



Lesson Objectives



To understand the following Database Objects:

- Basic Data Types
- Data Integrity
- Examples of CREATE TABLE
- Examples of ALTER TABLE
- Database Objects
- Index
- View
- 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`.

Database Objects:

Following is a list of Database objects:

- Tables
- Views
- Indexes
- Clusters
- Synonyms
- Sequences
- Procedures
- Functions
- Packages
- Triggers .

Basic Data Types



Given below are the basic Data Types:

Datatype	Description
CHAR(n)	Stores fixed length string. Maximum length = 2000 bytes For example: NAME CHAR(15)
VARCHAR2(n)	Stores variable length string. Maximum length = 4000 bytes For example: DESCRIPTION VARCHAR2(100)
LONG(n)	Stores variable length string . Maximum length = 2 GIGA bytes For example: SYNOPSIS LONG(5000)
NUMBER(p,s)	Stores numeric data . Range is 1E-129 to 9.99E125 Max Number of significant digits = 38 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. 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 supported in SQL:

Scalar Data Types:

- The traditional Data Types are called "Scalar Data Types". The slide discusses some of the Scalar Data Types.

Basic Data Types contd..



Datatype	Description
TIMESTAMP	Stores the time to be stored as a date with fractional seconds.
P	Extension to the DATE datatype
	There are some variations of the data type

Basic Data Types supported in SQL:

The TimeStamp data type is introduced in Oracle 9i. This data type provides support for time zones. It is an extension of the DATE datatype.

TIMESTAMP [(fractional_seconds_precision)] – It stores the year, month & day of the date data type. In addition to that it also stores hour, minute, second and fractional second value.

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));
```

Creating tables is done with the create table command. You can add rows to a table with the INSERT statement, after creating a table.

The above create table command does the following:

- Defines the table name
- Defines the columns in the table and the datatypes of those columns

In above example, we create a table called BOOK_MASTER which has 4 columns. The first column and third column is defined as NUMBER datatype. This means we will be storing numbers in this column. The second and fourth columns are of VARCHAR2 datatype. We will be storing text data in these columns

Syntax:

```
CREATE TABLE table_name
(
  {col_name.col_datatype [[CONSTRAINT
  const_name][col_constraint]],...
  [table_constraint],...
}
[AS query]
```

- If a table is created as shown in the slide, then there is no restriction on the data that can be stored in the table.
- However, if we wish to put some restriction on the data, which can be stored in the table, then we must supply some “constraints” for the columns. We will see the Constraints as next topic.

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.

Data Integrity: Integrity Constraint

- An Integrity Constraint is a declarative method of defining a rule for a column of a table.

Example of Data Integrity:

- Assume that you define an "Integrity Constraint" for the Staff_Sal column of the Staff_Master table.
- This Integrity Constraint enforces the rule that "no row in this table can contain a numeric value greater than 10,000 in this column".
- If an INSERT or UPDATE statement attempts to violate this "Integrity Constraint", Oracle rolls back the statement and returns an "information error" message.

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,...))
```

Applying Constraints:

In Oracle you can apply constraints

Column Level - References a single column and is defined within a specification for the owning column; can be any type of integrity constraint

Table Level - References one or more columns and is defined separately from the definitions of the columns in the table; can define any constraints except NOT NULL

Types of Integrity Constraints



Let us see the types of Data Integrity Constraints:

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

Types of Data Integrity:

- Oracle supports the following Integrity Constraints:
 - NOT NULL constraints for the rules associated with nulls in a column. “Null” is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a “null” value (the absence of a value) in that column.
 - UNIQUE key constraints for the rule associated with unique column values. A “unique value” rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a “unique value” in that column (or set of columns).
 - PRIMARY KEY constraints for the rule associated with primary identification values. A “primary key” value rule defined on a key (a column or set of columns) specifies that “each row in the table can be uniquely identified by the values in the key”.
 - FOREIGN KEY constraints for the rules associated with referential integrity. A “Referential Integrity” rule defined on a key (a column or set of columns) in one table guarantees that “the values in that key, match the values in a key in a related table (the referenced value)”. Oracle currently supports the use of FOREIGN KEY integrity constraints to define the referential integrity actions, including:
 - update and delete No Action
 - delete CASCADE
 - delete SET NULL
 - CHECK constraints for complex integrity rules

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 );
```

NOT NULL constraint:

- Often there may be records in a table that do not have values for every field.
 - This could be because the information is not available at the time of the data entry or because the field is not applicable in every case.
- If the column is created as NULLABLE, in the absence of a user-defined value the DBMS will place a NULL value in the column.
- A NULL value is different from a blank or a zero. It is used for a quantity that is “unknown”.
- “Null” is a rule defined on a single column that allows or disallows, inserts or updates on rows containing a “null” value (the absence of a value) in that column.
- A NULL value can be inserted into a column of any data type.
- **Principles of NULL values:**
 - Setting a NULL value is appropriate when the “actual value” is unknown, or when a value is not meaningful.
 - A NULL value is not equivalent to the value of “zero” if the data type is number, and it is not equivalent to “spaces” if the data type is character.
 - A NULL value will evaluate to NULL in any expression
 - **For example:** NULL multiplied by 10 is NULL.
 - NULL value can be inserted into columns of any data type.
 - If the column has a NULL value, Oracle ignores any UNIQUE, FOREIGN KEY, and CHECK constraints that may be attached to the column.

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 )) ;
```

