



# INTERNATIONAL STANDARD ISO/IEC 9075-2:1999 TECHNICAL CORRIGENDUM 2

Published 2003-06-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION  
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

## Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)

### TECHNICAL CORRIGENDUM 2

*Technologies de l'information — Langues de base de données — SQL —*

*Partie 2: Fondations (SQL/Fondations)*

RECTIFICATIF TECHNIQUE 2

ITeH STANDARD PREVIEW  
(standards.iteh.ai)

ISO/IEC 9075-2:1999/Cor 2:2003

Technical Corrigendum 2 to ISO/IEC 9075-2:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 32, Data management and interchange. ISO/IEC 9075-2:1999/Cor. 2:2003 cancels and replaces ISO/IEC 9075-2:1999/Cor. 1:2000.

#### Statement of purpose for rationale:

A statement indicating the rationale for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases the reason is editorial or to clarify the wording; in some cases it is to correct an error or an omission in the original wording.

#### Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General and Conformance Rules, the new rules have been numbered as follows:

Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7) a.1), 7) a.2), etc.]. Those inserted before Rule 1) are numbered 0.1), 0.2), etc.

Where this Corrigendum introduces new Subclauses, the new subclauses have been numbered as follows:

Subclauses inserted between, for example, Subclause 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc.

Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060

Ref. No. ISO/IEC 9075-2:1999/Cor.2:2003(E)

© ISO/IEC 2003 – All rights reserved

Published in Switzerland

Contents

	Page
2 Normative references . . . . .	9
3.1.1 Definitions taken from ISO/IEC 10646 . . . . .	9
3.1.2 Definitions taken from Unicode . . . . .	9
3.1.5 Definitions provided in Part 2 . . . . .	10
3.3.1.2 Other terms . . . . .	12
4.1 Data types . . . . .	12
4.2 Character strings . . . . .	15
4.2.1 Character strings and collating sequences . . . . .	16
4.2.2.1 Operators that operate on character strings and return character strings . . . . .	17
4.2.4 Named character sets . . . . .	17
4.3 Binary strings . . . . .	21
4.3.1 Binary string comparison . . . . .	22
4.4.1 Bit string comparison and assignment . . . . .	22
4.5 Numbers . . . . .	22
4.5.1 Characteristics of numbers . . . . .	23
4.5.2 Operations involving numbers . . . . .	24
4.6.1 Comparison and assignment of booleans . . . . .	24
4.7.1 Datetimes . . . . .	24
4.7.2 Intervals . . . . .	25
4.7.3 Operations involving datetimes and intervals . . . . .	25
4.8 User-defined types . . . . .	26
4.8.1 Observers and mutators . . . . .	27
4.8.2 Constructors . . . . .	27
4.8.4 User-defined type comparison and assignment . . . . .	28
4.9 Row types . . . . .	28
4.10 Reference types . . . . .	28

**ITeH STANDARD PREVIEW**  
(standards.iteh.ai)  
<https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003>

4.11 Collection types	29
4.11.2 Collection comparison	30
4.12 Type conversions and mixing of data types	30
4.13 Data conversions	30
4.16 Tables	30
4.16.1 Types of tables	31
4.16.3 Operations involving tables	32
4.17 Integrity constraints	32
4.17.1 Checking of constraints	33
4.17.2 Table constraints	33
4.18.1 General rules and definitions	34
4.18.4 Known functional dependencies in a <joined table>	34
4.18.9 Known functional dependencies in the result of <having clause>	34
4.18.10 Known functional dependencies in a <query specification>	35
4.20 SQL-schemas	35
4.21 SQL-client modules	35
4.23 SQL-invoked routines	36
4.24 Built-in functions	39
4.25 SQL-paths	40
4.26.4 Locators	40
4.27 Diagnostics area	40
4.30.1 Classes of SQL-statements	41
4.30.2 SQL-statements classified by functions	42
4.30.3 SQL-statements and transaction states	42
4.30.4 SQL-statement atomicity	43
4.31.1 Authorization identifiers	44
4.32 SQL-transactions	44
4.34 SQL-sessions	45
4.34.1 Execution contexts	45
4.35 Triggers	46
4.35.2 Execution of triggers	46
4.36 Client-server operation	47
5.2 <token> and <separator>	47
5.3 <literal>	51
5.4 Names and identifiers	52
6.1 <data type>	53
6.3 <value specification> and <target specification>	56
6.5 <identifier chain>	58
6.6 <column reference>	60
6.8 <field reference>	62
6.11 <method invocation>	62
6.14 <dereference operation>	63
6.16 <set function specification>	63
6.17 <numeric value function>	65
6.18 <string value function>	66
6.19 <datetime value function>	72
6.20 <interval value function>	72
6.22 <cast specification>	73
6.23 <value expression>	77
6.24 <new specification>	79
6.25 <subtype treatment>	80
6.26 <numeric value expression>	81

