

Guide to Using SQL: Synonyms and the Rename Statement

A feature of Oracle Rdb

By Ian Smith
Oracle Rdb Relational Technology Group
Oracle Corporation

GUIDE TO USING SQL: SYNONYMS AND THE RENAME STATEMENT	3
SYNONYMS	3
SETTING UP THE DATABASE	5
OBJECT NAME SPACES.....	6
TRACKING DATABASE OBJECT DEPENDENCIES	7
DISPLAYING SYNONYMS	7
RMU EXTRACT AND SYNONYMS.....	8
<i>Example 1: Switching Tables</i>	8
<i>Example 2: Adding an Enhanced Function</i>	9
THE RENAME STATEMENT	11
OVERVIEW OF THE RENAME ACTION	11
FREQUENTLY ASKED QUESTIONS	14

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 release 7.1 and later versions.

Guide to Using SQL: Synonyms and the Rename statement

Oracle Rdb release 7.1 introduced a new *synonyms* feature that allows a database administrator to define alternate names for existing database objects. This feature is the foundation for the **rename** statement that was added in Oracle Rdb release 7.1.2.

Synonyms

The dictionary defines *synonym* as a word that means exactly the same as another word, or a word that can be interchanged with another word. This describes the intent of synonyms in Oracle Rdb. The database administrator has a facility to define equivalent names for an object in the database, in particular those objects that are commonly referenced in data definition statements (DDL) and data manipulation statement (DML).

The following table lists those objects that can be given a synonym.

Object Types	DML Usage	DDL Usage
Table or View	Can be referenced in a select , insert , update , or delete statement. Subselect clauses referencing the synonym can appear in various places as a value for set assignment, and as a parameter to call statements.	Can be referenced by view, trigger, constraint, module (declare transaction clause, default clause for a module variable), storage map and index definitions. Target for alter , comment , drop , grant , rename table , revoke , and truncate ¹ .
Domain	Can be referenced in a CAST function as the source of the data type. Interactive SQL allows the domain to be referred to in the edit using clause, or declare variable	Can be referenced as the source data type for a column. Target for alter , comment on , drop , and rename .

¹ **truncate** is only permitted for tables, not views.

	statement.	
Sequence	Can be referenced in a select , insert , update , or delete statement. Can appear in various places as a value for set assignment, and as a parameter to call statements.	Target for alter , comment on , drop , grant , rename , and revoke .
Synonym		Can be used as a substitute name in all DDL statements. Chains of synonyms can be created. Can be synonyms target for alter , comment on , drop , and rename .
Module		Target for alter , comment on , drop , grant , rename , and revoke .
Function	Can be referenced in a select , insert , update , or delete statement. Can appear in various places as a value for set assignment, and as a parameter to call statements.	Can be referenced in a computed by , automatic as , or default clause. Target for alter , comment on , drop , grant , rename , and revoke .
Procedure	Can be referenced in a call statement.	Can be referenced from functions procedures, and triggers actions. Target for alter , comment on , drop , grant , rename , and revoke .

As database systems grow and applications are enhanced there is a tendency to evolve the naming conventions, change names of tables, views and routines to better reflect their new and changed functionality. For example, a stored procedure named ADD_CUSTOMER might grow to verify credit limits as well as being used to add a new customer. Therefore, it might be natural to change the name to something like VERIFY_NEW_CUSTOMER.

The problem comes in changing the name in all places before placing the revised application into production. Synonyms allow both names to be used in the application and database. These references might exist in SQL module language applications, stored procedures, and interactive SQL maintenance scripts. Synonyms allow a level of safety when the switch to the new name is performed.

