ISO/IEC JTC 1/SC 32 N1253

Summary of Voting on Document SC 32 N 1206,

Title: ISO/IEC FCD 9075-13 Information technology -- Database Language SQL - Part 13: SQL/JRT (for SQL:200n)

"P" Member	Approval	Approval with Comments	Disapproval	Abstention
Australia	**X**			
Belgium				
Brazil				
Canada			**X**	
China	**X**			
Czech Republic	**X**			
Egypt				
Finland				
Germany		**X**		
Italy				**X**
Japan			**X**	
Korea, Republic of	**X**			
Netherlands, The		**X**		
Norway				
Portugal				
Sweden	X			
United Kingdom			**X**	
United States			**X**	
Total "P"	5	**2**	**4**	**1**
"O" Member				
Austria				
Denmark				
France				
Russian Federation				
Switzerland				
Total "O"				

ITALY

Lack of Experts

National Body CAN Comments — 2005-02-03

32N1198, *ISO/IEC CD 9075-01 Information technology - Database Languages - SQL - Part 1: Framework (SQL/Framework)*
32N1199, *ISO/IEC CD 9075-02 Information technology - Database Languages - SQL - Part 2: Foundation (SQL/Foundation)*
32N1201, *ISO/IEC CD 9075-03 Information technology - Database Languages - SQL - Part 3: Call-Level Interface (SQL/CLI)*
32N1202, *ISO/IEC CD 9075-04 Information technology - Database Languages - SQL - Part 4: Persistent Stored Modules (SQL/PSM)*
32N1203, *ISO/IEC CD 9075-09 Information technology - Database Languages - SQL - Part 9: Management of External Data (SQL/MED)*
32N1204, *ISO/IEC CD 9075-10 Information technology - Database Languages - SQL - Part 10: Object language bindings (SQL/OLB)*
32N1205, *ISO/IEC CD 9075-11 Information technology - Database Languages - SQL - Part 11: Information and Definition Schemas (SQL/Schemata)*
32N1206, *ISO/IEC CD 9075-13 Information technology - Database Languages - SQL - Part 13: SQL Routines and Types Using the Java™ Programming Language(SQL/JRT)*

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
CD SQL/Framework						
	CAN-P01-001		1-Major Technical	*P01-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/Foundation						
	CAN-P02-001		1-Major Technical	*P02-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/CLI						
	CAN-P03-001		1-Major Technical	*P03-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/PSM						
	CAN-P04-001		1-Major Technical	*P04-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	
CD SQL/MED						
	CAN-P09-001		1-Major Technical	*P09-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/OLB						
	CAN-P10-001		1-Major Technical	*P10-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/Schemata						
	CAN-P11-001		1-Major Technical	*P11-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
CD SQL/JRT						
	CAN-P13-001		1-Major Technical	*P13-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	

ISO/IEC JTC1/SC32/WG3:TXL-034 – DIN NI-32 N 0642

Authoritative Version: Adobe Acrobat Portable Document Format (PDF)

ISO

International Organization for Standardization

DIN

Deutsches Institut für Normung

Din NI-32
Database
ISO/IEC JTC 1/SC 32
Data Management and Interchange
WG 3
Database Languages

Title: German Comments on SC32 N 1156: ISO/IEC CD 9075-1, 2, 3, 4, 9, 10, 11, 13:200x(E)

Status: Consolidated comments to assist with resolution of CD ballot comments

Author: Jörn Bartels (Germany)

National Body DEU Comments — 2005-03-08

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
ISO/IEC FCD 9075-01:200x(E) SQL/Framework						
1	DEU-P01-010		1-Major Technical	*P01-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
ISO/IEC FCD 9075-02:200x(E) SQL/Foundation						
2	DEU-P02-010		1-Major Technical	*P02-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
3	DEU-P02-020		1-Major Technical	*P02-07.6 Table reference*	Queries of the form SELECT … FROM <joined table> ….. Do not seem to be supported anymore. This is due to changes proposed in DRS-077. **Solution** None provided with comment.	
ISO/IEC FCD 9075-03:200x(E) SQL/CLI						
4	DEU-P03-010		1-Major Technical	*P03-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
ISO/IEC FCD 9075-04:200x(E) SQL/PSM						
5	DEU-P04-010		1-Major Technical	*P04-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
6	DEU-P11-020	DEU-P11-030	1-Major Technical	*P04-18.2 "MODULE_PRIVILEGES" Table*	The table MODULE_PRIVILEGES stores the privileges granted on a specific module. The same information could be stored in the table USAGE_PRIVILEGES. This would simplify the definition schema and standardise the way, how privileges are stored. **Solution** None provided with comment.	
7	DEU-P11-030	DEU-P11-020	1-Major Technical	*P04-18.2 "MODULE_PRIVILEGES" Table*	The constraint MODULE_PRIVILEGE_GRANTOR_CHECK and MODULE_PRIVILEGE_GRANTEE_CHECK reference still the tables ROLES and USERS. They are gone!	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution Replace the check constraints with a foreign key on AUTHORIZATIONS.	
ISO/IEC FCD 9075-09:200x(E) SQL/MED						
8	DEU-P09-010		1-Major Technical	*P09-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
9	DEU-P09-020		2-Minor Technical	*P09-25 Definition Schema*	Some tables which are introduced by this clause have for most columns no NOT NULL constraint, where it should be defined. This are at least 25.4 FOREIGN_DATA_WRAPPERS base table, 25.8 FOREIGN_TABLES base table and 25.10 ROUTINE_MAPPINGS base table **Solution** None provided with comment.	
10	DEU-P09-030		3-Major Editorial	*P09-25.2 DATA_TYPE_DESCRIPTOR" table*	The constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS is entirely replaced. This leads to problems of desynchronisation with SQL/Schemata. It does also not allow modifications from other parts (like SQL/XML) of the standard. The constraint does also currently not check the NULL applicability of the columns, as described in Description 2), which are inserted by this constraint. **Solution** None provided with comment.	
11	DEU-P09-040		3-Major Editorial	*P09-25.2 DATA_TYPE_DESCRIPTOR" table*	The Descriptions 2) and 3) are in conflict with each other. They describe both the NULLability of the newly introduced columns. There is a conflict if both come to different results. **Solution** The Descriptions 2) and 3) should be merged.	
12	DEU-P09-050		2-Minor Technical	*P09-25.4 "FOREIGN_DATA_WRAPPERS" table*	There is no constraint, which verifies the existence of the catalog and the authorization Identifier, which is used. **Solution** None provided with comment.	
13	DEU-P09-060		2-Minor Technical	*P09-25.6 "FOREIGN_SERVERS" table*	There is no constraint, which verifies the existence of the catalog and the authorization Identifier, which is used. **Solution** None provided with comment.	
14	DEU-P09-070		2-Minor Technical	*P09-25.12 "TABLES" table*	There is no constraint, which verifies that for a FOREIGN table there is also an entry in the table FOREIGN_TABLES. This could be done as it is done already in constraint TABLES_CHECK_NOT_VIEW of the table TABLES.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	
15	DEU-P09-080		2-Minor Technical	*P09-25.13 "USAGE_PRIVILEGES" table*	There is no constraint added, which checks the values of OBJECT_CATALOG and OBJECT_NAME as described in Description 1) There is also no modification of the constraint USAGE_PRIVILEGES_CHECK_REFERENCES_OBJECT which allows OBJECT_SCHEMA to be the empty string. It is currently also not allowed, that the OBJECT_TYPE is anything except 'DOMAIN', 'CHARACTER SET', 'COLLATION', 'TRANSLATION', 'SEQUENCE'. The use for a foreign-data wrapper or a foreign server requires a modification of constraint USAGE_PRIVILEGES_OBJECT_TYPE_CHECK. **Solution** None provided with comment.	
16	DEU-P09-090		2-Minor Technical	*P09-25.15 "USER_MAPPINGS" table*	There is no foreign key check for the column AUTHORIZATION_IDENTIFIER. **Solution** None provided with comment.	
ISO/IEC FCD 9075-10:200x(E) SQL/OLB						
17	DEU-P10-010		1-Major Technical	*P10-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
ISO/IEC FCD 9075-11:200x(E) SQL/Schemata						
18	DEU-P11-010		1-Major Technical	*P11-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	
19	DEU-P11-020		1-Major Technical	*P11-No specific location*	It is not clear, which tables should have a direct or indirect relationship to the table SCHEMATA. For some tables is a foreign key defined, for some is a check constraint defined, which checks the foreign key relationship only when there are schemas in the same catalog. There should be an explanation for this distinction and all relationships need to be checked for correctness. An example for a dubious relationship is the constraint TRIGGERS_REFERENCES_TABLES. Is it really possible, to define a trigger on a table of another catalog. This is especially strange, as the table	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					TRIGGERED_UPDATE_COLUMNS has a direct foreign key to COLUMNS. **Solution** None provided with comment.	
20	DEU-P11-030		3-Major Editorial	*P11-5.54 Short name views*	The View definitions in 5.54 Short name views should be sorted according to the order of the base views (i.E. position of CONSTR_COL_USAGE). **Solution** Order them according to base view order.	
21	DEU-P11-040		2-Minor Technical	*P11-5.78 "SQL_LANGUAGES" View*	The View SQL_LANGUAGES is depricated. In Note 6 is a reference to SQL/Framework Subclause 6.4. There is a backwards reference, in Note 9, which says "The equivalent information is available to the SQL user in the Information Schema." When we delete the view, it is not clear if the Note 9 should also be deleted. **Solution** None provided with comment.	
22	DEU-P11-050		2-Minor Technical	*P11-6.9 "CHARACTER_ENCODING_FORMS" Table*	The NOT NULL Constraints are not needed, as all columns are part of the primary key. **Solution** Delete the NOT NULL constraints.	
23	DEU-P11-060		2-Minor Technical	*P11-6.10 "CHARACTER_REPERTOIRES" Table*	The NOT NULL Constraints on the column CHARACTER_REPERTOIRE_NAME is not needed, as the column is part of the primary key. **Solution** Delete the NOT NULL constraint.	
24	DEU-P11-070		4-Minor Editorial	*P11-6.11 "CHARACTER_SETS" Table*	The column NUMBER_OF_CHARACTERS is in the last Edition of the Standard depricated and should now be deleted. **Solution** Delete the column. Do the same in the View Definition 5.12 CHARACTER_SETS view and in 5.78 Short name views in the view CHARACTER_SETS_S. Delete the corresponding List Elements 1) and 2) in Annex C.	
25	DEU-P11-080		2-Minor Technical	*P11-6.16 "COLLATIONS" Table*	There is no constraint for the column CHARACTER_REPERTOIRE_NAME defined. It needs to reference the Table CHARACTER_REPERTOIRES. **Solution** Add the constraint COLLATIONS_FOREIGN_KEY_CHARACTER_REPERTOIRES FOREIGN KEY (CHARACTER_REPERTOIRES) REFERENCES CHARACTER_REPERTOIRES.	
26	DEU-P11-090		4-Minor Editorial	*P11-6.16 "COLLATIONS" Table*	The columns COLLATION_TYPE, COLLATION_DICTIONARY, and COLLATION_DEFINITION are in the last edition of the Standard depricated and should now be deleted. **Solution** Delete the columns. Do the same in the View Definition 5.15 COLLATIONS view and in 5.78 Short name views in the view COLLATIONS_S. Delete the corresponding List Elements 3) and 4) in Annex C.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
27	DEU-P11-100		2-Minor Technical	*P11-6.16 "COLLATIONS" Table*	The column PAD_ATTRIBUTE has no NOT NULL check constraint, even that there is in the description no explanation of the meaning of a possible NULL value. **Solution** None provided with comment.	
28	DEU-P11-110		2-Minor Technical	*P11-6.20 "COLUMNS" Table*	For the columns IS_GENERATED and IDENTITY_GENERATION are no check constraints specified, but in the description are Lists of allowed values. **Solution** Add to the column IS_GENERATED the following column level constraint: CONSTRAINT COLUMNS_IS_GENERATED_CHECK CHECK (IS_GENERATED in ('NEVER', 'ALWAYS')) Add to the column IDENTITY_GENERATION the following column level constraint: CONSTRAINT COLUMNS_ IDENTITY_GENERATION_CHECK CHECK (IDENTITY_GENERATION IN ('ALWAYS', 'BY DEFAULT'))	
29	DEU-P11-120		2-Minor Technical	*P11-6.21 "DATA_TYPE_DESCRIPTORS" Table*	The constraint DATA_TYPE_DESCRIPTOR_FOREIGN_KEY_SCHEMATA assures that the values of USER_DEFINED_TYPE_CATALOG and USER_DEFINED_TYPE_SCHEMA have corresponding rows in the table SCHEMATA. The constraint DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_UDT allows that the value for the column USER_DEFINED_TYPE_CATALOG has no corresponding row in SCHEMATA. As this is not possible according to the first constraint, we could rewrite this constraint as a foreign key. It is not clear, if this is intended. **Solution** None provided with comment.	
30	DEU-P11-130		2-Minor Technical	*P11-6.21 "DATA_TYPE_DESCRIPTORS" Table*	The columns SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are not checked against the possible values in the table TABLES. **Solution** None provided with comment.	
31	DEU-P11-140		4-Minor Editorial	*P11-6.21 "DATA_TYPE_DESCRIPTORS" Table*	The constraint DATA_TYPE_DESCRIPTOR_CHECK_OBJECT_TYPE should be a column constraint, as it references only the column OBJECT_TYPE. **Solution** Remove the preceding comma.	
32	DEU-P11-150		4-Minor Editorial	*P11-6.21 "DATA_TYPE_DESCRIPTORS" Table*	In the constraint DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_COLLATION_CHARACTER_SET_APPLICABILITY should be a comma "," at the end of the 10th line of the constraint. **Solution** Add the missing comma.	
33	DEU-P11-		2-Minor Technical	*P11-6.24 "DOMAIN_CONSTRAINTS" Table*	Should there be a NOT NULL check constraint for the columns IS_DEFERRABLE and INITIALLY_DEFERRED?	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
	160				In the description is no explanation of the meaning of a possible null value. **Solution** None provided with comment.	
34	DEU-P11-170		2-Minor Technical	*P11-6.31 "PARAMETERS" Table*	The foreign key constraint PARAMETERS_FOREIGN_KEY_SCHEMATA does not check the name of the routine. This constraint should be removed and instead there should be a foreign key to ROUTINES be defined. **Solution** None provided with comment.	
35	DEU-P11-180		2-Minor Technical	*P11-6.31 "PARAMETERS" Table*	There needs to be a unique constraint defined, which guarantees the uniqueness of a parameter name for a routine. **Solution** Add the constraint: CONSTRAINT PARAMETERS_UNIQUE_CHECK (UNIQUE SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, PARAMETER_NAME)	
36	DEU-P11-190		2-Minor Technical	*P11-6.31 "PARAMETERS" Table*	For the columns FROM_SQL_SPECIFIC_CATALOG, FROM_SQL_SPECIFIC_SCHEMA, and FROM_SQL_SPECIFIC_ NAME and TO_SQL_SPECIFIC_CATALOG, TO_SQL_SPECIFIC_SCHEMA, and TO_SQL_SPECIFIC_NAME is no foreign key check defined. **Solution** None provided with comment.	
37	DEU-P11-200		1-Major Technical	*P11-6.36 "ROUTINE_PRIVILEGES" Table*	The table ROUTINE_PRIVILEGES stores the privileges granted on a specific routine. The same information could be stored in the table USAGE_PRIVILEGES. This would simplify the definition schema and standardise the way, how privileges are stored. **Solution** None provided with comment.	
38	DEU-P11-210		2-Minor Technical	*P11-6.41 "SCHEMATA" Table*	For the columns DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA and DEFAULT_CHARACTER_SET_NAME is a foreign key referencing the table CHARACTER_SETS missing. **Solution** Add the missing Foreign Key constraint: CONSTRAINT SCHEMATA_FOREIGN_KEY_ CHARACTER_SETS FOREIGN KEY (DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, DEFAULT_CHARACTER_SET_NAME) REFERENCES CHARACTER_SETS	
39	DEU-P11-		4-Minor Editorial	*P11-6.48 "TABLE_CONSTRAINTS" Table*	The constraint TABLE_CONSTRAINTS_UNIQUE_CHECK is not needed, as the uniqueness of the constraint name is already checked by the assertion UNIQUE_CONSTRAINT_NAME in subclause 6.4	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
	220				**Solution** A possible solution is to remove the superflouus constraint.	
40	DEU-P11-230		4-Minor Editorial	*P11-6.50 "TABLE_PRIVILEGES" Table*	In the constraint TABLE_PRIVILEGES_TYPE_CHECK is the last element of the inlist misspelled. It should be REFERENCES instead of EFERENCES **Solution** Fix the typo.	
41	DEU-P11-240		4-Minor Editorial	*P11-6.54 "TRIGGERED_UPDATE_COLUMNS" Table*	The constraint TRIGGERED_UPDATE_COLUMNS_FOREIGN_KEY_TRIGGERS is not needed, as a more restrictive relationship is already guaranteed by constraint TRIGGERED_UPDATE_COLUMNS_EVENT_MANIPULATION_CHECK. **Solution** A possible solution is to remove the superflouus constraint.	
42	DEU-P11-250		2-Minor Technical	*P11-6.55 "TRIGGER_COLUMN_USAGE" Table*	The table TRIGGER_COLUMN_USAGE should have a foreign Key to the table TRIGGER_TABLE_USAGE, and not to TRIGGERS. **Solution** Add the following constraint: TRIGGER_COLUMN_USAGE_FOREIGN_KEY_TRIGGER_TABLE_USAGE FOREIGN KEY (TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME) REFERENCES TRIGGER_TABLE_USAGE It might be possible to remove the constraint TRIGGER_COLUMN_USAGE_FOREIGN_KEY_TRIGGERS.	
43	DEU-P11-260		2-Minor Technical	*P11-6.62 "USER_DEFINED_TYPES" Table*	In the last query of the constraint USER_DEFINED_TYPES_CHECK_SOURCE_TYPE is the column OBJECT_TYPE not in the reference List of the IN clause. **Solution** None provided with comment.	
44	DEU-P11-270		2-Minor Technical	*P11-Appendix C 6)*	The columns FEATURE_ID and FEATURE_NAME of the view SQL_PACKAGES are in the last Edition of the Standard depricated and should now be deleted. But without these columns does the view not provide any usefull information. Should the entire View be deleted? **Solution** None provided with comment.	
ISO/IEC FCD 9075-13:200x(E) SQL/JRT						
45	DEU-P13-010		1-Major Technical	*P13-No specific location*	All Possible Problems and Editor's Notes must be satisfactorily resolved and all problems discovered during the course of the ballot resolution process must be satisfactorily resolved. **Solution** None provided with comment.	

