# DBMS SQL

## Lesson 09: Database Objects

Capgemini

# Lesson Objectives

➢ To understand the following Database Objects:
- Table
- Index
- Synonym
- Sequence
- View

# 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 |
|---|---|
| CHAR(n) | Stores fixed length string. Maximum length = 2000 bytes<br>For example: NAME CHAR(15) |
| VARCHAR2(n) | Stores variable length string. Maximum length = 4000 bytes<br>For example: DESCRIPTION VARCHAR2(100) |
| LONG(n) | Stores variable length string . Maximum length = 2 GIGA bytes<br>For example: SYNOPSIS  LONG(5000) |
| NUMBER(p,s) | Stores numeric data . Range is 1E-129 to 9.99E125<br>Max Number of significant digits = 38<br>For example: SALARY NUMBER(9,2) |
| DATE | Stores DATE. Range from January 1, 4712 BC to December 31, 9999 AD. Both DATE and TIME are stored. Requires 7 bytes.<br>For example: HIREDATE DATE |
| RAW(n) | Stores data in binary format such as signature, photograph. Maximum size = 255 bytes |
| LONG RAW(n) | Same as RAW. Maximum size = 2 Gigabytes |

# Basic Data Types contd..

| Datatype | Description |
|----------|-------------|
| **TIMESTAMP** | **Stores the time to be stored as a date with fractional seconds. Extension to the DATA datatype** <br> **There are some variations of the data type** |

➢ **TimeStamp Datatype variations**

| | |
|---|---|
| **TIMESTAMP [(fractional_seconds_precision)]** | **By default** |
| **TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE** | **This variant includes a time zone as displacement value which is difference between local time and UTC** |
| **TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE** | **Inlcudes a time zone displacement in its value. The server returns the data in the users local time zone** |

# Basic Data Types contd..

| Datatype | Description |
|----------|-------------|
| **BLOB** | Binary Large Object<br>• **Stores any kind of data in binary format.**<br>• **Typically used for multimedia data such as images, audio, and video.** |
| **CLOB** | Character Large Object<br>• **Stores string data in the database character set format. Used for large strings or documents that exclusively use the database character set.**<br>• **Characters in the database character set are in a fixed width format.** |
| **NCLOB** | National Character Set Large Object<br>• **Stores string data in National Character Set format. Used for large strings or documents in the National Character Set.**<br>• **Supports characters of varying width format.** |
| **BFILE** | External Binary File<br>• **A binary file stored outside the database in the host operating system file system, but accessible from database tables.**<br>• **BFILEs can be accessed from your application on a read-only basis. Use BFILEs to store static data, such as image data, that does not need to be manipulated in applications.**<br>• **Any kind of data, that is, any operating system file, can be stored in a BFILE. For example: You can store character data in a BFILE, and then load the BFILE data into a CLOB specifying** |

# 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
- Unique Column Values
- Primary Key Values
- 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 );
```

# 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_dept117 AS
> SELECT student_code, student_name
> FROM student_master WHERE  dept_code = 117

# 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 – Enable | Disable clause

➢ ENABLE | DISABLE Clause:
- The ENABLE | DISABLE clause allows constraints to be enabled or disabled according to the user choice without removing them from a table.
- Refer the following example:

ALTER TABLE staff_master DISABLE CONSTRAINT
SYS_C000934;

# 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;

# 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 it's 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 Sequence

➢ A "Sequence" is an object, which can be used to generate sequential numbers.

➢ A Sequence is used to fill up columns, which are declared as UNIQUE or PRIMARY KEY.

➢ A Sequence uses "NEXTVAL" to retrieve the next value in the sequence order.

# Creating a Sequence

➤ For example, suppose we have created a sequence "seq_no", then it's next value can be obtained as "seq_no.nextval".

CREATE SEQUENCE seq_name

[INCREMENT BY n1] [START WITH n2]

[MAXVALUE n3] [MINVALUE n4] [CYCLE|NOCYCLE]

[CACHE|NOCACHE];

# Creating a Sequence

➢ Here is one more example of sequence:
- s1 will generate numbers 1,2,3....,10000, and then stop.

```
CREATE SEQUENCE  s1
            INCREMENT BY 1
            START WITH 1
            MAXVALUE 10000
            NOCYCLE ;
```

# NEXTVAL and CURRVAL pseudo columns

➤ NEXTVAL returns the next available sequence value.
- It returns a unique value every time it is referenced, even for different users.

➤ CURRVAL obtains the current sequence value.

➤ NEXTVAL must be issued for the Sequence before CURRVAL can be referenced.

# Characteristics of Sequence

➢ Characteristics of a Sequence:

- Caching the Sequence values in memory to give faster access to those Sequence values
- Gaps in Sequence values can occur when:
  - a rollback occurs
  - the system crashes
  - a Sequence is used in another table
- Viewing the next available value by querying the USER_SEQUENCES table, when the sequence is created with NOCACHE

# Drop a Sequence

➢ A Sequence can be removed from the data dictionary by using the DROP SEQUENCE statement.

➢ Once removed, the Sequence can no longer be referenced.

DROP SEQUENCE dept_deptid_seq;

Sequence dropped.

# 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
- The pseudocolumn ROWNUM keyword

# Inline View

➢ "Inline view" is not a schema object like a regular View. However, it is a temporary query with an alias.

> SELECT dept_name,staff_name,staff_sal
> FROM staff_master staff,
> (SELECT dept_name,dept_code
>  FROM department_master) dept
> WHERE staff.dept_code=dept.dept_code

# Inline View

➢ You can use Order By clause, as well, in the Inline View.

- This is very useful when you want to find the top n values in a table.

SELECT rownum,staff_name

FROM (SELECT staff_name,staff_sal

FROM staff_master ORDER BY staff_sal desc)

WHERE rownum < 5

# 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;
```

# Guidelines

➢ 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

# Guidelines

➢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

➢ 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: You cannot use ORDER BY clause, in the Inline View
- True / False

➢ Question 5: Gaps in sequence values can occur when there is a rollback
- True / False

➢ Question 3: Synonyms are useful in hiding ownership details of an object.
- True / False