A curious feature of a synonym is that each may have its own synonym or many synonyms. For example, a table might have a formal name such as REL001_ORDER_TABLE, a simple name to allow easier interactive querying ORDER_TABLE, and a lazy name, OT, for people like myself

who prefer to type fewer characters. They could also be defined as synonyms for the base table name, or as chains.

```
SQL> create synonym ORDER_TABLE for table REL001_ORDER_TABLE;  
SQL> create synonym OT for table REL001_ORDER_TABLE;
```

Or they could be defined as a chain of synonyms, each referencing the previous definition:

```
SQL> create synonym ORDER_TABLE for table REL001_ORDER_TABLE;  
SQL> create synonym OT for table ORDER_TABLE;
```

The advantage of chaining is that an **alter synonym** statement can change ORDER_TABLE and the change will automatically be reflected by all synonyms up the chain. You can just issue a single alter synonym statement instead of many. The following **alter synonym** statement will implicitly change the OT synonym to reference the new table name:

```
SQL> alter synonym ORDER_TABLE for REL001_ORDER_TABLE_REVISION02;
```

Oracle Rdb assumes, but does not check, that the new table is identical in structure (column names and types) to the previously referenced table, or at least contains a superset of columns of the original table. An incompatible change would be detected at runtime if a routine or trigger, for instance, referenced a column that no longer existed, or had a different (and incompatible) data type.

These synonym chains can be connected up to 64 times before Oracle Rdb reports that an internal limit has been exceeded. This should be ample for most complex environments.

Setting Up the Database

The database administrator must enable support of the synonyms feature using the **alter database ... synonyms are enabled**² clause. When this statement is first executed a new system table named Rdb\$OBJECT_SYNONYMS³ is created and used to record the translation of the synonym to a base object, or to another synonym. This new table can never be removed from the database as

² This clause is not compatible with **multischema is on**, and therefore multischema databases do not support the features described in the paper.

³ Observant Rdb users might notice that a table Rdb\$SYNONYMS exists in a multischema database. However, this table provides a different function and is not used by the synonyms feature.

there is no **synonyms are disabled** clause. However, there is very little overhead having this table in the database. Use the **show database** command to determine if synonyms are enabled.

```
SQL> show database *
Default alias:
  Oracle Rdb database in file personnel
  Multischema mode is disabled
  Synonyms are enabled
  ...
```

If the database is already defined as **multischema is enabled** then you will not be permitted to enable the synonyms feature. At this time these two settings are mutually exclusive due to the complexity imposed by the multischema metadata.

Object Name Spaces

The name given to an Oracle Rdb object exists in separate name spaces. For instance, because Oracle Rdb keeps separate name spaces for objects and you could name a table, an index, a trigger and a constraint all with the same name.

The name spaces used in Oracle Rdb are:

- Table, view and sequence
- Constraints
- Triggers
- Indices
- Routines (functions and procedures)
- Modules
- Storage Maps
- Storage Areas
- Domains
- Query Outlines

Sequences and views share the table name space because their name can appear in the similar locations as a table name. Therefore, using unique names across all three objects allows Oracle Rdb to choose the correct object.

When you create a synonym the name cannot exist in any other name space. This prevents the user

creating an ambiguous synonym. For example:

```
SQL> create synonym ID_DOM for sequence ID_SEQ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-INVSYNONYM, invalid synonym name "ID_DOM" - name used by another object
```

Tracking Database Object Dependencies

When various objects are referenced in a data definition statement a dependency row is stored in the RDB\$INTERRELATIONS table. These rows always describe the base objects name and not the synonym allowing Oracle Rdb to detect a dependency no matter what name is used. In most cases an error message will report the base object name and not the synonym named in the command.

Displaying Synonyms

The **show synonym** command formats the contents of the Rdb\$OBJECT_SYNONYMS table. For example, the following **show** command displays all synonyms defined in the database:

```
SQL> show synonyms;
Synonyms in database with filename personnel
CURRENT_POSITION      View          CURRENT_JOB
CURRENT_SALARY        View          ACTIVE_SALARY_PLAN
STAFFING              Table        EMPLOYEES
UNIVERSAL_TIMESTAMP   Domain       STANDARD_DATE
```

The **show** commands for each object will also list synonyms below the list of objects. For example, a **show view** command will display the names of each view and any synonyms that exist for views.

```
SQL> show views;
User tables in database with filename personnel
ACTIVE_SALARY_PLAN    A view.
CURRENT_INFO          A view.
CURRENT_JOB           A view.
CURRENT_POSITION      A synonym for view CURRENT_JOB
CURRENT_SALARY        A synonym for view ACTIVE_SALARY_PLAN
```