END OF PAPER

ISO/IEC JTC1/SC32/WG3 TXL-nnn

3 March, 2005

ISO

International Organization for Standardization

ISO/IEC JTC 1/SC 32
Data Management and Interchange
WG 3
Database Languages

Title: Japan Ballot Comments on CD 9075:2007(E)

Status: Document to accompany ballot response

Author: Masashi Tsuchida ,Takaaki Shiratori, Takashi Kotera

Abstract: We present the comments of Japan on the CD ballot documents, to accompany our ballot response on that document.

References: [1] WG3:TXL-002 = 32N1198, ISO/IEC CD 9075-1, Information technology . Database languages SQL . Part 1: Framework (SQL/Framework)]
[2] WG3:TXL-003 = 32N1199, ISO/IEC CD 9075-2, Information technology . Database languages . SQL . Part 2: Foundation (SQL/Foundation)
[3] WG3:TXL-004 = 32N1201, ISO/IEC CD 9075-3, Information technology . Database languages . SQL . Part 3: Call-Level Interface (SQL/CLI)
[4] WG3:TXL-005 = 32N1202, ISO/IEC CD 9075-4, Information technology . Database languages . SQL . Part 4: Persistent Stored Modules (SQL/PSM)
[5] WG3:TXL-006 = 32N1203, ISO/IEC CD 9075-9, Information technology . Database languages . SQL . Part 9: Management of

External Data (SQL/MED)
[6] WG3:TXL-007 = 32N1204, ISO/IEC CD 9075-10, Information technology . Database languages . SQL . Part 10: Object language bindings (SQL/OLB)
[7] WG3:TXL-008 = 32N1205, ISO/IEC CD 9075-11, Information technology . Database languages . SQL . Part 11: Information and Definition Schemas (SQL/Schemata)
[8] WG3:TXL-009 = 32N1206, ISO/IEC CD 9075-11, Information technology . Database languages . SQL . Part 13: Information and Definition Schemas (SQL/JRT)

ISO/IEC JTC1/SC32/WG3 TXL-nnn

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
SQL/Framework						
001	JPN-P01-001		1-Major Technical	*P01-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
SQL/Foundation						
002	JPN-P02-002		1-Major Technical	*P02-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
003	JPN-P02-003		1-Major Technical	*P02-11.3, <table definition>*	It is allowed that <table contents source> which is <as subquery clause> with WITH DATA is specified for a temporary table. But a temporary table can not be materialized at table definition. **Solution** None provided with comment.	
004	JPN-P02-004		1-Major Technical	*P02-14.8, <insert statement>*	An INSERT statement has no different effects on identity columns specified GENERATED ALWAYS and that specified GENERATED BY DEFAULT. **Solution** None provided with comment.	
SQL/CLI						
005	JPN-P03-001		1-Major Technical	*P03-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
SQL/PSM						
006	JPN-P04-001		1-Major Technical	*P04-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
SQL/MED						
007	JPN-P05-001		1-Major Technical	*P05-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
SQL/OLB						
008	JPN-P10-001		1-Major Technical	*P10-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
SQL/Schema						
009	JPN-P11-001		1-Major Technical	*P11-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	
SQL/JRT						
009	JPN-P13-001		1-Major Technical	*P13-No specific location*	There are quite a few features to discuss a emerging next standard. Japan thinks that we should take enough time to add new features. **Solution** None provided with comment.	

