Institute of Technology Tralee
Institiúid Teicneolaíochta Trá Lí

Database Concepts
Lab 07 – Changing a Database Definition

In this lab you will learn how to use Oracle scripts to:

- Modify the definition of a database
- Add a column to a database table
- Remove a column from a database table
- Change the data type / size of a column
- Add a constraint to a database table or column
- Remove a constraint from a database table or column

Before You Start:

Ensure that you have the DVDSYS database in your tablespace.
Execute your script DVDData.sql to populate this database.

Run Oracle SQL Developer and connect to your database.

Try the examples illustrated during this session on this database.

Changing the Database Definition

Changing the database definition b***efore*** it has gone operational is not an issue. The DBA can simply amend the script containing the database build and re-execute it. Sample data may then be loaded into the database as required.

Changing the database definition ***after*** it has gone operational is not as simple. Care must be taken to ensure that any changes made to the database do not invalidate the data contained in it. Ideally, the tables created will not require any structural modifications. When planning and designing a database, you must strive for perfection! However, even perfect tables need changes from time to time.

Certain modifications can be made to the structure of a table without any restrictions. Some modifications are allowed subject to integrity rules being upheld. Other modifications are never allowed to an existing table's structure.

Modifications never allowed

Changing a column's name

Changing the name of a column in a table would require any software application interacting with that table to be changed also. Structural independence would be compromised.

Removing the last column/attribute of a table

A table must have at least one column.

If you have already created a table and there is a need to make a change that is not allowed, you will need to DROP the table and re-create it. You may lose essential data – be careful!

Modifications allowed *without* restrictions

Adding an attribute (column) to a table

Adding a column to a table does not require existing software applications to be changed (unless the new column needs to be referenced by the application).

Deleting a foreign key constraint from a table

Consistency of data may be compromised when a foreign key constraint is removed.

Deleting a primary key constraint from a table

It should be noted however, that when a primary key constraint is removed from a table, any references to it from other tables in the database are also removed.

Increasing the size of an attribute (column)

For example, VARCHAR2(15) can be changed to VARCHAR2(20) without compromising existing data.

Modifications allowed *with* restrictions

Adding a foreign key constraint to a table

This is allowed ***only if*** the current values of the foreign key attribute are NULL or the value of the attribute exists in the referenced table's primary key.

Adding a primary key constraint

This is allowed only if the current values of the attribute are NOT NULL and are unique.

Changing an attribute's data type

This is allowed only if there is no data in that column.

Reducing an attribute's size

This is allowed only if there is no data in that column.

Adding a UNIQUE constraint to an attribute

This is possible only if the current data values in the attribute are unique.

Adding a CHECK constraint to an attribute

This is possible only if the current data values in the attribute comply with the new constraint.

Adding a DEFAULT constraint to an attribute

This is possible only if there is no data in the attribute.

The SQL ALTER TABLE Statement

This statement is used to change an existing table definition.

The table indicated in the ALTER TABLE statement must already exist.

This statement can be used to:

- Add a new column to a table
- Remove an existing column from a table
- Modify the data type for an existing column in a table
- Add or remove a constraint (Primary Key, Foreign Key, NOT NULL, etc.).

Adding a New Column to an Existing Table

The general syntax is:

> **ALTER TABLE** *tablename*
> **ADD** *column name datatype*[NOT] NULL;

Add a column DOB (Date of birth) to the table Members.

> **ALTER TABLE** Members
> **ADD** DOB Date;

After execution the column added has NULL values for all existing records.

Modifying an Existing Column

It is also possible to change the data type of an attribute.

The general syntax is:

> **ALTER TABLE** *tablename*
> **MODIFY** *columnname newdatatype* [NOT] null;

The SQL statement to change the ***data type*** of the attribute ***Description*** in the table Age_Ratings from ***char*** to ***varchar2*** is as follows:

> **ALTER TABLE** Age_Ratings
> **MODIFY** Description varchar2(25);

The appearance of the table is unchanged. SELECT all records to see the table.

The SQL statement to change the ***size*** of the attribute Description from varchar2(25) to char(30) is as follows:

> **ALTER TABLE** Age_Ratings
> **MODIFY** Description char(30);

Here, both the datatype and size are changed at the same time. Again, the appearance of the table is unchanged.

