

## Lesson 07 Database Objects

DBMS/SQL

# Lesson Objectives

To understand the following Database Objects:

- Basic Data Types
- Data Integrity
- Examples of CREATE TABLE
- Examples of ALTER TABLE
- Database Objects
- Index
- Synonym
- Sequence
- View
- Deleting Database Objects
- Tips and Tricks



# Overview

A database is a collection of structures with appropriate authorizations and accesses that are defined.

The structures in the database like tables, indexes, etc. are called as objects in the database.

All objects that belong to the same user are said to be the "schema" for the particular user.

Information about existing objects can be retrieved from `dba_/user_/all_objects`.

# Basic Data Types

Given below are the basic Data Types:

Datatype	Description
<b>CHAR(n)</b>	<b>Stores fixed length string. Maximum length = 2000 bytes</b> <b>For example: NAME CHAR(15)</b>
<b>VARCHAR(n)</b>	<b>Stores variable length string. Maximum length = 4000 bytes</b> <b>For example: DESCRIPTION VARCHAR2(100)</b>
<b>BIGINT</b>	<b>Stores variable length . Maximum length = <math>2^{64}-1</math></b>
<b>integer(p,s)</b>	<b>Stores numeric data . Range is 1E-129 to 9.99E125</b> <b>Max Number of significant digits = 38</b> <b>For example: SALARY NUMBER(9,2)</b>
<b>DATE</b>	<b>Stores DATE. Range from January 1, 4712 BC to December 31, 9999 AD.</b> <b>Both DATE and TIME are stored. Requires 7 bytes.</b> <b>For example: HIREDATE DATE</b>

## Basic Data Types contd..

Datatype	Description
<b>TIMESTAM P</b>	<b>Stores the time to be stored as a date with fractional seconds. Extension to the DATA datatype There are some variations of the data type</b>

## Table

Tables are objects, which store the user data.

Use the CREATE TABLE statement to create a table, which is the basic structure to hold data.

For example:

```
CREATE TABLE book_master  
(book_code number,  
  book_name varchar2(50),  
  book_pub_year number,  
  book_pub_author varchar2(50));
```

## What is Data Integrity?

### Data Integrity:

- “Data Integrity” allows to define certain “data quality requirements” that must be met by the data in the database.
- Oracle uses “Integrity Constraints” to prevent invalid data entry into the base tables of the database.
  - You can define “Integrity Constraints” to enforce the business rules you want to associate with the information in a database.
  - If any of the results of a “DML statement” execution violate an “integrity constraint”, Oracle rolls back the statement and returns an error.

## Advantages

### Advantages of Integrity Constraints:

- Integrity Constraints have advantages over other alternatives. They are:
  - Enforcing “business rules” in the code of a database application.
  - Using “stored procedures” to completely control access to data.
  - Enforcing “business rules” with triggered stored database procedures.



## Applying Constraints

Constraints can be defined at

- Column Level

```
CREATE TABLE tablename  
  (column datatype [DEFAULT expr]  
  [column_constraint] ,  
  .....)
```

- Table Level

```
CREATE TABLE tablename  
  (column datatype,  
   column datatype  
   .....  
   [CONSTRAINT constraint_name] constraint_type  
   (column,...))
```

## Types of Integrity Constraints

Let us see the types of Data Integrity Constraints:

- Nulls
- Default
- Unique Column Values
- Primary Key Values
- Check
- Referential Integrity

## NOT NULL Constraint

The user will not be allowed to enter null value.

For Example:

- A NULL value is different from a blank or a zero. It is used for a quantity that is “unknown”.
- A NULL value can be inserted into a column of any data type.

```
CREATE TABLE student_master
(student_code number(4) NOT NULL,
 dept_code   number(4) CONSTRAINT dept_code_nn
                                NOT NULL );
```

### 7.3: Examples of CREATE TABLE

## DEFAULT clause

If no value is given, then instead of using a “Not Null” constraint, it is sometimes useful to specify a default value for an attribute.

For Example:

- When a record is inserted the default value can be considered.

```
CREATE TABLE staff_master(  
  Staff_Code number(8) PRIMARY KEY,  
  Staff_Name varchar2(50) NOT NULL,  
  Staff_dob date,  
  Hiredate date DEFAULT sysdate,  
  ....)
```

## UNIQUE constraint

The keyword UNIQUE specifies that no two records can have the same attribute value for this column.

For Example:

```
CREATE TABLE student_master  
(student_code number(4),  
  student_name varchar2(30) ,  
  CONSTRAINT stu_id_uk UNIQUE(student_code )) ;
```

## PRIMARY KEY constraint

The Primary Key constraint enables a unique identification of each record in a table.