ISO/IEC JTC1/SC32/WG3 TXL-031
2004-01-23

ISO
International Organization for Standardization
ISO/IEC JTC 1/SC 32
Data Management and Interchange
WG 3
Database Languages

Project: ISO: 1.32.3.5

Title: Ballot Comment on ISO/IEC CD 9075-1, -2, -3, -4, -9, -10, -11, and -13

Status: Netherlands National Body Comments

Author: Stephen Cannan (Editor)

References:

[1] SC32 N01198, CD 9075-1 Information Technology - Database Language SQL - Part 1: Framework (SQL/Framework) Jim Melton (Editor), December, 2004.
[2] SC32 N01199, CD 9075-2 Information Technology - Database Language SQL - Part 2: Foundation (SQL/Foundation) Jim Melton (Editor), December, 2004.
[3] SC32 N01201, CD 9075-3 Information Technology - Database Language SQL - Part 3: Call Level Interface (SQL/CLI) Jim Melton (Editor), December, 2004.
[4] SC32 N01202, CD 9075-4 Information Technology - Database Language SQL - Part 4: Persistent Stored Modules (SQL/PSM) Jim Melton (Editor), December, 2004.
[5] SC32 N01203, CD 9075-9 Information Technology - Database Language SQL - Part 9: Management of External Data (SQL/MED) Jim Melton (Editor), December, 2004.
[6] SC32 N01204, CD 9075-10 Information Technology - Database Language SQL - Part 10: Object Language Bindings (SQL/OLB) Jim Melton (Editor), December, 2004.
[7] SC32 N01205, CD 9075-11 Information Technology - Database Language SQL - Part 11: Schemata (SQL/Schemata) Jim Melton (Editor), December, 2004.
[8] SC32 N01206, CD 9075-13 Information Technology - Database Language SQL - Part 13: Java Routines and Types (SQL/JRT) Jim Melton (Editor), December, 2004.

The Netherlands vote is:

SQL/Framework	No with comments
SQL/Foundation	No with comments
SQL/CLI	No with comments
SQL/PSM	No with comments
SQL/MED	No with comments
SQL/OLB	No with comments
SQL/Schemata	No with comments
SQL/JRT	Yes with comments

If all problems and technical errors, i.e. those identified in this ballot, and those identified during the editing meeting(s), are resolved to our satisfaction, then the Netherlands will change its NO votes to YES votes.