This change is permitted since the size of the attribute is being ***increased***.
To reduce the size of the attribute, the column must be empty.

Adding a Constraint

ALTER TABLE can also be used to add a constraint to a table. The types of constraints which may be added using ALTER TABLE include:

- Primary Key
- Foreign Key
- Check / Default / Unique

The syntax to add a PRIMARY KEY constraint is:

> **ALTER TABLE** *tablename*
> **ADD CONSTRAINT** *constraint_name*
> PRIMARY KEY (Col1 [,col2,..]);

For example (assuming the primary key for the table Rentals is not already defined):

> **ALTER TABLE** Rentals
> **ADD CONSTRAINT** pk_Rentals **PRIMARY KEY** (Rental_Id);

The syntax to add a FOREIGN KEY constraint is:

> **ALTER TABLE** *tablename*
> **ADD CONSTRAINT** *constraint-name* **FOREIGN KEY** (*colname*) REFERENCES
> *Parent-table (col name)*;

NOTE:
There are two ways in which Primary/Foreign key constraints may be defined:

- Use the **CREATE TABLE** query with PRIMARY/FOREIGN KEY constraints to reference tables already created.

- Create all tables without PRIMARY/FOREIGN key constraints. Then use the **ALTER TABLE** query to add the required constraint.

The first option is the preferred option. This ensures that primary keys are defined as part of the table definition.

Dropping a Constraint

Primary key and foreign key constraints can be dropped from a table definition. This is done by *dropping* the primary key or foreign key constraint. You must specify the constraint name when doing this.

The general syntax for dropping a constraint is:

ALTER TABLE *tablename*
DROP CONSTRAINT *constraint-name*;

For example, to remove the foreign key constraint, ***fk_Rental_DVDs*** on the table Rentals:

ALTER TABLE Rentals
DROP CONSTRAINT fk_Rental_DVDs;

Dropping a Column

You must be careful not to remove essential data from your database when removing column(s) from a table.

Please note the following:

- Only one column may be dropped at a time.
- If the column being dropped contains data, then the data is lost.
- When a column is dropped, there must be at least one remaining column in the table ie. it is not possible to drop the last column of a table.
- It is not possible to recover a dropped column!

General syntax:

ALTER TABLE *table name*
DROP COLUMN *columnname*;

The SQL statement to drop the column ***email*** from the table ***Members*** is as follows:

ALTER TABLE Members
DROP COLUMN email;

Oracle also allows you to mark columns as unused. This is not advisable, mainly because the storage space used by unused columns is not released.

General syntax:

ALTER TABLE *table name*
SET UNUSED (*columnname)*;

The SQL statement to mark the column email as unused is as follows:

ALTER TABLE Members
SET UNUSED(email);

Unused columns are not displayed with other columns and they are not displayed in the table’s structure.

The following statement may drop unused columns:

> **ALTER TABLE** *table name*
> **DROP UNUSED COLUMNS**;

For example:

> **ALTER TABLE** Members
> **DROP UNUSED COLUMNS**;

If no columns are marked as unused, this statement does not return any error message.

Renaming a Table

You can rename a table provided you are the ***owner*** of the table.

General syntax:

> **RENAME** *old table name* **TO** *new table name*;

For example:

> **RENAME** Stock **TO** StockItems;

Oracle will display a 'Table renamed' message when this statement is executed.
The RENAME statement may also be used to change the name of other Oracle objects such as views.

Truncating a Table

Truncating a table is removing all records/rows from the table. The structure of the table stays intact. You must be the owner of the table WITH the **DELETE TABLE** privilege to truncate a table.

The SQL **DELETE** statement can be used to delete 1 or more or all rows from a table and **is reversible.** The **TRUNCATE** statement, however, is not reversible.

Truncation releases storage space occupied by the table, but deletion does not.

General syntax:

> **TRUNCATE TABLE** *table name*;

For example:

> **TRUNCATE TABLE** Members;

Be careful when you use this statement as it cannot be reversed!

Exercise

1. Rebuild and populate your DVDSys database.
2. Write an Oracle script which contains the ALTER TABLE statements you have tried in this session.
3. Test the script to ensure that it executes successfully.
4. Check the database definition to ensure that the changes have been applied.