For Example:

```
CREATE TABLE Staff Master
(staff_code number(6)
CONSTRAINT staff_id_pk PRIMARY KEY,
staff_name varchar2(20)
.....);
```

## CHECK constraint

CHECK constraint allows users to restrict possible attribute values for a column to admissible ones.

For Example:

```
CREATE TABLE staff_master
( staff_code number(2),
  staff_name varchar2(20),
  staff_sal number(10,2) CONSTRAINT staff_sal_min
                                CHECK (staff_sal > 1000),
  ....) ;
```

## FOREIGN KEY constraint

The FOREIGN KEY constraint specifies a “column” or a “list of columns” as a foreign key of the referencing table.

The referencing table is called the “child-table”, and the referenced table is called “parent-table”.

For Example:

```
CREATE TABLE student_master
(student_code number(6) ,
dept_code number(4) CONSTRAINT stu_dept_fk
REFERENCES
department_master(dept_code),
student_name varchar2(30) );
```



## Create new table based on existing table

Constraints on an “old table” will not be applicable for a “new table”.

```
CREATE TABLE student_dept10 AS  
SELECT student_code, student_name  
FROM student_master WHERE dept_code = 10
```

## ALTER Table

Given below is an example of ALTER TABLE:

```
ALTER TABLE table_name
    [ADD (col_name col_datatype col_constraint ,...)]|
    [ADD (table_constraint)]|
    [DROP CONSTRAINT constraint_name]|
    [MODIFY existing_col_name new_col_datatype
                                     new_constraint new_default]
    [DROP COLUMN existing_col_name]
    [SET UNUSED COLUMN existing_col)name];
```

## ALTER Table – Add clause

The “Add” keyword is used to add a column or constraint to an existing table.

- For adding three more columns to the emp table, refer the following example:

```
ALTER TABLE Student_Master  
ADD (last_name varchar2(25) );
```

## ALTER Table – Add clause

For adding Referential Integrity on “mgr\_code” column, refer the following example:

```
ALTER TABLE staff_master  
    ADD CONSTRAINT FK FOREIGN KEY (mgr_code)  
REFERENCES staff_master(staff_code);
```

## ALTER Table – MODIFY clause

### MODIFY clause:

- The “Modify” keyword allows making modification to the existing columns of a table.
  - For Modifying the width of “sal” column, refer the following example:

```
ALTER TABLE staff_master  
MODIFY (staff_sal number (12,2) ) ;
```

## ALTER Table – DROP clause

The DROP clause is used to remove constraints from a table.

- For Dropping the FOREIGN KEY constraint on “department”, refer the following example:

```
ALTER TABLE student_master  
DROP CONSTRAINT stu_dept_fk ;
```

## Dropping Column

Given below are the ways for “Dropping” a column:

- 1a. Marking the columns as unused and then later dropping them.
- 1b. The following command can be used later to permanently drop the columns.

```
ALTER TABLE staff_master SET UNUSED COLUMN staff_address;  
ALTER TABLE staff_master SET UNUSED (staff_sal, hiredate);
```

```
ALTER TABLE emp DROP UNUSED COLUMNS;
```

## Dropping Column

- Directly dropping the columns.

```
ALTER TABLE staff_master DROP COLUMN staff_sal;
```



## Drop a Table

The DROP TABLE command is used to remove the definition of a table from the database.

For Example:

```
DROP TABLE staff_master;
```

```
DROP TABLE Department_master  
CASCADE CONSTRAINTS;
```

## Rename a Table

The RENAME command is used to give a new name to the table. Views can also be renamed using this command

For Example:

```
RENAME staff_master TO new_staffmaster;
```

## Truncating a Table

The TRUNCATE command is used to permanently remove the data from a table, keeping the table structure intact.

For Example:

```
TRUNCATE TABLE staff_master ;
```

## User\_Tables & User\_Objects

To view the names of tables owned by the user, use the following query:

To view distinct object types owned by the user, use the following query:

```
SELECT table_name  
FROM user_tables
```

```
SELECT DISTINCT object_type  
FROM user_objects ;
```

## Usage of Index

Index is a database object that functions as a “performance-tuning” method for allowing faster retrieval of records.

Index creates an entry for each value that appears in the indexed columns.

The absence or presence of an Index does not require change in wording of any SQL statement.