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
SQL/Framework						
	NLD-P01-001		3-Major Editorial	*P01-04.04, SQL data types*	FRM-002 The following Language Opportunity has been noted: **Source:** DBL:BBN-167/X3H2-98-386 **Language Opportunity:** Section needs a better organization There should be a section called SQL Data Types. Then a short definition of what is meant by an SQL data type. Then the list of the five types of data types (predefined, row type, user-defined type, collection type, and reference type). Then there should be a definition for each. **Solution** None provided with comment.	
	NLD-P01-002		2-Minor Technical	*P01-06.03.03.03, Rule evaluation order*	**WG3-P01-001** The referenced subclause includes the following text: In general, if some syntactic element contains more than one other syntactic element, then the General Rules for contained elements that appear earlier in the production for the containing syntactic element are applied before the General Rules for contained elements that appear later. For example, in the production: `<A> ::= <B> <C>` the Syntax Rules and Access Rules for <A>,<B>,and <C>are effectively applied simultaneously. The General Rules for <B>are applied before the General Rules for <C>, and the General Rules for <A>are applied after the General Rules for both <B>and <C>. In SQL/Foundation, Subclause 13.5, "<SQL procedure statement>", is a clear exception to this general rule for General Rules, for the GRs of the particular contained statement (e.g., an <insert statement>) are clearly intended to be invoked only when a GR in Subclause 13.5 explicitly states that the contained statement to be executed. Now, it might be that the introductory words, "In general", can be taken to imply that there are some exceptions, but in that case shouldn't the exceptions be explicitly mentioned? **Solution** None provided with comment.	
SQL/Foundation						
	NLD-P02-001		1-Major Technical	*P02-04.32.01, General description of cursors*	FND-975 The following Possible Problem has been noted: **Source:** WG3:SIA-030 = H2-2004-??? **Possible Problem:** Subclause 4.32.1, “General description of cursors”, contains: For every <declare cursor> **in** [emphasis added] an SQL-client module, a cursor	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					is effectively created when an SQL-transaction (see Subclause 4.35, “SQL-transactions”) referencing the SQL-client module is initiated. For every <dynamic declare cursor> in an <SQL-client module definition>, a cursor is effectively created when an SQL-transaction (see Subclause 4.35, “SQL-transactions”) referencing the <SQLclient module definition> is initiated. An extended dynamic cursor is **also** [emphasis added] effectively created when an <allocate cursor statement> is executed within an SQL-session and destroyed when that SQL-session is terminated. This text suffers from several problems, all of which probably need to be addressed at the same time: 1) The first paragraph entertains the notion of a piece of SQL syntax appearing inside something that is not a piece of SQL syntax. It seems that either "<declare cursor>" should be replaced by "cursor", or "SQL-client module" should be replaced by "<SQL-client module definition>". In either case there would be knock-on effects on the remaining text. Note that the second paragraph prefers to talk about syntactic containment exclusively, but its text is too suspect for it to be used as a guideline for correcting the first paragraph. 2) The first paragraph entertains the notion of an SQL-transaction referencing an SQL-client module. Regardless of whether this should be SQL-client module or <SQL-client module definition>, it is not clear exactly what it means for an SQL-transaction that is the process of being initiated to “reference” that thing. Text elsewhere in SQL/Foundation (for example, in Subclause 16.7, “<commit statement>”), strongly suggests that several distinct SQL-client modules can be “associated with” the same current SQL-transaction. Can they be associated with the SQL-transaction without also being referenced by it? For that matter, can they be referenced by it without also being associated with it? If “referenced by” and “associated with” are synonymous, then how can all the SQL-client modules referenced by SQL-transaction *T* be known when *T* is initiated? 3) The second paragraph entertains the notion of creation of a cursor, and yet we have not been able to find any mention of this concept in any General Rule. Subclause 19.8, “<deallocate prepared statement>”, GR3) does require destruction of certain cursors, and this is corroborated (redundantly?) by Subclause 19.15, “<allocate cursor statement>”, GR3)d). However, neither Subclause 19.6, “<prepare statement>”, nor Subclause 19.15, “<allocate cursor statement>”, has any GR requiring a cursor to be created. 4) As already noted, a cursor, having been created in somewhat mysterious circumstances, is never destroyed (unless it happens to be an allocated cursor, or a cursor declared in a <compound statement>, see SQL/PSM, Subclause 13.1, “<compound statement>”, GR3)c)ii)2) and GR5)). It seems, then, that if *n* SQL-transactions in the same SQL-session “reference” the same SQL-client module,	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					then each <declare cursor> contained in the corresponding <SQL-client module definition> causes the creation of *n* distinct cursors. And yet Subclause 14.2, "<open statement>", SR1), says "Let *CR* be the cursor specified by *DC*", where *DC* is a <declare cursor>. There are two problems with this: — It is not clear which of those *n* cursors is the one specified by *DC*. Of course, if the standard clearly specified that all but one of these had been destroyed by this time, then there would be no ambiguity. — The cited text in Subclause 4.32.1, "General description of cursors", makes it clear that a cursor comes into existence at run-time and therefore, not being a schema object, should not be referred to in a syntax rule. Since the SQL-session context already includes cursor positions, perhaps it should also be defined to include cursors. 5) The final sentence contains the word "also", which could be understood as suggesting that some way of creating an extended dynamic cursor has already been mentioned. Moreover, it ignores the possibility of such a cursor being destroyed as a consequence of its prepared statement being deallocated before SQL-session termination. **Solution** None provided with comment.	
	NLD-P02-002		2-Minor Technical	*P02-03.01, Definitions*	FND-953 The following Language Opportunity has been noted: **Language Opportunity:** There has been a discussion about Unicode 4.0 on the ISODBL list. [Ake has] found out that Note 7 in SQL/Foundation will be affected, bescuase it contains explicit code points. U+180E and U+205F have been added to the "Zs" class in Unicode 4.0. Note that U+200B currently is of class "Zs", although it should not be treated as white-space. The Unicode Technical Committee will probably change the class for U+200B (ZERO-WIDTH SPACE) to "Cf" in the near future. **Solution** None provided with comment.	
	NLD-P02-003		2-Minor Technical	*P02-04.10, Collection types*	FND-845 The following Language Opportunity has been noted: **Source:** WG3:YYJ-016 (CAN-P02-001, USA-P02-005) **Language Opportunity:** The next edition of the SQL standard should standardize the syntax and semantics of one or more additional collection types. **Solution** None provided with comment.	
	NLD-P02-004		1-Major Technical	*P02-04.14.02, Types of tables*	FND-944 The following Possible Problem has been noted: **Source:** WG3:HBA-042 = H2-2003-___ **Possible Problem:** This Subclause, with paragraph numbers added for expository purposes, says of created temporary tables:	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					1) A global temporary table is a named table defined by a <table definition> that specifies GLOBAL TEMPORARY. A created local temporary table is a named table defined by a <table definition> that specifies LOCAL TEMPORARY. Global and created local temporary tables are effectively materialized only when referenced in an SQL-session. Every SQL-client module in every SQL-session that references a created local temporary table causes a distinct instance of that created local temporary table to be materialized. That is, the contents of a global temporary table or a created local temporary table cannot be shared between SQL-sessions. 2) In addition, the contents of a created local temporary table cannot be shared between SQL-client modules of a single SQL-session. The definition of a global temporary table or a created local temporary table appears in a schema. In SQL language, the name and the scope of the name of a global temporary table or a created local temporary table are indistinguishable from those of a persistent base table. However, because global temporary table contents are distinct within SQL-sessions, and created local temporary tables are distinct within SQL-client modules within SQL-sessions, the *effective* <schema name> of the schema in which the global temporary table or the created local temporary table is instantiated is an implementation-dependent <schema name> that may be thought of as having been effectively derived from the <schema name> of the schema in which the global temporary table or created local temporary table is defined and the implementation-dependent SQLsession identifier associated with the SQL-session. 3) In addition, the *effective* <schema name> of the schema in which the created local temporary table is instantiated may be thought of as being further qualified by a unique implementation-dependent name associated with the SQL-client module in which the created local temporary table is referenced. 4) A declared local temporary table is a module local temporary table. A module local temporary table is a named table defined by a <temporary table declaration> in an SQL-client module. A module local temporary table is effectively materialized the first time it is referenced in an SQL-session, and it persists for that SQL-session. Neither the first sentence of paragraph 1 nor the General Rules of Subclause 11.3, "<table definition>", make it clear that a <table definition> creates a persistent (temporary) table descriptor. *Materialised* is not defined and the meaning added by the qualifier *effectively* is unclear; the use of *instantiated* in paragraph 3 suggests a distinction that is probably unintended. In paragraph 2, "*effective* <schema name> ... may be thought of as ..." [emphasis in original] doesn't tell us what the purpose of this thinking is, nor how the *effective* <schema name> differs from any possible *actual* one. It also misleads us into imagining that a local temporary table created in the same schema as a	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					global one might have the same <table name>, when on reflection it clearly can't. This is not, be it noted, analogous to extended names, as explicitly intended by DBL:LON-156. Furthermore, “unique implementation-dependent name associated with the SQL-client module in which the created local temporary table is referenced” is unclear. Is this name persistent? One interpretation is that every occurrence of the <table name> must be contained in the same <SQL-client module definition>. But perhaps it is intended to mean that there is no restriction on where the <table name> can occur; but only occurrences in externally-invoked procedures in the same module refer to the same thing (i.e. those of other modules refer to their own “local” temporary table). In paragraph 2, it is not clear whether a distinction is intended between “global temporary table contents” and “created local temporary tables”, but presumably not. In paragraph 2, the meaning of “distinct within SQL-sessions” is unclear, because there is only one SQLsession active at any one time (even though there may be dormant ones). **Solution** None provided with comment.	
	NLD-P02-005		1-Major Technical	*P02-04.14.02, Types of tables*	FND-945 The following Possible Problem has been noted: **Source:** WG3:HBA-042 = H2-2003-___ **Possible Problem:** In this Subclause, as modified by [PSM-WD], Subclause 4.3.1, “Types of tables”, with paragraph numbers added for expository purposes, says of declared temporary tables: 1) A declared local temporary table is a module local temporary table. A module local temporary table is a named table defined by a <temporary table declaration> in an SQL-client module. A module local temporary table is effectively materialized the first time it is referenced in an SQL-session, and it persists for that SQL-session. 2) A declared local temporary table may be declared in an SQL-client module. 3) Inserted by SQL/PSM A declared local temporary table may be declared in an SQL-server module. 4) A declared local temporary table that is declared in an SQL-client module is a named table defined by a <temporary table declaration> that is effectively materialized the first time any <externallyinvoked procedure> in the <SQL-client module definition> that contains the <temporary table declaration> is executed. A declared local temporary table is accessible only by <externally-invoked procedure>s in the <SQL-client module definition> that contains the <temporary table declaration>. The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					implementation-dependent SQL-session identifier associated with the SQL-session and a unique implementation-dependent name associated with the <SQL-client module definition> that contains the <temporary table declaration>. 5) Inserted by SQL/PSM A declared local temporary table that is declared in an SQL-server module is a named table defined by a <temporary table declaration> that is effectively materialized the first time any <module routine> in the <SQL-server module definition> that contains the <temporary table declaration> is executed. A declared local temporary table is accessible only by <module routine>s in the <SQL-server module definition> that contains the <temporary table declaration>. The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and the name of the <SQL-server module definition> that contains the <temporary table declaration>. The second sentence of paragraph 1 is no longer true when paragraphs 3 and 5 have been inserted by PSM. Moreover, whatever truth is expressed by paragraph 1 is repeated by paragraphs 2 and 4, which are specific to SQL-client modules. Evidently paragraphs 2 and 4 were inserted to correspond to paragraphs 3 and 5, so making paragraph 1 redundant, which should have been deleted at the same time. To say, in paragraphs 4 and 5, that a declared local temporary table has an *effective* <schema name> is misleading, since its name must be prefixed by MODULE. **Solution** None provided with comment.	
	NLD-P02-006		2-Minor Technical	*P02-04.14.02, Types of tables*	FND-969 The following Language Opportunity has been noted: **Source:** WG3:SIA-018 = H2-2004-429 **Language Opportunity:** Every view component is an underlying table. The reason that underlying table terminology was not used was that the hierarchy of underlying tables does not follow the hierarchy of syntactic containment, owing to the distinctive treatment accorded the tables and derived tables in the FROM clause of a <query specification> compared with other derived tables found in a <query specification>. If this issue can be overcome, it may be possible to eliminate the notion of view component and just use underlying tables. **Solution** None provided with comment.	
	NLD-P02-007		1-Major Technical	*P02-04.17, Integrity constraints*	FND-703 The following Possible Problem has been noted: **Source:** WG3:BBN-139/X3H2-98-363 **Possible Problem:** It seems that SQL3's specification of deferrable constraints is ill-specified. Referential constraints are based on the notion of marking rows for deletion	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					before the rows are effectively deleted at the end of the SQL statement. This is necessary because updates cascaded by referential constraints need to be “propagated” through rows marked for deletion in order to avoid anomalies (non deterministic behavior). If a referential constraint is deferred, then rows that need to be kept around for the execution of referential constraints will not be present at the end of the transaction (or when the referential constraint is turned to immediate). These rows will be deleted at the end of the SQL statements. So, it is unclear how referential constraints are checked in these cases (e.g., are we supposed to maintain multiple versions of the database and check the constraints against those versions? If so, how do the updates are "propagated" to the current version of the database?). Another problem with deferrable constraints is that stored procedures and triggers can never rely on the existence of a consistent database during their execution because the application that caused the invocation of the stored procedure and/or trigger could have deferred the checking of certain constraints prior to the invocation of the procedure or trigger. (Please note that this has also a major impact to the implementation of such concepts because plans generated by optimizers (e.g., the exploitation of a unique index) can be invalidated by deferring such constraints.) Also it is not clear to me that deferrable constraints and triggers work smoothly. First, BEFORE triggers execute BEFORE the SQL statement that activates them. However, the BEFORE execution cannot be guaranteed if referential constraints are deferred because the execution of the BEFORE trigger needs to be deferred as well. Second, if the BEFORE trigger is modifying the values of transition variables such that they can be inserted/updated with correct values in the database, what will happen with such values if the BEFORE trigger executes after the database has been updated? Third, triggers are executed in a well defined order. This is important to guarantee that changes to the database are done in a deterministic manner. If constraints are deferred, then one may end up deferring the execution of several instances of the same trigger for which there is no well defined order of execution. This will lead to non-deterministic behavior in the database. **Solution** None provided with comment.	
	NLD-P02-008		2-Minor Technical	*P02-04.27, SQL-invoked routines*	FND-725 The following Language Opportunity has been noted: **Source:** WG3:FRA-122/X3H2-98-688) **Language Opportunity:** Subclause 4.27, “SQL-invoked routines”, does not adequately describe the concepts of dynamic binding and subject function selection. **Solution** None provided with comment.	
	NLD-P02-009		2-Minor	*P02-04.32,*	FND-607 The following Language Opportunity has been noted:	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
			Technical	*Cursors*	**Source:** DBL:LGW-146/X3H2-97-349 **Language Opportunity:** The ability to hold a cursor through rollback will be extremely useful to applications. Yet the second bullet of this Subclause says "a holdable-cursor is closed no matter what its state if the SQL-transaction is terminated with a rollback operation." This provision is not always necessary according to Jim Gray and Andrewas Reuter "Transaction Processing: Concepts and Techniques". **Solution** None provided with comment.	
	NLD-P02-010		2-Minor Technical	*P02-04.32.01, General description of cursors*	FND-929 The following Possible Problem has been noted: **Source:** WG3:HBA-040 **Possible Problem:** Although the second paragraph of this subclause defines terms to denote both varieties of dynamic cursors, it does not provide a way of referring to a cursor that is *not* dynamic. **Solution** None provided with comment.	
	NLD-P02-011		1-Major Technical	*P02-04.33.04, SQL-statements and transaction states*	FND-923 The following Possible Problem has been noted: **Source:** WG3:HBA-029 **Possible Problem:** Subclause 4.33.4, “SQL-statements and transaction states”, includes: If the initiation of an SQL-transaction occurs in an atomic execution context, and an SQL-transaction has already completed in this context, then an exception condition is raised: *invalid transaction termination.* At first sight it doesn't seem possible for transaction termination to be followed by transaction initiation "in" the same atomic execution context. In general, transaction initiation is caused by execution of an SQL-statement of the transaction-initiating kind and transaction termination is caused by executing an SQL-statement of a different kind (COMMIT or ROLLBACK). Note that Subclause 13.5, “<SQL procedure statement>”, GR2), specifies that a new statement execution context is created whenever an <SQL procedure statement> is executed. Note also that in Subclause 4.33.5, “SQL-statement atomicity and statement execution contexts”, we are told (last paragraph) that an SQL-transaction cannot be explicitly terminated within an atomic execution context. We conclude that the cited paragraph is relevant only when execution of a transaction-initiating statement (a) actually causes a transaction to be initiated, and (b) causes an exception to be raised of the special transaction rollback kind (this being the only way of implicitly terminating a transaction). In the light of this observation, we perceive the following problems: 1) There is no General Rule in, for example, Subclause 13.5, “<SQL procedure statement>”, to confirm the cited text.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					2) If execution of an SQL-statement causes an exception to be raised, then all changes to SQL-data and schemas are cancelled anyway. As the failing statement is also the one that initiated the transaction, the effect seems to be the same as that of a successful rollback, so what's the point in raising an additional exception expressing the fact that the transaction cannot be terminated? After all, the user executing the statement in question wasn't even trying to terminate the current transaction! We wonder if the rule was intended to cater for some eventuality other than the only one we can find. Even if the foregoing analysis proves to be refutable, it might be a good idea to add an explanation to Subclause 4.33.4, “SQL-statements and transaction states”. **Solution** None provided with comment.	
	NLD-P02-012		1-Major Technical	*P02-04.33.05, SQL-statement atomicity and statement execution contexts*	FND-924 The following Possible Problem has been noted: **Source:** WG3:HBA-029 **Possible Problem:** Subclause 4.33.5, “SQL-statement atomicity and statement execution contexts”, includes: The statement execution context brought into existence by the execution of an atomic SQL-statement or the evaluation of a <subquery> is an atomic execution context. The inclusion of "or the evaluation of a <subquery>", and the GRs of Subclause 7.15, “<subquery>”, that back it up, seem questionable. Isn't expression evaluation always atomic? The question also arises as to whether deletion of the questionable text (and GRs) would make any material difference to the standard. A search of the SQL:2003 Foundation FDIS for the word "atomic" reveals no GRs that are conditional upon the atomicity or non-atomicity of a statement execution context. Instead, there are some special GRs for <subquery> that enforce its atomicity by creating and destroying a savepoint level, and in Subclause 13.5, “<SQL procedure statement>”, for undoing any changes to SQLdata or schemas made execution of by an atomic statement that terminates with an exception. It seems, then, that the only effects caused by atomicity are to do with savepoints and database updates. But it appears that database updates are not possible during evaluation of a subquery, being outlawed by Subclause 7.13, “<query expression>”, SR23): 1) 23) A <query expression> QE shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that possible modifies SQL-data. Note that the BNF production for <subquery> is <left paren> <query	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					expression> <right paren> and a <query expression> cannot contain an SQL procedure statement. SR23) in combination with the GRs of Subclause 10.4, “<routine invocation>”, makes it impossible for an evaluation of an <routine invocation> caused by evaluation of a <subquery> to cause an SQL-data change statement to be executed. Therefore it is impossible for evaluation of a <subquery> to have any effect on SQL-data or schemas (possibly explaining the lack of a GR in Subclause 7.15, “<subquery>”, specifying that changes to SQL-data and schemas are to be cancelled). Therefore any savepoints established during evaluation of a <subquery> have to be ineffectual. Therefore there is no point in establishing a new, atomic, statement execution context for the evaluation of a <subquery>. But that's not all! Consider the <query expression> SELECT foo() FROM T, and suppose that there is some flaw in the reasoning that leads to the conclusion that the effect of the invocation of foo() cannot possibly depend on whether the current statement execution context is atomic. In that case SELECT foo() FROM T would not in general be equivalent to SELECT * FROM (SELECT foo() FROM T) T, for the shorter expression does not contain a <subquery>, whereas the longer one does. In general, the consequences of the effect of evaluation of a <query expression> possibly varying according to whether it is enclosed in parentheses are very unclear and would surely raise very awkward problems for optimisers. The foregoing analysis also brings into question the following sentence in Subclause 4.33.4, “SQLstatements and transaction states”: 1) If an <SQL-control statement> causes the evaluation of a <subquery> and there is no current SQLtransaction, then an SQL-transaction is initiated before evaluation of the <subquery>. Perhaps this is pointless, too. If it proves not to be, the wisdom of starting a transaction in the middle of executing an SQL-statement - - at an indeterminate point in that execution, to boot - - is surely questionable. In any case, we note that the sentence is not borne out by the GRs of Subclause 7.15, “<subquery>”. **Solution** Delete "or evaluation of a <subquery>" from the cited sentence of Subclause 4.33.5, “SQL-statement atomicity and statement execution contexts”; possibly delete the cited sentence of Subclause 4.33.4, “SQL-statements and transaction states”; delete GRs 1) ("Let OLDSEC ..."and 4) ("All savepoints ...") of Subclause 7.15, “<subquery>”. A search of the SQL:2003 Foundation FDIS for "<subquery>" reveals that a change might also be needed in Subclause 4.33.3, “SQL-statements and SQL-data access indication”. Other Parts of SQL:2003 have not been checked to see if they might be affected. But see WG3:HBA-041.	
	NLD-P02-013		1-Major	*P02-04.35.02,*	FND-972 The following Possible Problem has been noted:	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
			Technical	*Savepoints*	**Source:** WG3:SIA-031 = H2-2004-??? **Possible Problem:** Neither here nor anywhere else is there any definition of the term savepoint. Paragraph 5 of this Subclause contains: If a <rollback statement> references a savepoint SS, then all changes made to SQL-data or schema subsequent to the establishment of the savepoint are canceled, all savepoints established since SS was established are destroyed, and the SQL-transaction is restored to its state as it was immediately following the execution of the <savepoint statement>. The state of an SQL-transaction is not defined, nor is it referred to in Subclause 16.8, “<rollback statement>”. Presumably what is being referred to is some (or all) of the SQL-session context. Which <savepoint statement> is not specified. Presumably the one that established *SS*. So perhaps "... following the establishment of *SS*"; or even "... as it was at that time". Perhaps what is really meant is something to the effect of, A savepoint is a preserved copy of (the values of specified elements) of the SQL-session context at the time a <savepoint statement> was executed, plus sufficient data to enable all subsequent changes to SQL-data or schemas in the current SQL-transaction to be canceled. When a <rollback statement> is executed, that contains a <savepoint specifier> *SS*, then all changes made to SQL-data or schema subsequent to the establishment of *SS* are canceled, all savepoints established since SS was established are destroyed, and elements of the SQL-session context are restored to the values that were preserved in *SS*. Paragraph 6 says: It is implementation-defined whether or not, or how, a <rollback statement> that references a <savepoint specifier> affects diagnostics area contents, the contents of SQL descriptor areas, and the status of prepared statements. This implementation-defined element is not mentioned in Annex B, “Implementation-defined elements” (or in Annex C, “Implementation-dependent elements”). **Solution:** Specify what happens in terms of the contents of the SQL-session context. **Solution** None provided with comment.	
	NLD-P02-014		1-Major Technical	*P02-04.37, SQL-sessions*	FND-954 The following Possible Problem has been noted: **Source:** WG3:ZSH-037R1/H2-2003-??? **Possible Problem:** WG3:FRA-045r4 proposed no changes to what is now WG3:ZSH-013, Subclause 4.37, "SQLsession". However, according to WG3:FRA-045r4, Section 2.1, "Authorization stack":	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					There is a stack of SQL-session contexts. There is one cell on this stack when the SQLsession begins. An additional SQL-session context is pushed on the stack for each <routine invocation>, and is removed when the <routine invocation> completes execution. There is no reference to this anywhere in this subclause, although there are various statements of the form "An SQL-session has a ...". Moreover, the list of SQL-session contents is incorrect and incomplete. The term "current SQL-session identifier" is listed, where the meaning of "current" is indicated in the following NOTE (55 in WG3:ZSH-013) and evidently used to distinguish the "current" SQL-session from dormant SQL-sessions. It is therefore probably intended to refer to the SQL-session identifier of the currently active (as opposed to dormant) SQL-session. If this surmise is correct, then the "current SQL-session user identifier" is missing. There is no reference to the authorization stack, though the two terms used to refer to the components of the only visible cell of that stack are mentioned. **Solution** None provided with comment.	
	NLD-P02-015		1-Major Technical	*P02-04.37.04, Execution contexts*	FND-955 The following Possible Problem has been noted: **Source:** WG3:ZSH-037R1/H2-2003-??? **Possible Problem:** This subclause contains the statement: There is always a statement execution context, a routine execution context, and zero or more trigger execution contexts. There is a significant and unnecessary inconsistency between the descriptions of routine execution contexts and trigger execution contexts. Consider what happens if an SQL-invoked routine R1 invokes another, R2. Are there now one or more than one routine execution contexts? The answer is clearly there is one in each of two levels of the stack of SQL-session contexts, as is made clear by Subclause 10.4, "<routine invocation>". Whether there is a routine execution context when no routine has been invoked is debatable: it could be (and indeed is) said that there is an empty one; or it could be said that there is none. In which case, it would be true to say that "there are zero or more routine execution contexts", as is said for trigger execution contexts. Consider now how it arises that there is more than one trigger execution context. The only case that springs to mind is that of the triggered action of a trigger T1, causing another trigger T2 to fire. In this case, each trigger will have a trigger execution context. However, it seems fairly clear that the triggered action of T2 cannot access the state changes in the trigger execution context of T1. Therefore, to say that there are, during the execution of T2, two trigger execution contexts, although true in a sense, is unhelpful. Moreover, we seem to be saying that these two trigger execution contexts are in	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					the same SQLsession context; unless, of course, T1 invokes a routine that causes T2 to fire, in which case a new SQL-session context is created, containing a new routine execution context. However, whether or not it contains, when created, the trigger execution context of T1, we are unable to discover. **Solution** None provided with comment.	
	NLD-P02-016		1-Major Technical	*P02-05.04, Names and identifiers*	FND-932 The following Possible Problem has been noted: **Source:** WG3:HBA-050R1 **Possible Problem:** SR19) of this subclause is: 19) An <identifier> that is a <correlation name> is associated with a table within a particular scope. The scope of a <correlation name> is either a <select statement: single row>, <subquery>, or <query specification> (see Subclause 7.6, "<table reference>"), or is a <trigger definition> (see Subclause 11.39, "<trigger definition>"). Scopes may be nested. In different scopes, the same <correlation name> may be associated with different tables or with the same table. The inclusion of <subquery> is puzzling. For consider that if such a scope is contained in a <subquery>, then it must also be wholly contained in some <query specification> contained in that <subquery>. Furthermore, a <subquery> that contains more than one <query specification> cannot possibly constitute the scope of any <correlation name>. For example: `( SELECT * FROM T1 UNION SELECT * FROM T2 )` The scope of any correlation name defined with such a <subquery> would be confined to the particular <query specification> in which it is defined. A scalar expression could be added to the <subquery> that includes an outer reference, but the <correlation name> used in that reference would have a wider scope than the <subquery>. The inclusion of <query specification> is also suspect, because the scope of a <correlation name> is not necessarily a whole <query specification>. **Solution** None provided with comment.	
	NLD-P02-017		1-Major Technical	*P02-05.04, Names and identifiers*	FND-946 The following Possible Problem has been noted: **Source:** WG3:HBA-042 = H2-2003-___ **Possible Problem:** This subclause says (regarding local temporary tables): Something needs to be said, in either or both of the Syntax Rules and General Rules of Subclause 5.4, "Names and identifiers", about how a <table name> identifies a created temporary table. As a minimum, a reference to a created local temporary table must be prohibited in any <schema routine> R, because, by the time R is invoked, it cannot be regarded as being syntactically contained in the <SQLclient module definition>	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					that contained the <externally-invoked procedure> which created it. **Solution** As a minimum, a Syntax Rule should be added, to the effect that: 1) If <table name> identifies a created local temporary table, then <table name> shall not be contained in a <schema routine>. Note: This appears to remove the need for any reference to created local temporary tables in Subclause 10.4, "<routine invocation>", General Rule 5) d) i), which is a problem for PSM.	
	NLD-P02-018		2-Minor Technical	*P02-06.01, <data type>*	FND-729 The following Language Opportunity has been noted: **Source:** WG3:YGJ-112 (SQL/MM YGJ-023), Paul Cotton for WG4, July 6, 1999, and Paul Scarponcini via email on 6 July 1999 **Language Opportunity:** According to YGJ-112: "REF types need to be scoped; i.e., the table(s) they refer to must be explicitly provided. If a column is of type REF type, the scope may be defined at table creation time. If the column is of type UDT which contains REF type attributes, then the scope must be declared when the UDT is created. The SQL/MM Part 3: Spatial standard defines the UDTs for spatial data. The standard is unable to predict in which tables the referenced information will be stored; this is a function of database design. Therefore, column scoping must be expanded to support deeply nested references, i.e., REF types within a UDT or ARRAY. This would allow a user, when creating tables, to define the scope of a UDTs REF type as part of the column definition for a column of type UDT." When a <reference type> is used as the data type of an attribute of a structured type, the <scope clause> must be specified when the encompassing user-defined type is defined. It is a Language Opportunity to be able to specify the <scope clause> of the "nested" <reference type>s when a column is defined on the encompassing user-defined type. Paul Scarponcini added: This applies to ARRAYs as well (e.g., an ARRY of REF, and ARRAY of UDTs having REF attributes. The resultant syntax may be quite messy, as different REFs within the column may have different scopes. Would it be worth considering reversing the scope specification: when the reference dtable is created, specify that it shall be included in the scope for a particular column, rahter than specifying the referenced table when the referencing column is specified? **Solution** None provided with comment.	
	NLD-P02-019		2-Minor Technical	*P02-06.01, <data type>*	FND-730 The following Language Opportunity has been noted: **Source:** WG3:YGJ-112 (SQL/MM YGJ-023) and Paul Cotton for WG4, July 6, 1999	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Language Opportunity: According to YGJ-112: "A second limitation of SQL 99 with respect to REF types is that they only achieve uni-directional "pointers"." A REF type value may be de-referenced to obtain the instance to which it refers. It is a Language Opportunity to provide direct support for determining all instances of a REF type which refer to a particular instance. **Solution** None provided with comment.	
	NLD-P02-020		2-Minor Technical	*P02-06.01, <data type>*	FND-812 The following Language Opportunity has been noted: **Source:** WG3:PER-098R1/H2-2001-059 **Language Opportunity:** Perhaps Feature S096, “Optional array bounds”, can be folded in Feature S091, “Basic array support”. **Solution** None provided with comment.	
	NLD-P02-021		2-Minor Technical	*P02-06.04, <value specification> and <target specification>*	FND-692 The following Language Opportunity has been noted: **Source:** DBL:CWB-081/X3H2-98-068 **Language Opportunity:** Although there is provision for refining a <value expression> of row type or structured type, there is no provision for refining a <target specification>. As a result, a field of a row or an attribute of a structured type cannot be passed as output or in/out argument of an SQL-invoked routine, or used in other target contexts. This problem is partially remedied in PSM <assignment statement>. Possibly the support for refined targets can be adapted from PSM and moved to Foundation. **Solution** None provided with comment.	
	NLD-P02-022		2-Minor Technical	*P02-06.04, <value specification> and <target specification>*	FND-723 The following Language Opportunity has been noted: **Source:** WG3:FRA-132/X3H2-98-694 **Language Opportunity:** Currently we have no capability to treat an <element reference> as a <target specification>. This precludes their use as output arguments of routine invocations, for example. The same observation can be made of <field reference>, <dereference operation>, <reference resolution>, and <method invocation> (some of these subject to the restriction that the method must be a mutator). (Lest you object that [Fred is] thinking of allowing surreptitious updates to column values by referencing them as output arguments of a routine invocation, be it noted that these expressions can also be used with parameters and variables.) However, [Fred believes] that the general solution to this problem is to introduce a notion of l-values and r-values, as in the specification of C.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	
	NLD-P02-023		1-Major Technical	*P02-06.06, <identifer chain>*	**WG3-P02-001** Consider the expression SELECT * FROM T T1 WHERE C1 = (SELECT MAX (C1) FROM T T2 WHERE T1.C2 > C1). It is surely indisputable that the two references to C1 in the subquery are syntactically legal and are references to T2.C1, according to the normal block-scoping rules that are commonly used in SQL implementations. And yet SR8) appears to make them illegal. SR8)a)ii) is applicable: ii) … [the <identifier chain>] shall be contained in the scope of one or more range variables whose associated tables include a column whose <column name> is equivalent to I_1 or in the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <SQL parameter name> is equivalent to I_1. Let the phrase *possible scope tags* denote those range variables and <routine name>s. In the example, C1 is contained in the scope of both T1 and T2. The continuation of this subrule is a Case whose first subrule is: 1) If the number of possible scope tags in the innermost scope containing a possible scope tag is 1 (one), then let *IPST* be that possible scope tag. Now, if this condition were true in our example, and the single possible scope tag were T2, then all would be well, but unfortunately that does not appear to be the case. The innermost scope containing a possible scope tag for C1 consists of two fragments: SELECT MAX (C1) and WHERE T1.C2 > C1 (see Subclause 7.6, "<table reference>", SR5). How many of the two possible scope tags for C1 are "in" this scope? If "in" means "contained in", then the answer appears to be one, but the one in question is T1 (contained in the <where clause>), not T2. If on the other hand "in" means "that are in scope in", then the answer is two, for both T1 and T2 are in scope. Of course, "in" is not intended to mean either of those things; in fact, it is clear under this close examination that "in the innermost scope" is not an appropriate phrase here at all. Having shown that "in the innermost scope" is not appropriate, we now show that "containing a possible scope tag" isn't appropriate either. Consider the following slightly simpler example: SELECT * FROM T WHERE C1 = (SELECT MAX (C1) FROM T). How many possible scope tags do we have now? Well, MAX (C1) is in the scope of the T that is defined in the outer <from clause> and it is also in the scope of the other T that is defined in the <subquery>'s <from clause>. Do we have two possible scope tags that are both named T, or do we have just one possible scope tag with two distinct reasons for it being a possible scope tag? In any case, whether we have one or two, how many are "in the innermost scope containing a possible scope tag"? The scope of the T defined in the <subquery> is just SELECT MAX (C1), which contains no possible scope tags at all. The scope of the T defined in the	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					outer <from clause> consists of the fragments SELECT * and WHERE C1 = (SELECT MAX (C1) FROM T), which happens to contain T, though not the T that has this scope! It seems that when the same range variable name is used for two or more different purposes (and necessarily in that case with different scopes in each case), and when a column reference lies within each of those scopes, only the one *having* the innermost of those scopes is applicable (and so that one is applied). And when two or more *different* range variables are used, as in our first example, then they are all applicable but it is again the one *having* the innermost scope that is applied, provided, of course, that there is exactly one range variable qualifying as a possible scope tag, whose scope is the innermost of the scopes containing the column reference. **Solution** None provided with comment.	
	NLD-P02-024		2-Minor Technical	*P02-06.09, <set function specification>*	FND-819 The following Language Opportunity has been noted: **Source:** WG3:PER-044R1/H2-2000-619 **Language Opportunity:** The proponents of multiargument GROUPING function believe that it is a trivial extension of the single argument function, and therefore does not warrant a separate feature. This could be achieved by simply deleting the Conformance Rule that creates Feature T433, “Multiargument GROUPING function”, thereby allowing all GROUPING functions to fall under Feature T431, “Extended grouping capabilities”. **Solution** None provided with comment.	
	NLD-P02-025		2-Minor Technical	*P02-06.12, <cast specification>*	**WG3-P02-002** SR10) prohibits the containment of a <collate clause> in the target <data type>, *TD*. When a <data type> is specified, this is clear, but it is not so clear when a <domain name> is specified, in which case SR1) defines *TD* to be “the <data type> of the domain”. The BNF for <domain definition> (Subclause 11.24) doesn’t include a <data type>, though it does include a <predefined type>. A domain descriptor is said to include a data type descriptor, but note carefully that *every* character data type descriptor contains the fully qualified name of a collation. Note that a <predefined type> might include a <collate clause>, and also that if the <domain definition> contains a <collate clause>, then that is considered to be equivalent to the containment of that <collate clause> in the <predefined type>. Some tidying appears to be needed, but what the intended rule is in the case of casting to a domain needs to be determined before a precise redrafting can be proposed. Note that a related problem exists in the SQL:2007 WD for SQL/XML, Subclause 6.4, <XML cast specification>, SR10), so this P.P. might eventually need to be cloned as a CD ballot comment against Part 14.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	

