • 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 35, 37, 39, 41, 42, 43, 44, 46, 49, 51, 53, 54, 57, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 84, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 108, 109, 130, 131, 135, 169, 171, 175, 177, 180, 181, 194, 197, 199, 224, 232, 239, 253, 260  
WITH • 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 34, 36, 38, 40, 41, 42, 43, 45, 46, 48, 49, 52, 53, 55, 57, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 125, 168, 170, 200

## — Z —

ZONE • 14

# **Editor's Notes**

Some possible problem and language opportunities have been observed with the specifications contained in this document. Further contributions to this list are welcome. Deletions from the list (resulting from change proposals that correct the problems or from research indicating that the problems do not, in fact, exist) are even more welcome.

Because of the dynamic nature of this list (problems being removed because they are solved, new problems being added), each problem or opportunity has been assigned a "fixed" number. These numbers do not change from draft to draft.

## Possible Problems: Major Technical

**SCHEM-000** The following Possible Problem has been noted:

**Severity:** major technical

**Reference:**

**Note At:** None.

**Source:** Your humble Editor.

**Possible Problem:**

In the body of the Working Draft, there occasionally appears a point that requires particular attention, highlighted thus:

**\*\* Editor's Note (number 1) \*\***

Text of the problem.

**Solution:**

None provided with comment.

**SCHEM-034** The following Possible Problem has been noted:

**Severity:** major technical

**Reference:** P11, SQL/Schemata, No specific location

**Note At:** None.

**Source:** WG3:WLG-077

**Possible Problem:**

The tables CHARACTER\_ENCODING\_FORMS, CHARACTER\_REPERTOIRES, COLLATIONS, AUTHORIZATIONS and COLLATION\_CHARACTER\_SET\_APPLICABILITY have in the definitions of their respective descriptors in [FOUND CD], initial data assigned.

This is not reflected in the respective table definitions in [SCHEM CD].

The only exception, where this is done, is the definition of the CHARACTER\_SETS base table. In Description 6) to 8). But even that does not cover all predefined character sets in [FOUND CD].

**Solution:**

None provided with comment.

**SCHEM-035** The following Possible Problem has been noted:

**Severity:** major technical

**Reference:** P11, SQL/Schemata, [Subclause 6.10](#), “CHARACTER\_REPERTOIRES base table”

**Note At:** None.

**Source:** WG3:WLG-077

**Possible Problem:**

The table CHARACTER\_REPERTOIRE has a foreign key defined for a default collation. As CHARACTER\_REPERTOIRE has itself in its primary key only the name, it is not clear, which collation (Catalog and Schema) should here be referenced? Is there a special catalog or schema which holds these default collations?

This table was added by [ICN-054R2], and there it is also not explained, which catalog should be used.

**Solution:**

None provided with comment.

**SCHEM-036** The following Possible Problem has been noted:

**Severity:** major technical

**Reference:** P11, SQL/Schemata, [Subclause 6.17](#), “COLLATIONS base table”

**Note At:** None.

**Source:** WG3:WLG-077

**Possible Problem:**

The Collation Descriptor in [FOUND CD] Subclause 4.2.6 contains the Attribute "Name". This is a schema qualified name, and contains therefore catalog and schema name.

In the definition of the initial data it is not defined, to which catalog / schema this data belongs.

**Solution:**

None provided with comment.