6.27 <string value expression>	81
6.28 <datetime value expression>	83
6.29 <interval value expression>	84
6.30 <boolean value expression>	85
7.1 <row value constructor>	85
7.2 <row value expression>	86
7.3 <table value constructor>	87
7.6 <table reference>	87
7.7 <joined table>	93
7.8 <where clause>	94
7.9 <group by clause>	94
7.10 <having clause>	102
7.11 <query specification>	102
7.12 <query expression>	105
7.13 <search or cycle clause>	110
7.14 <subquery>	113
8.2 <comparison predicate>	113
8.3 <between predicate>	115
8.4 <in predicate>	115
8.5 <like predicate>	116
8.6 <similar predicate>	117
8.7 <null predicate>	119
8.8 <quantified comparison predicate>	119
8.10 <unique predicate>	120
8.11 <match predicate>	120
8.12 <overlaps predicate>	122
8.13 <distinct predicate>	123
8.14 <type predicate>	123
9.0 Determination of identical values	123
9.1 Retrieval assignment	125
9.2 Store assignment	126
9.3 Data types of results of aggregations	126
9.5 Type precedence list	127
10.3 <path specification>	127
10.4 <routine invocation>	127
10.5 <privileges>	136
10.6 <character set specification>	137
10.7 <specific routine designator>	138
10.8 <collate clause>	141
10.12 Execution of triggers	141
10.13 Execution of array-returning functions	141
10.14 Data type identity	143
11.1 <schema definition>	144
11.2 <drop schema statement>	145
11.3 <table definition>	146
11.4 <column definition>	149
11.5 <default clause>	150
11.7 <unique constraint definition>	150
11.8 <referential constraint definition>	151
11.9 <check constraint definition>	157
11.16 <drop column scope clause>	157
11.17 <drop column definition>	158

iTeh STANDARD PREVIEW  
(standards.iteh.ai)

ISO/IEC 9075-2:1999/Cor 2:2003

[https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-](https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003)

83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003

11.19 <drop table constraint definition>	158
11.20 <drop table statement>	159
11.21 <view definition>	159
11.22 <drop view statement>	161
11.23 <domain definition>	161
11.30 <character set definition>	162
11.31 <drop character set statement>	162
11.32 <collation definition>	163
11.33 <drop collation statement>	163
11.35 <translation definition>	163
11.36 <assertion definition>	164
11.37 <drop assertion statement>	164
11.38 <trigger definition>	166
11.40 <user-defined type definition>	166
11.41 <attribute definition>	177
11.43 <add attribute definition>	178
11.44 <drop attribute definition>	178
11.45 <add original method specification>	179
11.46 <add overriding method specification>	183
11.47 <drop method specification>	187
11.48 <drop data type statement>	190
11.49 <SQL-invoked routine>	191
11.50 <alter routine statement>	200
11.51 <drop routine statement>	201
11.52 <user-defined cast definition>	201
11.53 <drop user-defined cast statement>	203
11.54 <user-defined ordering definition>	203
11.55 <drop user-defined ordering statement>	205
11.56 <transform definition>	207
11.57 <drop transform statement>	207
12.2 <grant privilege statement>	208
12.4 <select statement: single row>	208
12.6 <revoke statement>	209
13.1 <SQL-client module definition>	211
13.2 <module name clause>	211
13.3 <externally-invoked procedure>	212
13.4 Calls to an <externally-invoked procedure>	212
13.5 <SQL procedure statement>	214
13.6 Data type correspondences	217
14.1 <declare cursor>	217
14.3 <fetch statement>	220
14.4 <close statement>	221
14.5 <select statement: single row>	221
14.6 <delete statement: positioned>	222
14.8 <insert statement>	223
14.9 <update statement: positioned>	224
14.10 <update statement: searched>	226
14.11 <temporary table declaration>	227
14.12 <free locator statement>	227
14.14 Effect of deleting rows from base table	227
14.17 Effect of inserting tables into base tables	228
14.18 Effect of inserting a table into a derived table	229

