Rdb Journal
technical corner

Guide to SQL Programming: SQL:1999 and Oracle Rdb V7.1

A feature of Oracle Rdb

By Ian Smith
Oracle Rdb Relational Technology Group
Oracle Corporation

The Rdb Technical Corner is a regular feature of the Oracle Rdb Web Journal. The examples in this article use SQL language from Oracle Rdb V7.1 and later versions.

SQL:1999 and Oracle Rdb V7.1[1]

After many years of design and review the Database Language SQL:1999 was ratified in late 1999 and is now the current SQL Language standard across the world. This standard is published by ANSI in the U.S.A. and by ISO internationally. Officially there no longer exists a SQL:92 standard. How does Oracle Rdb V7.1.2 stand in the light of this new SQL language standard?

Background

SQL:86 and SQL:89 were published as a complete language and vendors were expected to implement the specification, with only few exceptions such as optional language bindings. For instance, if the product didn't support the MUMPS language then those bindings were not required.

When SQL:92 was published in late 1992 it was such a large set of changes that the committee assumed (and rightly so) that not all vendors would meet this standard for many years. It was broken down into three classes; ENTRY, INTERMEDIATE and FULL. Each class a superset of the previous.

SQL:92 ENTRY was very similar to SQL:89 with only a handful of additional features, such as SQLSTATE support, delimited identifiers, column renaming with AS clause, and altered syntax for module language procedure header. Therefore, many vendors were early adopters of SQL:92 ENTRY level.

Rdb V6.0 delivered the changes in syntax and semantics and was certified as SQL:92 ENTRY level conformant. However, the Rdb product actually included many features that exceeded ENTRY level and was closer to SQL:92 INTERMEDIATE. For example, the product included domain definitions, LEFT, RIGHT and FULL OUTER JOIN support, DATE, TIME, TIMESTAMP, INTERVAL data types, and many semantics associated with nested subqueries. The product also

[1] This is a revised and updated version of a similarly titled article released in September, 2001.

included many features that were described in SQL:92 FULL such as CROSS JOIN, derived tables, and CHARACTER SET support.

The SQL:92/PSM (Persistent Stored Modules) package was added in 1996. This package defined external and stored routines and many procedural language features such as compound statements (BEGIN/END); FOR, LOOP and WHILE looping statements; IF, CASE and LEAVE conditional statement; DECLARE, SET assignment statements, and the CALL statement. When PSM was published Rdb already had an implementation with more than 90% of the specification.

Since Rdb V6.0 Oracle has continued to add new standard SQL syntax to the product during every release cycle. For instance Oracle Rdb7 added global, and local temporary tables, WITH HOLD cursors, SQL stored functions, external procedures, and new procedural language syntax.

The Current SQL Standard.

Now that SQL:1999 has been published and SQL:92 has been superseded, we in Rdb are getting more inquiries about SQL:1999 compliance.

Although over 7 years have passed since SQL:92 was published, there has been no database product certified as INTERMEDIATE or FULL level. There was obviously a demand for specific pieces of the SQL standard (e.g. character set support) or for language features not covered in the published standard (e.g. User Defined Data types, and object oriented development).

Therefore, the SQL:1999 standard is now presented as a set of CORE features with predefined packaged of features rather than the nested feature sets in SQL:92. Database products now must implement CORE and then select various packages as required by their customer base.

Getting From SQL:92 ENTRY to SQL:1999 CORE

Using information provided by our Oracle SQL standards committee representative, as well as reading parts of the current SQL:1999 specification I have prepared a list of features that are now part of the standard SQL that were added since SQL:92 ENTRY level. Oracle Rdb will require these features for SQL:1999 CORE compliance. Any errors or omissions in this list are probably mine.

Implicit numeric casting: the ability to perform store and retrieval assignments between different numeric types.

Rdb has always supported this functionality.

Implicit character casting: the ability to assign different character types.

Rdb has always supported this functionality.

VARCHAR data type: This also includes functions such as SUBSTRING, TRIM, POSITION, CHAR_LENGTH, etc.

Rdb has supported VARCHAR since V1.0. SUBSTRING was added in V3.1, CHAR_LENGTH and OCTET_LENGTH in V4.2, TRIM and POSITION in V6.1. V7.1 adds the CHARACTER VARYING syntax as a synonym for VARCHAR.

UPPER and LOWER functions.

Rdb has supported these since V4.2

Lowercase or mixed case identifiers. Also known as delimited identifiers.

Rdb has supported these since V4.2

Trailing underscore: the ability to use an underscore as the last character in an identifier.

Rdb has always supported this functionality.

Qualified asterisk in SELECT list: For example,
SELECT E.* FROM EMPS E. The qualification of * was not permitted by SQL:92.

Rdb has supported this since the initial versions of SQL.

Rename columns in the FROM clause.