If the **show table** or **show view** command is given a synonym it will inform you that the given name was a synonym and then display the details for the base table or view.

```
SQL> show table (column) e;
      E                               A synonym for table EMPLOYEES
Information for table EMPLOYEES

Columns for table EMPLOYEES:
Column Name           Data Type           Domain
-----
EMPLOYEE_ID          CHAR(5)             ID_NUMBER
... etc ...
SQL>
```

RMU Extract and Synonyms

When synonyms are enabled for a database, the RMU Extract command will include those synonyms with the table definition when you use the Option=SYNONYMS qualifier. The synonym chain for that table will follow immediately after the object definition.

An alternate report of synonyms can be extracted using Item=SYNONYMS qualifier on the RMU Extract command which extracts all the synonyms in one section. While this command is useful to display the definitions it is not always practical when rebuilding the database as the synonyms might be referenced by other definitions prior to the synonym being defined.

Example 1: Switching Tables

A financial institution records database transactions in a single table. Their design requires that every six months the table will be processed to calculate and generate interest payments. They wish to freeze the table at end of the month processing and resuming on a new table the next day.

This can be accomplished using several tables and a synonym.

Initially the application defines the table, TRANSACTION_DETAILS_001, and the synonym TRANSACTION_DETAILS. Applications written for this database only refer to the name TRANSACTION_DETAILS. When it becomes time to process the table, a simple **alter synonym** statement is used to switch to another identically defined table, such as TRANSACTION_DETAILS_002.

The **alter synonym** can be executed online, even if some applications are still running. These may be reporting applications that have used the previous synonym definition to reference the old table. Therefore, these applications will have to **disconnect** or restart at some point so that the synonym is reloaded and the new target table located.

How is this different from using a view? While it is true that the view could be dropped and redefined to reference the new table, the view includes table metadata locking semantics that will prevent the redirection while applications are active. This complicates the switch to the new table because the applications would be required to first **disconnect** and then wait while the view was changed. On the other hand the synonym change will be allowed online, and the change visible upon the next **attach** to the database. Another problem with the view approach is that any dependencies on the view would be invalidated by the **drop view**, and there are many options (**create trigger**, **create index** and **create storage map**) that are not permitted for views. Synonyms can be used without any such restrictions.

Example 2: Adding an Enhanced Function

When a function changes considerably it may be difficult to test in the database using the same function name. Therefore, a new function can be created and tested and placed into production for all new procedures and applications. Rather than editing the existing code it may be easier to use **drop function** and add a synonym using the old name of the function.

Consider this simple example. The procedure **add_employee** is used to insert a new employee into the **employees** table, along with accompanying **salary_history**, **job_history** rows. It also allocates and returns a new **employee_id**.

```
SQL> create module M
cont>         procedure ADD_EMPLOYEE
cont>             (in :last_name char(40)
cont>               ,in :first_name char(30)
cont>               ,out :employee_id id_dom);
cont>         begin ... end;
cont> end module;
SQL>
```

At some later time the function is superseded in an upward compatible way by defining the new parameters to be optional using the **default** clause, and using **varchar** instead of the original **char** types.

```
SQL> alter module M
cont>         add procedure ADD_EMPLOYEE_DETAILS
cont>         (in :last_name varchar(40)
cont>         ,in :first_name varchar(30)
cont>         ,out :employee_id id_dom
cont>         ,in :address_line_1 varchar(70) default null
cont>         ,in :address_line_2 varchar(70) default null);
cont>         begin ... end;
cont> end module;
```

By designing the replacement with the same required parameters the synonym can safely reference the new routine. In a single transaction the obsolete procedure can be dropped and replaced by a synonym.

```
SQL> alter module M
cont>         drop procedure ADD_EMPLOYEE cascade;
SQL>
SQL> create synonym ADD_EMPLOYEE
cont>         for procedure ADD_EMPLOYEE_DETAILS;
```

Now references to either ADD_EMPLOYEE or ADD_EMPLOYEE_DETAILS will activate the same routine.