14.20 Effect of replacing rows from base table	229
15.2 <return statement>	229
16.4 <savepoint statement>	230
16.5 <release savepoint statement>	230
16.6 <commit statement>	230
16.7 <rollback statement>	230
19.1 <get diagnostics statement>	231
19.2 Pushing and popping the diagnostics area stack	234
20.4 CARDINAL_NUMBER domain	235
20.9 APPLICABLE_ROLES view	235
20.11 ATTRIBUTES view	235
20.12 CHARACTER_SETS view	236
20.12a CHECK_CONSTRAINT_ROUTINE_USAGE view	237
20.14 COLLATION view	237
20.16 COLUMN_PRIVILEGES view	238
20.17 COLUMN_UDT_USAGE view	238
20.18 COLUMNS view	239
20.19 CONSTRAINT_COLUMN_USAGE view	240
20.21 DATA_TYPE_PRIVILEGES view	241
20.23 DIRECT_SUPERTYPES view	242
20.25 DOMAIN_UDT_USAGE view	242
20.26 DOMAINS view	242
20.27 ELEMENT_TYPES view	243
20.29 FIELDS view	244
20.30 KEY_COLUMN_USAGE view	244
20.31 METHOD_SPECIFICATION_PARAMETERS view	245
20.32 METHOD_SPECIFICATIONS view	246
20.33 PARAMETERS view	247
20.34 REFERENCED_TYPES view	248
20.35 REFERENTIAL_CONSTRAINTS view	249
20.38 ROLE_TABLE_GRANTS view	249
20.43 ROUTINE_PRIVILEGES view	250
20.44a ROUTINE_ROUTINE_USAGE view	250
20.45 ROUTINES view	251
20.53 TABLE_CONSTRAINTS view	252
20.54 TABLE_METHOD_PRIVILEGES view	252
20.55 TABLE_PRIVILEGES view	253
20.56 TABLES view	253
20.57 TRANSFORMS view	254
20.58 TRANSLATIONS view	254
20.59 TRIGGERED_UPDATE_COLUMNS view	255
20.60a TRIGGER_ROUTINE_USAGE view	255
20.62 TRIGGERS view	256
20.63 USAGE_PRIVILEGES view	258
20.64 UDT_PRIVILEGES view	258
20.65 USER_DEFINED_TYPES view	258
20.66a VIEW_ROUTINE_USAGE view	259
20.62 TRIGGERS view	260
20.68 VIEWS view	260
20.69 Short name views	261
20.70 Definition of SQL built-in functions	266
21.3 EQUAL_KEY_DEGREES assertion	266

Full STANDARD PREVIEW

(standards.iteh.ai)

ISO/IEC 9075-2:1999/Cor.2:2003

[https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-](https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003)

[83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003](https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003)