	NLD-P02-026		2-Minor Technical	*P02-06.12, <set function specification>*	FND-693 The following Language Opportunity has been noted:
					Source: FCD1/1998 NLD-P02-017, DBL:CWB-132/X3H2-98-187
					Language Opportunity:
					We do not understand SR 4). If an outer reference is permitted at all, surely it should be permitted any number of times, just as literals and host variable names can occur any number of times. We would add that we see no reason to prohibit outer references altogether. For example, if SUM(OUTER.C1) is legal, surely SUM(OUTER.C1+OUTER.C1) is also legal. Besides, why should column references that are not outer references be prohibited as soon as there is an outer reference? SR 4) of Subclause 6.9, "<set function specification>", says:
					4) The <value expression> simply contained in <set function specification> shall not contain a <set function specification> or a <subquery>. If the <value expression> contains a column reference that is an outer reference, then that outer reference shall be the only column reference contained in the <value expression>.
					We agree that the above rule is overly restrictive. However, we believe this rule was adopted in SQL-92 to prohibit query formulations of the form:
					`SELECT *`
					`FROM t1`
					`GROUP BY ...`
					`HAVING ... ( SELECT c21`
					`FROM t2`
					`GROUP BY ...`
					`WHERE ... ( SELECT c3`
					`FROM t3`
					`WHERE SUM ( t1.c12 + t2.c22 ) > ...`
					`)`
					`)`
					In the above example, outer references from multiple levels are being referenced in the same aggregate function. Semantically, this does not make sense and must be prohibited.
					Solution
					None provided with comment.	