The RENAME Statement

The **rename** statement can be used to change the name of an object in the database. The format of the command is shown here. The bold keywords are required and the references in brackets [] are optional.

```
rename [ object-type ] old-name to new-name;
```

The effect of this statement is to change the object name within the Oracle Rdb metadata. For instance, a **rename table** command will update the Rdb\$RELATIONS (table definition), Rdb\$RELATION_FIELDS (column definitions), Rdb\$INDICES (table index definitions), Rdb\$INTERRELATIONS (dependency table), and so on. The end result is that each system table will now reference the new object name.

Object-type can be one of the following keywords: domain, function, module, procedure, profile, role, sequence, synonym, table, user, or view. If it is omitted Oracle Rdb will search for the named object among all the system tables and use the first match that it finds. If your database uses the same name for different object types it is worthwhile using the full statement syntax to avoid any ambiguity.

The **rename** statement is a special form of the **alter** statement. In all cases the **rename** statement is equivalent to the corresponding **alter ... rename to** statement⁴. For the purposes of this article we will only describe the **rename** statement but the discussion applies equally to the **rename to** clause of the various **alter** statements.

Overview of the rename action

Consider what occurs when an object is created and used in an Oracle Rdb database. For example, when a sequence is created a row is written to the Rdb\$SEQUENCES system table recording the name and attributes of the sequence. This name can now be referenced by data definition commands such as **alter**, **drop**, **grant** and **revoke**, and appear within the definitions of constraints, views, stored procedures, triggers, **computed by** and **automatic as** columns. These definitions are stored in the database for the particular object as well as appearing in dependency rows of the

⁴ The **alter view** statement was introduced in Rdb V7.1.4.1, so in prior versions only the **rename view** statement can be used.

Rdb\$INTERRELATIONS system table. This table allows **drop** and **alter** statements to detect and report the usage of these objects.

When SQL processes a **rename sequence** statement it must change the name in the Rdb\$SEQUENCES table and also the dependency rows in Rdb\$INTERRELATIONS. However, Oracle Rdb makes no attempt to modify the original source definitions provided by the user that reference the sequence; these SQL sources are effectively descriptive text retained for the database administrator. Nor can Oracle Rdb modify the external source code stored in module language, or pre-compiled applications. Therefore, Oracle Rdb implicitly creates a synonym using the old name of the sequence (or other object) being renamed.

The database must support synonyms; otherwise this error will be reported.

```
SQL> rename sequence dept_id to next_department_id;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-UNSSYNONYM, this database does not have synonyms enabled
```

When the **rename sequence** statement is successful you can use the **show sequence** command to display all available sequences, including the synonym created with the old name of the sequence.

```
SQL> show sequence
Sequences in database with filename SQL$DATABASE
NEXT_DEPARTMENT_ID
RESTART_SEQUENCE
DEPT_ID                               A synonym for sequence NEXT_DEPARTMENT_ID
```

Each object listed by the **show** command has an indication for those that are synonyms. The **show sequence** command will accept either the name of the synonym⁵, or the base sequence name:

```
SQL> show sequence dept_id
DEPT_ID                               A synonym for sequence NEXT_DEPARTMENT_ID
NEXT_DEPARTMENT_ID
Sequence Id: 1
Initial Value: 1
Minimum Value: 1
.
.
.
```

⁵ You must be running Oracle Rdb release 7.1.3 or later to have synonym support in the SHOW command.

Now either the *old-name* or the *new-name* can be used to reference the object. Oracle Rdb relies on this feature to allow constraints, triggers and routines to continue functioning, as they will continue to use the old-name.