21.6 ASSERTIONS base table	267
21.7 ATTRIBUTES base table	267
20.7a CHARACTER_ENCODING_FORMS base table	267
21.7b CHARACTER_REPERTOIRES base table	268
21.8 CHARACTER_SETS base table	269
21.9a CHECK_CONSTR AINT_ROUTINE_USAGE base table	270
21.12 COLLATIONS base table	271
21.14 COLUMNS base table	272
21.15 DATA_TYPE_DESCRIPTOR base table	272
21.16 DIRECT_SUPERTABLES	277
21.23 METHOD_SPECIFICATION_PARAMETERS	277
21.24 METHOD_SPECIFICATIONS base table	278
21.25 PARAMETERS base table	279
21.30 ROUTINE_COLUMN_USAGE base table	279
21.31 ROUTINE_PRIVILEGES base table	279
21.31a ROUTINE_ROUTINE_USAGE base table	279
21.32 ROUTINE_TABLE_USAGE base table	280
21.33 ROUTINES base table	280
21.34 SCHEMATA base table	282
21.35 SQL_FEATURES base table	282
21.36 SQL_IMPLEMENTATION_INFO base table	283
21.37 SQL_LANGUAGES base table	284
21.38 SQL_SIZING base table	284
21.39 SQL_SIZING_PROFILES base table	285
21.43 TABLES base table	285
21.44 TRANSFORMS base table	285
21.45 TRANSLATIONS base table	286
21.47 TRIGGER_COLUMN_USAGE base table	286
21.47a TRIGGER_ROUTINE_USAGE base table	287
21.48 TRIGGER_TABLE_USAGE base table	288
21.49 TRIGGERS base table	288
21.52 USER_DEFINED_TYPES base table	289
21.54a VIEW_ROUTINE_USAGE base table	290
21.56 VIEWS base table	291
22.1 SQLSTATE	291
23.1 General conformance requirements	293
Annex A SQL conformance summary	293
Annex B Implementation-defined elements	299
Annex C Implementation-dependent elements	303
Annex E Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996	304
Annex F SQL feature and package taxonomy	307

**iTeh STANDARD PREVIEW**  
**(standards.iteh.ai)**  
(Blank page)

[ISO/IEC 9075-2:1999/Cor 2:2003](https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003)

<https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003>



# Information technology — Database languages — SQL —

## Part 2: Foundation (SQL/Foundation)

### TECHNICAL CORRIGENDUM 2

## 2 Normative references

1. *Rationale: Add missing references.*

Insert the following items to this Subclause:

ISO/IEC 8859-1:1998, Information technology — 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1.

ISO/IEC 6429:1992, Information technology — Control functions for coded character sets.

The Internet Assigned Numbers Authority, Character sets.  
<http://www.iana.org/assignments/character-sets>

### 3.1.1 Definitions taken from ISO/IEC 10646

**iTeh STANDARD PREVIEW**  
(standards.iteh.ai)

1. *Rationale: Remove unused definitions.*

Replace the Subclause with: <http://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003>

This part of ISO/IEC 9075 makes use of the following terms defined in ISO/IEC 10646:

- a) character  
NOTE 0.1 – This is identical to the Unicode definition of *abstract character*. In ISO/IEC 9075, when the relevant character repertoire is UCS, a character can be thought of as *that which is represented by one code point*.
- b) repertoire

### 3.1.2 Definitions taken from Unicode

1. *Rationale: Remove unused definitions.*

Replace the Subclause with:

This part of ISO/IEC 9075 makes use of the following terms defined in The Unicode Standard:

- a) character encoding form
- b) code point
- c) control character
- d) transcoding

### 3.1.5 Definitions provided in Part 2

1. *Rationale: Clarify assignable and comparable.*

Replace item a) with:

- a) **assignable:** (of types, taken pair wise) The characteristic of a data type, T1, that permits a value of T1 to be assigned to a site of a specified data type, T2 (where T1 may be the same as T2).

2. *Rationale: Clarify the definition of “assignment”*

Replace item b) with:

- b) **assignment:** The operation whose effect is to ensure that the value at a site *T* (known as the target) is identical to a given value *S* (known as the source). Assignment is frequently indicated by the use of the phrase “*T* is set to *S*” or “the value of *T* is set to *S*”.