## Usage of Index

### Syntax:

```
CREATE [UNIQUE] INDEX index_name  
ON table_name(col_name1 [ASC|DESC],col_name2,.....)
```

## Creating an Index

Example 1: A simple example of an Index is given below:

```
CREATE INDEX staff_sal_index ON  
staff_master(staff_sal);
```

Example 2: To allow only unique values in the field “ename”, the CREATE statement should appear as shown below:

```
CREATE UNIQUE INDEX staff_ename_unindex  
ON staff_master(staff_name );
```

## How are Indexes created?

Indexes can be either created “automatically” or “manually”.

- Automatically: A unique Index is automatically created when you define a PRIMARY KEY or UNIQUE constraint in a table definition.
- Manually: A non-unique index can be created on columns by users in order to speed up access to the rows.



## Usage of Synonym

A “Synonym” is an “alias” that is used for any table, view, materialized view, sequence, procedure, function, or package.

- Since a Synonym is simply an alias, it does not require storage except for storage of its definition in the data dictionary.
- Synonyms are often used for “security” and “convenience”.
- Synonyms can be created as either “public” or “private”.
- Synonyms are useful in hiding ownership details of an object.

## Usage of Synonym

### Syntax

■ where:

- Existing\_name is the name of a table, view, or sequence.
- PUBLIC is used to grant permission to all users for accessing the object by using the new name. (This is done only by a DBA.)

```
CREATE [PUBLIC] SYNONYM another_name FOR  
existing_name
```

## Creating a Synonym

Here is an example for synonym:

- Suppose a procedure "proc1" is created in a schema "scott". While calling this procedure, if the user refers it as "scott.proc1", then a synonym is created as:

```
Create synonym prc1 for scott.proc1;
```

## Usage of View

A View can be thought of as a “stored query” or a “virtual table”, i.e. a logical table based on one or more tables.

- A View can be used as if it is a table.
- A View does not contain data.

# Usage of View

## Syntax

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view [(alias[,  
alias]...)] AS subquery  
[WITH CHECK OPTION [CONSTRAINT constraint]]  
[WITH READ ONLY [CONSTRAINT constraint]];
```

## Creating a View

Given below is an example of a simple View:

```
CREATE VIEW staff_view  
AS  
SELECT * FROM staff_master  
WHERE hiredate > '01-jan-82';
```

## Creating a View

### Creating a Complex View:

- As shown in the example given below, create a Complex View that contains group functions to display values from two tables.

```
CREATE VIEW dept_sum_vu
(name, minsal, maxsal, avgsal)
AS SELECT dept.dept_name, MIN(staff.staff_sal),
      MAX(staff. staff_sal),AVG(staff. staff_sal)
FROM   staff_master staff, department_master dept
WHERE  staff.dept_code = dept.dept_code
GROUP BY dept.dept_name;
```

## Creating a View

### Creating a View with WITH CHECK OPTION:

```
CREATE VIEW staff_vw  
AS  
SELECT * FROM staff_master  
WHERE deptno =10 WITH CHECK OPTION constraint cn;
```



## Rules for performing operation on View

You can perform “DML operations” on simple Views.

You cannot remove a row if the View contains the following:

- Group functions
- A GROUP BY clause
- The DISTINCT keyword

## Deleting a Database Objects

### Example 2:

If new\_emp is a Synonym for a table, then the Table is not affected in any way. Only the duplicate name is removed.

```
DROP SYNONYM new_emp;
```

When creating tables based on subquery the number of specified columns if defined for the table should match to the number of columns in the subquery.

Create an index if

- A column contains a wide range of values
- A column contains a large number of null values
- One or more columns are frequently used together in a WHERE clause or a join condition
- The table is large and most queries are expected to retrieve less than 2 to 4 percent of the rows

An Index is not very useful if :

- The table is small
- The columns are not often used as a condition in the query
- Most queries are expected to retrieve more than 2 to 4 percent of rows in the table
- The table is updated frequently
- The indexed columns are referenced as part of an expression

# Summary

In this lesson, you have learnt:

- What are Database Objects?
- Basic Data Types
- Data Integrity
- Different types of Database Objects:
- Modification of Database Objects
- Deleting Database Objects



# Review – Questions

Question 1: Indexes can be created \_\_\_\_\_ or \_\_\_\_\_

Question 2: \_\_\_\_\_ obtains the current sequence value

Question 3: Synonyms can be created as either \_\_\_\_\_ or \_\_\_\_\_



# Review – Questions

Question 4: Gaps in sequence values can occur when there is a rollback

- True / False

Question 5: Synonyms are useful in hiding ownership details of an object.

- True / False