Over time, as these triggers and constraints are redefined, it might be possible to drop the old synonym when it no longer referenced. However, keep in mind that Oracle Rdb does not track external references, such as a SQL Module Language procedure that may be using that old name. Therefore, it is important that the synonyms created by the **rename** command not be dropped as they could leave the database unusable. For instance if the synonym is removed then that part of the application may no longer function and would report an error similar to the following example:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer
exist
-RDMS-F-BAD_SYM, unknown sequence symbol - DEPT_ID
```

This problem can easily be repaired by creating a synonym with the **create synonym** statement.

Once the rename completes the old name no longer has dependency rows in the database (for instance in the system table Rdb\$INTERRELATIONS) and so **drop** and **alter** cannot protect the metadata by reporting dependencies. Therefore, later releases⁶ of Oracle Rdb require DBADMIN privilege for the **drop synonym** statement when a **rename** statement created the synonym.

```
SQL> show synonym dept_id;
DEPT_ID
for sequence  NEXT_DEPARTMENT_ID
Comment:      Generated by RENAME SEQUENCE
SQL>
```

The comment for the synonym shows that it was created by **rename**. This comment can be replaced by the database administrator at any time using the **comment on synonym** or **alter synonym ... comment is** statements.

It was stated that the **rename** statement required that synonyms be enabled on the database. There is a small set of objects that do not require synonyms at all: **roles**, **users** and **profiles**. These objects are fully contained in the metadata. Therefore, if you have SQL scripts or applications that use the names of users or roles, then these applications will need to be modified if one of these objects is renamed.

⁶ This privilege requirement is enforced by Oracle Rdb release 7.1.4.4 and 7.2.0.2 and later.

Frequently asked Questions

How efficient are synonyms? The first reference to the synonym requires that the Oracle Rdb server traverse the chain and locate the referenced object. Once found the synonym name and base object name are stored in the symbol table; future accesses require no extra I/O. Oracle Rdb limits the longest chain to 64 elements.

What happens if I delete a synonym that makes up a chain? A **cascade** will be required for Oracle Rdb to allow this to happen. In this case Oracle Rdb will report an error at runtime as it will not be able to locate the base object through the chain. The error that is reported will be the same as seen by the database administrator after a **drop ... cascade** command is executed on a function or procedure. Such operations succeed but leave partial definitions in the database. Future references to an object that depends on those missing objects will result in an error reporting the missing object.

Can I have more than one synonym referencing an object? Yes. You can have as many synonyms as you like. However, take care not to overwhelm the database programmers with too many names as these will require extra management.

Can I use synonyms in the SHOW commands? Starting with Oracle Rdb release 7.1.3, SQL translates the synonyms in the **show** command. **Show** will also list any synonyms for the object type being listed. You can also use **show synonym** to display all synonyms in the database.

Can I use synonyms in RMU Load and Unload commands? Starting with Oracle Rdb release 7.1.4.4, and 7.2.0.2 RMU will accept synonyms for these commands. They will be translated internally to the name of the base table or view. Therefore, the record definition file (.RRD) or the unload interchange file (.UNL) will be created using the base table or view name. RMU Load will expect to be loading the base table name, so in some cases you may need to use the **Match_Name** qualifier with the name of the table as stored in the definition file.

Can I use synonyms in the RMU Extract command? The RMU Extract **Option=MATCH** qualifier does not currently support synonyms for objects (other than synonym names for the **Item=SYNONYM** qualifier). The wildcard matching supported by RMU Extract uses base object names only.

What happens if the DBA decides to rename the object back to its old name? You might expect that the first task would be to drop the synonym created with the old name. However, since this is

likely to be a common action SQL handles this automatically, and removes the synonym for you.

Is renaming a SYSTEM table permitted? For obvious reasons this is not allowed as this would prevent the database from being used. Client tools such as Interactive SQL, ODBC, SQL/Services, JDBC and so on assume the documented names of the Oracle Rdb system metadata.

What happens when a table has an identity column? In this case both the table and the identity sequence have the same name. SQL does not allow you to rename the **identity** sequence. However, when the table is renamed the identity sequence is also implicitly renamed so that the sequence and table remain bound together

ORACLE

Oracle Rdb
Guide to Using SQL: Synonyms and the Rename Command
May 2006

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.

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2006 Oracle Corporation
All rights reserved.