SQL Rename

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** statement[4]. 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.

Oracle rename table syntax

Oracle provides a rename table syntax as follows:

```
alter table
   table_name
rename to
   new_table_name;
```

or

```
RENAME table_name TO new_table_name;
```

For example, we could rename the customer table to old_customer with this syntax:

```
alter table emp

rename to employees;
```

or

```
RENAME emp TO employees;
```

When you rename an Oracle table you must be aware that Oracle does not update applications (HTML-DB, PL/SQL that referenced the old table name) and PL/SQL procedures may become invalid.

Renaming an Oracle table column

desc emp;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	Number	-	4	0	-	-	-	-
	ENAME	Varchar2	10	-	-	-	✓	-	-
	JOB	Varchar2	9	-	-	-	✓	-	-
	MGR	Number	-	4	0	-	✓	-	-
	HIREDATE	Date	7	-	-	-	✓	-	-
	SAL	Number	-	7	2	-	✓	-	-
	COMM	Number	-	7	2	-	✓	-	-
	DEPTNO	Number	-	2	0	-	✓	-	-

```
ALTER TABLE emp
RENAME COLUMN ename to LastName;
```

```
DESC emp;
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP	EMPNO	Number	-	4	0	-	-	-	-
	LASTNAME	Varchar2	10	-	-	-	✓	-	-
	JOB	Varchar2	9	-	-	-	✓	-	-
	MGR	Number	-	4	0	-	✓	-	-
	HIREDATE	Date	7	-	-	-	✓	-	-
	SAL	Number	-	7	2	-	✓	-	-
	COMM	Number	-	7	2	-	✓	-	-
	DEPTNO	Number	-	2	0	-	✓	-	-

Renaming Constraints

In addition to renaming tables and indexes Oracle9i Release 2 allows the renaming of columns and constraints on tables. In this example once the the `TEST1` table is created it is renamed along with it's columns, primary key constraint and the index that supports the primary key:

```
CREATE TABLE test1 (
      col1  NUMBER(10) NOT NULL,
      col2  VARCHAR2(50) NOT NULL);
```

```
Table created.
```

```
ALTER TABLE test1 ADD (
 CONSTRAINT test1_pk PRIMARY KEY (col1));
```

```
Table altered.
```

```
DESC test1;
```

```
 Name                  Null?    Type
 --------------------- -------- --------------------
 COL1                  NOT NULL NUMBER(10)
 COL2                  NOT NULL VARCHAR2(50)
```

```
SELECT constraint_name
    FROM   user_constraints
    WHERE  table_name      = 'TEST1'
    AND    constraint_type = 'P';
```

```
CONSTRAINT_NAME
------------------------------
TEST1_PK

1 row selected.
```

```
SELECT index_name, column_name
FROM   user_ind_columns
WHERE  table_name = 'TEST1';
```

```
INDEX_NAME            COLUMN_NAME
--------------------  --------------------
TEST1_PK              COL1

1 row selected.
```

Examples of Renaming tables, columns, primary key and supporting index.

```
ALTER TABLE test1 RENAME TO test;
```

```
Table altered.
```

```
ALTER TABLE test RENAME COLUMN col1 TO id;
```

```
Table altered.
```

```
ALTER TABLE test RENAME COLUMN col2 TO description;
```

```
Table altered.
```

```
ALTER TABLE test
RENAME CONSTRAINT test1_pk TO test_pk;
```

```
Table altered.
```

```
ALTER INDEX test1_pk RENAME TO test_pk;
```

```
Index altered.
```

```
DESC test;
```

```
 Name                  Null?    Type
 --------------------- -------- --------------------
 ID                    NOT NULL NUMBER(10)
 DESCRIPTION           NOT NULL VARCHAR2(50)
```

```
SELECT constraint_name
    FROM   user_constraints
    WHERE  table_name      = 'TEST'
    AND    constraint_type = 'P';
```

```
CONSTRAINT_NAME
--------------------
TEST_PK

1 row selected.
```

```
SELECT index_name, column_name
    FROM   user_ind_columns
    WHERE  table_name = 'TEST';
```

```
INDEX_NAME            COLUMN_NAME
--------------------  --------------------
TEST_PK               ID

1 row selected.
```