	NLD-P02-027		2-Minor Technical	*P02-06.15, <subtype treatment>*	FND-816 The following Language Opportunity has been noted: **Source:** WG3:PER-099/H2-2001-061 **Language Opportunity:** Perhaps Feature S162, "Subtype treatment for references", can be folded into Feature S161, "Subtype treatment". **Solution** None provided with comment.	
	NLD-P02-028		2-Minor	*P02-06.15,*	FND-829 The following Language Opportunity has been noted:	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
			Technical	*<subtype treatment>*	**Source:** WG3:PER-186/H2-2001-??? **Language Opportunity:** WG3:PER-099 extended <subtype treatment> so that an expression of type REF(*t1*) would be TREATed as one of type REF(*t2*) if *t2* is a subtype of *T1*. It was noted that, in that case, it should also be possible to TREAT: — An expression of type *t1* ARRAY[*n*] as one of type *t2* ARRAY[*n*]. — An expression of type *t1* MULTISET as one of type *t2* MULTISET. — An expression of type ROW(..., *f1 t1*, ...) as one of type ROW(..., *f1 t2*, ...). In the ROW case, it might even be possible to support TREATment over more than one field. For example, an expression of the type ROW(..., *f1 t1*, ..., *f2 t1*, ...) might be TREATable as ROW(..., *f1 t1*, ..., *f2 t2*, ...), as ROW(..., *f1 t2*, ..., *f2 t1*, ...), or as ROW(..., *f1 t2*, ..., *f2 t2*, ...), even though SQL does not (at the time of writing this Language Opportunity) support multiple inheritance in general. In the ROW case, it would also be necessary to decide whether field names must match as indicated in these examples. **Solution** None provided with comment.	
	NLD-P02-029		2-Minor Technical	*P02-06.28, <string value expression>*	FND-858 The following Language Opportunity has been noted: **Source:** WG3:ICN-054R2 = H2-2002-___ **Language Opportunity:** The term "character string operands" was used to replace a previously undefined term "components" in SR2. Is this the correct terminology to use? **Solution** None provided with comment.	
	NLD-P02-030		2-Minor Technical	*P02-06.34, <boolean value expression>*	FND-920 The following Language Opportunity has been noted: **Source:** WG3:ZSH-129 = H2-2002-___ **Language Opportunity:** The rules for known-not-null conditions in SR3) are more complicated than most implementations are prepared to implement, and not necessary for most users. The full implementation of known not null should be placed in a conformance feature. Without the feature, a much simpler definition should apply. **Solution** None provided with comment.	
	NLD-P02-031		2-Minor Technical	*P02-06.35, <array value expression>*	FND-808 The following Language Opportunity has been noted: **Source:** (was Possible Problem FND736) WG3:PER-171/H2-2001-??? (FCD1/2000 NLD-P02-027), from WG3:YGJ-074/X3H2-99-164R1 **Language Opportunity:** The ability to extract a subarray of an array would be useful. Such an ability would also satisfy a separate Language Opportunity to be able to truncate an array.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	
	NLD-P02-032		2-Minor Technical	*P02-07.04, <table expression>*	FND-756 The following Language Opportunity has been noted: **Source:** WG3:YGJ-069r1 = H2-99-155r3 and WG3:BHX-096/H2-2000-248R1 **Language Opportunity:** It might be useful to be able to filter windowed results based on the values of <OLAP function>, most likely through a new clause analogous to <where clause> and <having clause>, but following <window clause> **Solution** None provided with comment.	
	NLD-P02-033		2-Minor Technical	*P02-07.09, <group by clause>*	FND-610 The following Language Opportunity has been noted: **Source:** DBL:LGW-146/X3H2-97-349 **Language Opportunity:** Continuing work is needed to complete object support as outlined in "Providing Rich Query Functionality" (DBL:LHR-078 = X3H2-95-462) with regard to expanding GROUP BY to permit naming of grouping expressions and allowing those names to be used in the query. The ability to group the result of a table expression by the value of expressions is important to many applications. The ability to name these grouping expressions and use those names to retrieve the results of the grouping column cum expression in the select list of the table expression is equally important to avoid applications having to repeat the expression (giving opportunity for errors) in the select list. **Solution** None provided with comment.	
	NLD-P02-034		2-Minor Technical	*P02-07.12, <query specification>*	FND-528 The following Language Opportunity has been noted: **Source:** DBL:MAD-170/X3H2-96-544R1, point 2.1, FCD1/1998 CAN-P02-031, DBL:CWB-132/X3H2-98-187 **Language Opportunity:** DBL:MAD-170/X3H2-96-544R1, point 2.1, noted: The definition of a possibly nullable result column in the Syntax Rules of Subclause 7.12, "<query specification>", is broader than necessary, in that an aggregate of a column that is known not nullable is regarded as possibly nullable. For example, SUM(EMP.EMPNO) is defined as possibly nullable, even if EMP.EMPNO is declared NOT NULL. DBL:CWB-132/X3H2-98-187 added: The problem description makes the assumption that a <set function specification>, for example SUM(EMPNO), is known not nullable when EMPNO is known not nullable. However, GR 3)b) of Subclause 6.9, "<set function specification>", makes it clear that (with the exception of COUNT) <set function specification>s return NULL when they are applied to an empty table. Hence, we assume that <set function specification>s are possibly nullable,	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					except for COUNT. And, that is what SR 12) of Subclause 7.12, "<query specification>", specifies. Hence, we believe that there is no problem with SR 12) of Subclause 7.12, "<query specification>". **Solution** None provided with comment.	
	NLD-P02-035		2-Minor Technical	*P02-07.12, <query specification>*	FND-908 The following Language Opportunity has been noted: **Source:** P02, SQL/Foundation, Subclause 7.12, "<query specification>", CR 4) and CR 8) **Language Opportunity:** Conformance Rule 4) as formulated does not impose a restriction on the user writing SQL and as such does not follow the required model for Conformance Rules. Fred Zemke in an email to Stephen Cannan dated 2002-10-17 wrote: Subclause 7.12, "<query specification>" CR 4) - this is an example of the occasional practice of using the CRs to alter the definition of a defined term. This practice seems borderline to me. On the one hand, the CRs are regarded as merged with the SRs whenever the designated feature is absent, and definitions appear in the SRs, so it would seem possible to make a redefinition in a CR. On the other hand, does a redefinition constitute a limitation on the user? I think the better approach is the one taken regarding functional dependencies, for example, Subclause 7.12, "<query specification>", CR 3). This could have been done by defining a term such as 'group-invariant <value expression>' in the SRs, saying that all derived columns in the SELECT list of a grouped query must be group-invariant <value expression>s, and then the CR would alter the definition of groupinvariant <value expression>. Instead the approach taken is essentially to carve out two categories of derived column: the kind permitted in the SELECT list of a grouped query by the SRs, and the more restricted kind permitted by the CRs. Returning to 7.12 CR 4), the path would be to define two notions of updatable, using the more liberal one in the SRs and restricting to the more conservative one in the CRs. In fact, we already have two terms, updatable and simply updatable. The conclusion I am coming to is that this CR should be deleted, and, in any subclause that uses 'updatable' in an SR, there should be a CR that restricts to 'simply updatable' unless Feature T111 is present. Conformance Rule 8) as formulated does not impose a restriction on the user writing SQL and as such does not follow the required model for Conformance Rules. This rule should be deleted, and, in any subclause that uses 'updatable' in an SR, there should be a CR that restricts the use of UNION unless Feature T111 is present. **Solution** None provided with comment, but the body of the comment outlines a solution..	See Comment
	NLD-P02-036		2-Minor Technical	*P02-07.15, <subquery>*	FND-936 The following Language Opportunity has been noted: **Source:** WG3:HBA-050R1	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Language Opportunity: HBA-050 shows that the need for the BNF term <subquery> is nothing like as strong as it once might have been, and has given rise to a certain amount of difficulty and confusion. Perhaps it would be better to dispose of the term altogether (though <scalar subquery>, <row subquery>, and <table subquery> almost certainly need to be retained) and treat parenthesized <query expression>s in similar style to our treatment of parenthesized <value expression>s. Any proposal to address this Language Opportunity should of course check for existing uses of <subquery> in Parts other than Foundation. **Solution** None provided with comment.	
	NLD-P02-037		2-Minor Technical	*P02-08.02, <comparison predicate>*	FND-909 The following Language Opportunity has been noted: **Source:** WG3:ZSH-155 = H2-2002-___ **Language Opportunity:** The Syntax Rules convert all comparison predicates so that they only use < and =. The GRs for comparison of user-defined types spell out rules for > and other comparisons even though they have been transformed away. NOTE 167 following the GR claims that these unreachable GRs are there for informational purposes. In the case of RELATIVE order, there are some strong assumptions being made that *RF*(*X*,*Y*) = -*RF*(*Y*,*X*); otherwise, the system breaks down. We should document what are the expectations for the relative order function somewhere. We do not find such documentation either in <user-defined ordering function> or in Concepts. **Solution** None provided with comment.	
	NLD-P02-038		2-Minor Technical	*P02-09.03, Data types of results of aggregations*	FND-836 The following Language Opportunity has been noted: **Source:** WG3:YYJ-030R2 = H2-2001-___ and WG3:ZSH-155 = H2-2002-___ **Language Opportunity:** This subclause uses terms that are less precise than they should be. Specifically, the term result data type and data type of the result, without specifying the result of what. The first sentence of Function says: "Specify the result data type of the result of an aggregation ...". Moreover the term aggregation does not suggest the sense in which it is used here, having since been used extensively in the context of OLAP, see subclause 04.17.03 "Aggregate functions". A better title would be Data types of results of n-adic operations. Were this title adopted, the first sentence could be rewritten as, for example, Let IDTS be a set of data types specified in an application of this Subclause, and let O be the operation. **Solution** None provided with comment.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
	NLD-P02-039		2-Minor Technical	*P02-09.05, Type precedence list determination*	FND-709 The following Language Opportunity has been noted: **Source:** WG3:YGJ-021 **Language Opportunity:** Paper DBL:BBN-168 added a Syntax Rule to Subclause 11.50, "<SQL-invoked routine>", to prohibit the use of ROW because there is nothing in Subclause 9.5, "Type precedence list determination", to handle the type precedence requirements of anonymous row types. **Solution** None provided with comment.	
	NLD-P02-040		1-Major Technical	*P02-10.04, <routine invocation>*	FND-857 The following Possible Problem has been noted: **Source:** DCOR/2002, USA-STC-031 **Possible Problem:** There is no definition of how to pass booleans or LOBs to external programs. More generally, the question of how to convert any SQL type to a host language type at the interface to an SQL-invoked routine has never been addressed. Probably it was assumed that the same mechanism as was already defined for module language and embedded language applied, but in fact there are no rules to back up this assumption. If this assumption is correct, then the rules in Subclause 13.4, "Calls to an <externally-invoked procedure>", are probably appropriate. Perhaps they should be placed in a separate subclause so they can be referenced by both <routine invocation> and also <externally invoked procedure>. See also paper WG3:PER-176. **Solution** None provided with comment.	
	NLD-P02-041		1-Major Technical	*P02-10.04, <routine invocation>*	FND-956 The following Possible Problem has been noted: **Source:** WG3:ZSH-037R1/H2-2003-??? **Possible Problem:** 5) Preserve the current SQL-session context CSC and create a new SQL-session context RSC derived from CSC as follows: ... This appears to specify what happens to every element of an SQL-session context when a new SQL-session context is created. However, it does not say what happens to: — The zero or more trigger execution contexts — The values of all valid locators — The text defining the SQL-path (which in any case seems somewhat redundant, since the SQL-path is taken care of) — The SQL-session collations, if any — The text defining the default transform group name — The text defining the user-defined type name-transform group name pair for each userdefined type explicitly set by the user It would at least be clearer if it said:	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					5) Preserve the current SQL-session context CSC and create a new SQL-session context RSC as follows: ... **Solution** None provided with comment.	
	NLD-P02-042		2-Minor Technical	*P02-11, Schema definition and manipulation*	FND-710 The following Language Opportunity has been noted: **Source:** WG3:YGJ-021 **Language Opportunity:** A RENAME TABLE statement has been strongly desired for a very long time and any users will be expecting to see it in SQL3. **Solution** None provided with comment.	
	NLD-P02-043		2-Minor Technical	*P02-11, Schema dewfinition and manipulation*	FND-694 The following Language Opportunity has been noted: **Source:** DBL:CWB-114/X3H2-98-169 **Language Opportunity:** The current choices for <drop behavior>, RESTRICT and CASCADE, are too limiting. CASCADE is so sweeping that the user must hesitate to use it, not knowing what may be dropped. RESTRICT, on the other hand, is so limited that the user must find all dependencies and drop them in the proper order. There is a third model, based on the notion of invalidation. With this model, a dependent definition does not block a drop; instead, the dependent object is simply marked invalid. Later usage of an invalid object causes its recompilation, which may very well succeed since the cause of invalidation may have been repaired. **Solution** None provided with comment.	
	NLD-P02-044		2-Minor Technical	*P02-11.03, <table definition>*	FND-822 The following Language Opportunity has been noted: **Source:** WG3:PER-104/H2-2001-085R1 **Language Opportunity:** The ability to specify options for inheriting column default and identity column properties, as in the <like clause>, would also be beneficial for the <as subquery clause>. **Solution** None provided with comment.	
	NLD-P02-045		2-Minor Technical	*P02-11.03, <table definition>*	FND-874 The following Language Opportunity has been noted: **Source:** WG3:DRS-095 **Language Opportunity:** Since in section 1.1.2 [of WG3:DRS-095] we gave reasons for determining the <reference generation> implicitly, it would be most convenient if the <column constraint definition>s necessary for derived reference representations were implicit, and determined by examination of the corresponding user-defined type descriptor.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					Solution None provided with comment.	
	NLD-P02-046		2-Minor Technical	*P02-11.05, <default clause>*	FND-642 The following Language Opportunity has been noted: **Source:** DBL:LGW-152/X3H2-97-352 (also DBL:LGW-023/X3H2-97-044, SEQ# 222, CAN-F-062, converted to LO by WG3:BHX-038/H2-2000-018R3) **Language Opportunity:** It might be useful to allow default values for row types, perhaps by using row constructors. **Solution** None provided with comment.	
	NLD-P02-047		2-Minor Technical	*P02-11.05, <default clause>*	FND-712 The following Language Opportunity has been noted: **Source:** WG3:YGJ-021 and WG3:PER-098R1/H2-2001-059 **Language Opportunity:** It is not possible to specify default values for columns or attributes of an array type, a multiset type, a reference type, a row type, or a user-defined type. **Solution** None provided with comment.	