3. *Rationale: Clarify the definition of “comparable”*

Replace item i) with:

- i) **comparable** (of a pair of values): Capable of being compared, according to the rules of Subclause 8.2 “<comparison predicate>”. In most, but not all, cases the values of a data type can be compared one with another. For the specification of comparability of individual data types, see Subclauses 4.2 to 4.11.

4. *Rationale: Correct and clarify the notion of “distinct”*

Replace item l) with:

- l) **distinct:** (of a pair of comparable values): Informally: not equal or not both null or having a pair of corresponding components that are distinct. Formally:

Two null values of comparable type are not distinct.

A null value and a non-null value of comparable type are distinct.

For two non-null values, the following rules apply:

- Two values of predefined type or reference type are distinct if and only if they are not equal.
- If two values *V1* and *V2* are of a user-defined type whose comparison form is RELATIVE or MAP and the result of comparing them for equality according to Subclause 8.2, “<comparison predicate>” is *Unknown*, then it is implementation-dependent whether they are distinct or not; otherwise, they are distinct if and only if they are not equal.
- If two values *V1* and *V2* are of a structured type whose comparison form is STATE, then they are distinct if their most specific types are different, or if there is an attribute *A* of their common most specific type such that the value of *A* in *V1* and the value of *A* in *V2* are distinct.

- Two rows (or partial rows) are distinct if and only if at least one of their pairs of respective fields is distinct.
- Two arrays are distinct if the arrays do not have the same cardinality or if, for arrays of the same cardinality, elements in the same ordinal position in the two arrays are distinct.

NOTE 0.1 — The result of evaluating whether or not two comparable values are distinct is never *Unknown*. The result of evaluating whether or not two values that are not comparable, for example, values of a user-defined type that has no comparison type, is not defined.

5. *Rationale: Define "equals".*

Insert the following definition:

- o.1) **equal** (of a pair of comparable values): Yielding *True* if passed as arguments in a <comparison predicate> in which the <comp op> is <equals operator>. See Subclause 8.2, "<comparison predicate>".

6. *Rationale: Define "identical"*

Insert the following definition:

- s.1) **identical** (of a pair of values): Indistinguishable, in the sense that it is impossible, by any means available in SQL, to detect any difference between them. For the full definition, see Subclause 9.0 "Determination of identical values".

7. *Rationale: Clarify the distinction between character sets and character repertoires.*  
<https://standards.iteh.ai/catalog/standards/sist/6c2cfc78-2787-4151-9b4f-83ea8bd1be10/iso-iec-9075-2-1999-cor-2-2003>

Replace item rr) with:

- rr) **transliteration**: A method of translating characters in one character set into characters of the same or a different character set.

8. *Rationale: Remove <identifier ignorable character>s from the definition of <white space>.*

In item vv) delete the following bullet items:

- U+200C, Zero Width Non-Joiner
- U+200D, Zero Width Joiner
- U+200E, Left-To-Right Mark
- U+200F, Right-To-Left Mark

9. *Rationale: Enhance the definition of <white space> for implementations supporting Unicode 3.0.*

In item vv) insert the following bullet item:

- U+202F, Narrow No-Break Space

10. *Rationale: Remove Zero Width Space from the definition of <white space>*

In item vv), delete the following bullet item:

- U+200B, Zero Width Space

### 3.3.1.2 Other terms

1. *Rationale: Add definition of direct result of executing an SQL-statement.*

Insert the following Subclause:

#### 3.3.1.2 Other terms

An SQL-statement *SI* may be said to be executed as a *direct result of executing an SQL-statement* if *SI* is the SQL-statement contained in an <externally-invoked procedure> or <SQL-invoked routine> that has been executed.

## 4.1 Data types

1. *Rationale: Change "user-defined data type" to "user-defined type".*

Replace the 3<sup>rd</sup> paragraph with:

SQL supports three sorts of data types: *predefined data types*, *constructed types*, and *user-defined types*. Predefined data types are sometimes called the "built-in data types", though not in this International Standard. User-defined types can be defined by a standard, by an implementation, or by an application.