Rdb added the AS clause in Rdb V6.0

EXCEPT DISTINCT: This operator is syntactically similar to the UNION operator. The result is the set difference of the two table expressions.

Rdb has supported this feature since Rdb V7.1.0.2.

UNION and EXCEPT in which matching columns have different data types.

Rdb supports data type coercion for UNION and EXCEPT.

UNION and EXCEPT permitted in subqueries.

Rdb supports UNION and EXCEPT in subqueries.

Complex expressions in VALUES clause.

Rdb has supported this since the earliest releases of SQL.

Value expressions in ORDER BY clause.

Rdb has supported this since V7.0.1.

WITH HOLD cursors. This feature allows cursors to remain open across a COMMIT statement.

Rdb has supported this feature with extensions since V7.0. The Rdb support optionally allows HOLD cursors across ROLLBACK.

LOCALTIME and LOCALTIMESTAMP as column default options.

Rdb V7.1 supports these value expressions as defaults.

NOT NULL inferred for PRIMARY KEY. SQL:92 ENTRY required that NOT NULL also be specified.

Rdb supports this feature.

Names in foreign key can be specified in any order. For instance, if the PRIMARY KEY of a table is defined as (PART_CLASS, PART_ID) then a FOREIGN KEY can specify the mapping as (PART_ID, PART_CLASS). SQL:92 ENTRY insisted that the columns names be in the same order as the PRIMARY KEY or UNIQUE constraint that is referenced.

Rdb V7.1 supports this functionality. The constraints must be defined when the DIALECT is set to either 'SQL99' or 'ORACLE LEVEL2'.

WORK is now an optional keyword. SQL:92 ENTRY required COMMIT WORK and ROLLBACK WORK.

I think every vendor allows this, Rdb is no exception.

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE.

Rdb supports this, and also the other isolation levels: READ COMMITTED, and REPEATABLE READ.

SET TRANSACTION READ ONLY and READ WRITE.

Rdb supports this syntax. However, in Rdb SET TRANSACTION is a transaction initiating statement. SQL:1999 CORE assumes that the first executable statement will cause the transaction to be started. SQL:1999 provides a START TRANSACTION statement to initiate a transaction.

Oracle Rdb V7.1 supports START TRANSACTION. However, SET TRANSACTION is not a passive attribute establishing statement.

Queries with subqueries may be updatable.

Rdb already supports this feature in views and cursors.

Information Schema, Documentation Schema, etc.

Special tables that describe the schema objects. These tables look quite different from the Rdb system tables but provide essentially the same information. Rdb does not support this feature.

Basic Schema Definition and manipulation: This includes CREATE SCHEMA, CREATE TABLE, CREATE VIEW, GRANT, ALTER TABLE … ADD column, DROP TABLE, DROP VIEW, and REVOKE.

Rdb supports these features. The one exception is the REVOKE … RESTRICT syntax.

Basic joined tables: Inner joins, right and left outer joins, with full nesting with arbitrary comparison operators.

Rdb has supported this feature since V6.0.

Basic date and time support: DATE, TIME and TIMESTAMP data types, without time zone support. CAST support, and built-ins LOCALTIME and LOCALTIMESTAMP.

Rdb has mostly supported these features since V4.1. There is just one exception: fractional seconds precision for TIME and TIMESTAMP should allow 6 digits, Rdb supports at most 2.

Grouped operations: This really just allows the GROUP BY, and HAVING clauses in views, aggregation and nested subqueries.

Rdb has supported this feature since V3.1.

Multiple module support: The ability to link several embedded compilation units to form a single executable.

Rdb has always supported this functionality.

CAST function: the ability to explicitly convert between different data types.

Rdb has supported this functionality since Rdb V4.1.

Explicit defaults: the keyword DEFAULT can be used to assign default values in an INSERT statement or UPDATE SET clause.

Rdb V7.1 supports this functionality.

CASE expressions: These include searched CASE, simple CASE, NULLIF and COALESCE expressions.

Rdb has supported this functionality since Rdb V6.0.

Scalar subquery values:

Rdb already supports this functionality.

Value expression in IS NULL predicate.

Rdb already supports this functionality.

Basic flagging: this flagger should alert the user of any vendor extensions, or standard features not in SQL:1999 CORE.

Rdb implements a SQL:92 flagger, but not a SQL:1999 flagger.

Distinct data types: This is very similar to the DOMAIN support already in Rdb, but value assignment is strictly enforced.

Rdb does not support this functionality.

SQL-invoked routines.

Rdb has supported this functionality since V6.0. This support was for external functions and SQL stored procedures. Rdb V7.0 supported external procedures and SQL stored functions. This also includes the CALL and RETURN statements.