	NLD-P02-048		2-Minor Technical	*P02-11.08, <referential constraint definition>*	FND-349 The following Language Opportunity has been noted:
					Source: WG3:YGJ-074/X3H2-99-164R1 (Bill Kelley noted the following Language Opportunity, which has been modified by Fred Zemke)
					Language Opportunity:
					For collections types, referential integrity is not definable for elements of collections.
					Example: Assume table EMPLOYEE has PRIMARY KEY EMP_ID of type INTEGER:
					`CREATE TABLE MANAGER (`
					`EMPNO INTEGER,`
					`MANAGES INTEGER ARRAY[20] )`
					Here "MANAGES" refers to a set of employees, but there is no way to say that they should reference employees. That is, if one were to write:
					`CREATE TABLE MANAGER (`
					`EMPNO INTEGER,`
					`MANAGES INTEGER ARRAY[20] REFERENCES EMPLOYEE )`
					then EMPLOYEE.EMPNO must be a column of array type, and teh constraint says that the array value in MANAGER.MANAGES must either be null or be equal to an array value in EMPLOYEE.EMPNO.
					What is needed is a new syntax, perhaps:
					`CREATE TABLE MANAGER (`
					`EMPNO INTEGER,`
					`MANAGES INTEGER ARRAY[20] ELEMENT REFERENCES EMPLOYEE(EMPNO) )`
					ELEMENT REFERENCES would mean that each array element of MANAGER.MANAGES must either be null or equal value in	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					EMPLOYEE.EMPNO. **** Editor's Note (number 15) **** (Editor's note: In my opinion, Bill is simply trying to solve the problem using the wrong tools. INTEGER ARRAY[n] is meant to have elements of integers, not elements of employee IDs...which is a different thing altogether.) **Solution** None provided with comment.	
	NLD-P02-049		2-Minor Technical	*P02-11.10, <alter table statement>*	FND-747 The following Language Opportunity has been noted: **Source:** WG3:RTM-028/X3H2-99-252R1 **Language Opportunity:** It might be useful to have an option so that a conventional (SQL-92) table can evolve to become a table of type. However, any such proposal must avoid the pitfalls noted during development of SQL:1999 for evolution to a table of "named row type" (to use the terminology current before structured types were introduced). The proposal must account for the <reference type specification> of the user-defined type. If <reference generation> is DERIVED, it may be necessary to require a unique constraint or primary key constriant on the appropriate columns. If <references generation> is USER GENERATED, it may be necessary to require that the table has no rows. Probably the self-referencing column must be added to the table as part of its evolution to a table of structured type. It is unlikely that the unaltered table will have as its first column a reference to the very type to which the table will be evolving. And, if perchance that condition were met, what would be do with the previously existing values in that column? **Solution** None provided with comment.	