2. *Rationale: Correct oversight in definition of directly based on.*

Replace the 11<sup>th</sup> paragraph with:

A structured type *ST* is *directly based on* a data type *DT* if any of the following are true:

- DT* is the declared type of some attribute of *ST*.
- DT* is the direct supertype of *ST*.
- DT* is a direct subtype of *ST*.
- DT* is compatible with *ST*.

3. *Rationale: Add definition of non-referentially based on.*

Insert the following paragraph immediately after the 15<sup>th</sup> paragraph:

A data type *DT1* is non-referentially based on a data type *DT2* if *DT1* is not a reference type and is either directly based on *DT2* or is directly based on some data type that is non-referentially based on *DT2*.

4. *Rationale: Define “usage-dependent” for use in Access Rules*

Insert the following inserted paragraph immediately before the 16th paragraph:

A type *TY* is *usage-dependent* on a user-defined type *UDT* if one of the following conditions is true:

- *TY* is *UDT*;
- *TY* is a reference type whose referenced type is *UDT*;
- *TY* is a row type, and the declared type of a field of *TY* is usage-dependent on *UDT*; or
- *TY* is a collection type, and the declared element type of *TY* is usage-dependent on *UDT*.

5. *Rationale: Provide consistent rules for comparison operations.*

Insert the following paragraphs after the last paragraph:

Ordering and comparison of values of the predefined data types requires knowledge only about those predefined data types. However, to be able to compare and order values of constructed types or of user-defined types, additional information is required. We say that some type *T* is *S*-ordered, for some set of types *S*, if, in order to compare and order values of type *T*, information about ordering at least one of the types in *S* is first required. A definition of *S*-ordered is required for several *S* (that is, for several sets of types), but not for all possible such sets.

The general definition of *S*-ordered is this:

Let *T* be a type and let *S* be a set of types. Then *T* is *S*-ordered if one of the following is true:

1. *T* is a member of *S*.
2. *T* is a row type and the declared type of some field of *T* is *S*-ordered.
3. *T* is an array type and the element type of *T* is *S*-ordered.
4. *T* is a structured type whose comparison form is STATE and the declared type of some attribute of *T* is *S*-ordered.
5. *T* is a user-defined type whose comparison form is MAP and the return type of the SQL-invoked function that is identified by the <map function specification> is *S*-ordered.
6. *T* is a reference type with a derived representation and the declared type of some attribute enumerated by the <derived representation> is *S*-ordered.

The notion of *S*-ordered is applied in the following definitions:

A type *T* is *LOB-ordered* if *T* is *L*-ordered, where *L* is the set of large object types.

A type *T* is *array-ordered* if *T* is *ARR*-ordered, where *ARR* is the set of array types.

A type *T* is *reference-ordered* if *T* is *REF*-ordered, where *REF* is the set of reference types.

A type  $T$  is *DT-EC-ordered* if  $T$  is *DTE-ordered*, where *DTE* is the set of distinct types with EQUALS ONLY comparison form (DT-EC stands for “distinct type - equality comparison”)

A type  $T$  is *DT-FC-ordered* if  $T$  is *DTF-ordered*, where *DTF* is the set of distinct types with FULL comparison form.

A type  $T$  is *DT-NC-ordered* if  $T$  is *DTN-ordered*, where *DTN* is the set of distinct types with no comparison form.

A type  $T$  is *ST-EC-ordered* if  $T$  is *STE-ordered*, where *STE* is the set of structured types with EQUALS ONLY comparison form.

A type  $T$  is *ST-FC-ordered* if  $T$  is *STF-ordered*, where *STF* is the set of structured types with FULL comparison form.

A type  $T$  is *ST-NC-ordered* if  $T$  is *STN-ordered*, where *STN* is the set of structured types with no comparison form.

A type  $T$  is *ST-ordered* if  $T$  is either ST-EC-ordered, ST-FC-ordered or ST-NC-ordered.