Oracle Rdb and CORE compliance with SQL:1999

Oracle Rdb V7.1 continues to add new features many drawn from the SQL:1999 database standard. These include:

- START TRANSACTION statement
- ITERATE loop control statement
- WHILE (revised to be SQL:1999 PSM syntax)
- REPEAT looping statement
- searched CASE statement
- DETERMINISTIC, NOT DETERMINISTIC, RETURNS NULL ON NULL INPUT, and CALLED ON NULL INPUT clauses for functions
- support for module global variables
- INSERT with DEFAULT VALUES clause
- DEFAULT keyword for INSERT and UPDATE
- full SIGNAL statement syntax
- LOCALTIME, LOCALTIMESTAMP and ABS built in functions
- BETWEEN SYMMETRIC predicate support
- USER and ROLE support including the GRANT/REVOKE enhancements
- INITIALLY IMMEDIATE and INITIALLY DEFERRED clauses for constraints,
- UNIQUE predicate
- TABLE query specification (short hand for SELECT * FROM)
- DISTINCT keyword for UNION
- FOREIGN KEY reference semantics
- ALTER MODULE, ALTER PROCEDURE and ALTER FUNCTION commands
- COMMIT AND CHAIN statement
- ROLLBACK AND CHAIN statement
- EXCEPT DISTINCT operator
- INTERSECT DISTINCT operator
- VAR_POP, VAR_SAMP, STDDEV_POP and STDDEV_SAMP statistical functions
- FILTER clause for statistical expressions

The following list describes SQL:1999 CORE features not currently in Oracle Rdb, some of which are planned for future releases

SET TRANSACTION as an attribute change statement.

The SQL standard defines this statement as affecting the settings for the next transaction. In Rdb this statement actually starts a transaction. Therefore, the following legitimate SQL-standard statement sequence would not behave in a standard fashion if executed on the current version of Rdb:

```
set transaction read write;
set transaction isolation level read committed;
```

So while it is true that Rdb supports the syntax, it does not currently execute it with the SQL standard semantics. It will take some time to decide the impact of such a fundamental change. In Rdb V7.1 an application can use the START TRANSACTION statement to get complete SQL:1999 conformance.

```
Start transaction read write isolation level read committed;
```

Basic Information Schema, Documentation Schema, etc.

SQL:1999 describes several “views” that describe objects stored in the database. They are very similar to Rdb system tables but they present the information in a different way. SQL:1999 CORE requires these views:

- COLUMNS,
- TABLES,
- VIEWS,
- TABLE_CONSTRAINTS,
- REFERENTIAL_CONSTRAINTS,
- CHECK_CONTRAINTS,
- SQL_FEATURES,
- SQL_SIZING,
- SQL_LANGUAGES,
- USER_DEFINED_TYPES, ROUTINES
- and PARAMETERS

Oracle could provide a set of views and information tables to implement this requirement if enough customer requests are received. It would be packaged much like the INFORMATION tables available in Rdb V7.1, and optionally applied to a database.

TIME and TIMESTAMP precision

The standard requires that we support fractional seconds precision up to 6 digits. The current TIME, TIMESTAMP and INTERVAL implementation will be enhanced to support (or exceed) this precision. Support for expanded fractional seconds precision is planned for a future release of Oracle Rdb.

CREATE TYPE

The syntax is fairly simple (also include DROP TYPE). The implementation will use the existing RDB$TYPES system table which is now part of Rdb V7.1 databases. The distinct types as described by SQL:1999 CORE are very similar to domains, however, there is strict type checking for assignment between columns, parameters and variables. Support for CREATE TYPE is planned for a future release of Oracle Rdb.

Flagger for SQL:1999

The flagger allows the user to detect any SQL syntax (or implied semantics) that exceeds the SQL standard level specified. It is hard to know how useful or popular such a feature would be for Rdb customers. The fact that there is currently no SQL:1999 (or even SQL:92) validation suite probably means there is very little interest in such a feature in Rdb.

REVOKE … RESTRICT

REVOKE now includes two drop actions: CASCADE and RESTRICT, the default being restrict. The aim of RESTRICT is to detect any relationship that is broken because of a missing privilege. For example, if stored module M has an AUTHORIZATION for SMITH then it runs as SMITH. If access to table T, referenced by the routines in M were denied to SMITH then module M would no longer function (in much the same way as if the table itself was deleted).

RESTRICT would cause the REVOKE to fail in this case, CASCADE (according to SQL:1999) would cause implicit DROP of the objects.

The current model for REVOKE implemented by Rdb is to perform the operation and leave the metadata structure intact. It certainly doesn't perform RESTRICT checking, nor does Oracle wish to implement the SQL:1999 destructive CASCADE operation which requires dropping any objects that no longer function without privileges.

Oracle Rdb
Guide to SQL Programming: SQL:1999 and Oracle Rdb V7.1
May 2003

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software that powers the Internet.