	NLD-P02-050		1-Major Technical	*P02-11.22, <view definition>*	FND-933 The following Possible Problem has been noted:
					Source: WG3:HBA-050R1
					CR4) of this subclause is:
					4) Without Feature F751, “View CHECK enhancements”, conforming SQL language shall not contain <view definition> that contains a <subquery> and contains CHECK OPTION.
					This is suspect. Even if it really was intended to rule out, as it apparently does, examples of the following form (note the <subquery>):
					`CREATE VIEW V AS`
					`SELECT ...`
					`FROM ( SELECT ... FROM ... ) AS T`
					`WHERE ...`
					`WITH CHECK OPTION`
					then surely it should also be ruling out examples of the following equivalent form:
					`CREATE VIEW V AS`	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					WITH A AS (SELECT ... FROM ...) SELECT ... FROM A WHERE ... WITH CHECK OPTION But it doesn't. (Note the lack of any <subquery>.) A minor additional point is that "WITH CHECK OPTION" would be safer than just "CHECK OPTION", in case WITHOUT CHECK OPTION is ever added to the language. **Solution** None provided with comment.	
	NLD-P02-051		1-Major Technical	*P02-11.30, <drop domain statement>*	FND-938 The following Possible Problem has been noted: **Source:** WG3:HBA-028 **Possible Problem:** GR1)c) refers to "the explicit or implicit <constraint name list>". The BNF production for <drop domain definition> does not included a <constraint name list>, nor do the Syntax Rules specify an implicit one in any circumstances. **Solution** Delete GR1)c) and edit the lead-in of GR1)d) as shown here: d) For every domain constraint descriptor included in the domain descriptor of D [begin deletion] whose <constraint name> is not contained in the excluded constraint list[end deletion]:	
	NLD-P02-052		2-Minor Technical	*P02-11.39, <trigger definition>*	FND-611 The following Language Opportunity has been noted: **Source:** DBL:LGW-146/X3H2-97-349 **Language Opportunity:** SQL3 should consider adding syntax to allow the user to specify the ordering in which triggers on the same effect should be fired. **Solution** None provided with comment.	
	NLD-P02-053		2-Minor Technical	*P02-11.41, <user-defined type definition>*	FND-603 The following Language Opportunity has been noted: **Source:** DBL:LGW-131/X3H2-97-293, 24 July, 1997; also USA-081 in first CD ballot for SQL/Foundation and WG3:YGJ-074/X3H2-99-164R1 **Language Opportunity:** Subclause 11.41, "<user-defined type definition>", contains a Syntax Rule reading: 6)g) [A user-defined type] shall not be based on itself. This syntax rule prevents the UDT facility from modeling a recursively-defined data type such as "Tree". Here is a simple example of a UDT definition that is not possible because of that SR: CREATE TYPE Tree (node_value INTEGER,	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By

					`left_subtree Tree,`
					`right_subtree Tree )`
					Solution
					None provided with comment.	

	NLD-P02-054		2-Minor Technical	*P02-11.50, <SQL-invoked routine>*	FND-713 The following Language Opportunity has been noted: **Source:** WG3:YGJ-021 **Language Opportunity:** Currently all parameters must be of some specific concrete type. There needs to be a mechanism to declare that a parameter is a character string of arbitrary, unspecified type, at least when invoking PSM. (And there should be some mechanism within PSM to interrogate the character set and length of a character string parameter). Otherwise the subject routine rules allow you to resolve to the same PSM routine no matter what the parameter's character set, but when the function is invoked, you will get an error when trying to assign the input argument to the parameter's type if the input argument's character set is different from the one declared in the function's signature. There should also be a mechanism to declare that the return type of a function is determined by a parameter's type. **Solution** None provided with comment.	
			1-Major Technical	*P02-12.01, <grant statement>*	**WG3-P02-003** GRs 4)b), 4)c), 4)d), 4)e), 5), 6), though curiously not 7) all contain the phrase "[f]ollowing the successful execution of the <grant statement> …". Given that GRs are to be evaluated in the order in which they are written, and that "successful execution" usually (though not always) means that the last GR has been reached, that wording doesn't seem to make sense. Note that the BNF for <grant statement> specifies that it is either a <grant privilege statement> or a <grant role statement>. The rule evaluation order specified in Framework, Subclause 6.3.3.3, makes it clear that the rules for the contained statements are applied before the rules for the containing statement. Perhaps, then, the wording we have questioned should be changed to something to the effect of "following the successful execution of the contained statement"; but if the contained statement fails, then doesn't evaluation of the rules of the containing statement end too? In that case, each of these phrases can simply be deleted. Note also that GRs 1), 2), 3), 4) 5), 6) and 7) all specify the execution of "the following <grant statement> …". Whoever drafts a solution to this problem might like to check that there is no infinite recursion going on here. It might be that "the following <grant statement>" should better be "the following <grant privilege statement>" or "the following <grant role statement>", as applicable. **Solution** None provided with comment.	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
	NLD-P02-055		2-Minor Technical	*P02-12.07, <revoke statement>*	FND-734 The following Language Opportunity has been noted: **Source:** Email from Fred Zemke, 1999-06-09 and WG3:ZSH-155 = H2-2002-___ **Language Opportunity:** The OLAP Amendment has created a new kind of dependency, of a view, *etc.*, containing an OLAP function that references a user-defined ordering in its ORDER BY clause, which is dependent on the userdefined ordering. <drop routine statement> has been edited to account for this dependency; does any other statement need to be edited? **Solution** None provided with comment.	
	NLD-P02-056		2-Minor Technical	*P02-12.07, <revoke statement>*	FND-911 The following Language Opportunity has been noted: **Source:** WG3:ZSH-155 = H2-2002-___ **Language Opportunity:** Syntax Rule 36) says: 36) If RESTRICT is specified, then there shall be no abandoned privilege descriptor, abandoned view, abandoned table constraint, abandoned assertion, abandoned domain constraint, lost domain, lost column, lost schema, and no descriptor that includes an impacted data type descriptor, impacted collation, impacted charater set, abandoned user-defined type, forsaken column decriptor, forsaken domain descriptor, or abandoned routine descriptor. This SR has several problems: — It is unclear whether there should be a comma following "schema", though we recognize that a schems is a descriptor. (Note: This problem has been fixed by the addition of "and no" between "schema," and "descriptor".) — It is unclear whether the object of "includes" is a nested list. (Note: This problem has been resolved by making it clear that it is a nested list.) — The terms used to refer to impacted, etc., objects are inconsistent with those used to so designate them. While it is **descriptors** that are said to be abandoned, impacted, etc., this rule referes to "impacted **columns**", etc. — Several possible candidates for inclusion in the list are absent for no obvious reason; they include abandoned table descriptor, abandoned trigger descriptor, and contaiminated column descriptor. We suggest improving the clarity by using a possibly nested bullet list. **Solution** None provided with comment.	
	NLD-P02-057		1-Major Technical	*P02-12.07, <revoke statement>*	FND-979 The following Possible Problem has been noted: **Source:** WG3:SIA-018 = H2-2003-429 **Possible Problem:** WG3:SIA-018 introduced the notions of view components, view component privilege descriptors, and view privilege dependency descriptors pertaining to a	

SEQ #	Cmnt ID	See Also	Severity	Reference	Description	Addressed By
					given view. WG3:SIA-018 proposed explicit rules that specify the creation of view privilege dependency descriptors, but failed to specify explicit rules that specify the destruction of view privilege dependency descriptors. Although a view privilege dependency descriptor can be assumed to be destroyed whenever either its supporting privilege descriptor or the dependent privilege descriptor is destroyed, the standard would be clearer if this were done in the appropriate place(s) in the GRs of Subclause 12.7, "<revoke statement>". **Solution** None provided with comment.	
	NLD-P02-058		2-Minor Technical	*P02-13.01, <SQL-client module definition>*	FND-921 The following Language Opportunity has been noted: **Source:** FCD1/2002, GBR-P02-485 **Language Opportunity:** None of the GRs in this Subclause relate to the creation of an SQL module. Moreover, General Rule 4) relates to the invocation of an externally-invoked procedure. **Solution** None provided with comment.	
	NLD-P02-059		2-Minor Technical	*P02-13.03, <externally-invoked procedure>*	FND-844 The following Language Opportunity has been noted: **Source:** WG3:YYJ-034 = H2-2001-___ **Language Opportunity:** The use of savepoint levels, introduced by WG3:PER-061 and extended by WG3:YYJ-034, still does not cover the case of externally-invoked procedures. **Solution** None provided with comment.	
	NLD-P02-060		2-Minor Technical	*P02-13.05, <SQL procedure statement>*	FND-925 The following Possible Problem has been noted: **Source:** WG3:HBA-029 **Possible Problem:** Subclause 13.5, "<SQL procedure statement>", includes two GRs to the effect that if the statement being executed is an atomic one, then all changes to SQL-data and schemas are cancelled. Shouldn't this be conditional on whether the current execution context is atomic, rather than on the statement type? Not that this would make any material difference, but as things stand there appears to be no point in the final sentence of GR2) of this Subclause: 1) 2) A statement execution context NEWSEC is established for the execution of S. Let OLDSEC be the most recent statement execution context. NEWSEC becomes the most recent statement execution context. NEWSEC is an atomic execution context, and therefore the most recent atomic execution context, if and only if S is an atomic SQL-statement. Although there are GRs in various subclauses that do enforce atomicity where it is required, none of these rules references the atomicity or non-atomicity of an execution context.	