A type  $T$  is *UDT-EC-ordered* if  $T$  is either DT-EC-ordered or ST-EC-ordered (UDT stands for “user-defined type”).

A type  $T$  is *UDT-FC-ordered* if  $T$  is either DT-FC-ordered or ST-FC-ordered

A type  $T$  is *UDT-NC-ordered* if  $T$  is either DT-NC-ordered or ST-NC-ordered

6. *Rationale: Definition required to define certain kinds of non-determinism involving comparison of datetime with time zone with datetime without time zone.*

Insert the following paragraph after the last paragraph:

The notion of *constituent* of a declared type  $DT$  is defined recursively as follows:

- $DT$  is a constituent of  $DT$ .
- If  $DT$  is a row type, then each field of  $DT$  is a constituent of  $DT$
- If  $DT$  is a collection type, then the element type of  $DT$  is a constituent of  $DT$ .
- Every constituent of a constituent of  $DT$  is a constituent of  $DT$ .

7. *Rationale: Clarify assignable and comparable.*

Insert the following paragraph after the last paragraph:

Two data types,  $T1$  and  $T2$ , are said to be *compatible* if  $T1$  is assignable to  $T2$ ,  $T2$  is assignable to  $T1$ , and their descriptors include the same data type name. If they are row types, it must further be the case that the declared types of their corresponding fields are pairwise compatible. If they are collection types, it must further be the case that their element types are compatible. If they are reference types, it must further be the case that their referenced types are compatible.

NOTE 1.1 — The data types "CHARACTER(*n*) CHARACTER SET *CS1*" and "CHARACTER(*m*) CHARACTER SET *CS2*", where *CS1* ≠ *CS2*, have descriptors that include the same data type name (CHARACTER), but are not mutually assignable; therefore, they are not compatible.

## 4.2 Character strings

1. *Rationale: Clarify the distinction between character sets and character repertoires.*

Replace the 1<sup>st</sup> and 2<sup>nd</sup> paragraphs with:

A character string is a sequence of characters. All the characters in a character string are taken from a single character set. A character string has a length, which is the number of characters in the sequence. The length is 0 (zero) or a positive integer.

A character type is described by a character type descriptor. A character type descriptor contains:

- The name of the specific character type (CHARACTER, CHARACTER VARYING, and CHARACTER LARGE OBJECT; NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, and NATIONAL CHARACTER LARGE OBJECT are represented as CHARACTER, CHARACTER VARYING, and CHARACTER LARGE OBJECT, respectively).
- The length or maximum length in characters of the character type.
- The catalog name, schema name, and character set name of the character set of the character type.
- The catalog name, schema name, and collation name of the collation of the character type.

The character set of a character type may be specified explicitly or implicitly.

The <key word>s NATIONAL CHARACTER are used to specify an implementation-defined character set. Special syntax (N'string') is provided for representing literals in that character set.

With two exceptions, a character string expression is assignable only to sites of a character type whose character set is the same. The exceptions are as specified in Subclause 4.2.4, "Universal character sets", and such other cases as may be implementation-defined. If a store assignment would result in the loss of non-`<space>` characters due to truncation, then an exception condition is raised. If a retrieval assignment or evaluation of a `<cast specification>` would result in the loss of characters due to truncation, then a warning condition is raised.

Character sets fall into three categories: those defined by national or international standards, those defined by SQL-implementations, and those defined by applications. The character sets defined by ISO/IEC 10646 and The Unicode Standard are known as *Universal Character Sets* (UCS) and their treatment is described in Subclause 4.2.4, "Universal character sets". Every character set contains the `<space>` character (equivalent to U+0020). An application defines a character set by assigning a new name to a character set from one of the first two categories. They can be defined to "reside" in any schema chosen by the application. Character sets defined by standards or by SQL-implementations reside in the Information Schema (named INFORMATION\_SCHEMA) in each catalog, as do collations defined by standards and collations, transliterations, and transcodings defined by SQL-implementations.