UNIQUE constraint:

- The UNIQUE constraint does not allow duplicate values in a column.
 - If the UNIQUE constraint encompasses two or more columns, then two equal combinations are not allowed.
 - However, if a column is not explicitly defined as NOT NULL, then NULLS can be inserted multiple number of times.
- A UNIQUE constraint can be extended over multiple columns.
- A “unique value” rule defined on a column (or set of columns) allows inserting or updating a row only if it contains a “unique value” in that column (or set of columns).

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)
.....);
```

PRIMARY KEY constraint:

- On a technical level, a PRIMARY KEY combines a UNIQUE constraint and a NOT NULL constraint.
- Additionally, a table can have at the most one PRIMARY KEY.
- After creating a PRIMARY KEY, it can be referenced by a FOREIGN KEY.
- A “primary key” value rule defined on a key (a column or set of columns) specifies that “each row in the table can be uniquely identified by the values in the key”.
- The example on the slide defines the primary key constraint at the column level. The same example is seen below with the constraint defined at table level

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

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),
  ....) ;
```

CHECK constraint:

- A CHECK constraint allows to state a minimum requirement for the value in a column.
- If more complicated requirements are desired, an INSERT trigger must be used.

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) );
```

FOREIGN KEY Constraint Keywords:

- Given below are a few Foreign Key constraint keywords:
 - FOREIGN KEY: Defines the column in the child table at the table constraint level.
 - REFERENCES: Identifies the table and column in the parent table.
 - ON DELETE CASCADE: Deletes the dependent rows in the child table when a row in the parent table is deleted.
 - ON DELETE SET NULL: Converts dependent FOREIGN KEY values to NULL.
- You can query the USER_CONSTRAINTS table to view all constraint definitions and names.
- You can view the columns associated with the constraint names in the USER_CONS_COLUMNS view.
- In EMP table, for deptno column, if we want to allow only those values that already exist in deptno column of the DEPT table, we must enforce what is known as REFERENTIAL INTEGRITY. To enforce REFERENTIAL INTEGRITY, declare deptno field of DEPT table as PRIMARY KEY, and deptno field of EMP table as FOREIGN KEY as follows

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
```

Creating new table based on an existing table:

- The example in the slide shows how to create a new table based on an existing table.
 - When the new table will be created, it will contain student information from department 10.
 - Constraints on an old table will not be applicable for the new table except NOT NULL constraint.

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];
```

Examples of ALTER TABLE:

- Table_name must be an existing table.
- A column can be removed from an existing table by using ALTER TABLE.
- The uses of modifying columns are:
 - Can increase the width of a character column, any time.
 - Can increase the number of digits in a number, any time.
 - Can increase or decrease the number of decimal places in a number column, any time. Any reduction on precision and scale can be on empty columns only.
 - Can add only NOT NULL constraint by using Column constraints. All other constraints have to be specified as Table constraints.

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:

- The ADD clause allows to add a column or constraint. You can also add multiple columns in one statement separated by comma.
- A column with constraints can also be added as shown in the following example:

```
ALTER TABLE Department_Master  
ADD (dept_name varchar2(10) NOT NULL);
```

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- Modify Clause

The use of modifying the columns with the Enable | Disable clause are:

- Can increase "column width" of a character any time.
- Can increase the "number of digits" in a number at any time.
- Can increase or decrease the "number of decimal places" in a number column at any time. Any reduction on "precision" and "scale" can only be on empty columns.
- Can only add the NOT NULL constraint by using "column constraints". Rest all other constraints have to be specified as "table constraints".

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 ;
```

ALTER TABLE – Drop Clause

- To remove the PRIMARY KEY constraint on the DEPARTMENT table and drop the associated FOREIGN KEY constraint on the DEPT_CODE column.

```
ALTER TABLE department_master  
DROP PRIMARY KEY CASCADE;
```

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;
```

Ways for "Dropping" a column:

1. **Marking the columns as "Unused" and then later dropping them:**

- Oracle onwards a new feature to "drop" and "set" the "unused columns" in a table is added
 - First command as shown in the slide, allows you to mark a column as unused and
 - Second command as shown in the following slide, lets you drop the unused column from the table to create more free space.
 - This feature drops the column from a table and releases any space back to the segment.
- **Note that:**
 - Columns once marked as unused cannot be recovered.
 - Marking the columns as unused does not release the space occupied by them back to the database.
 - Until you actually drop these columns, they continue to count towards the absolute limit of 1000 columns per table.
 - If you mark a column of data type LONG as UNUSED, you cannot add another LONG column to the table until you actually drop the unused LONG column.
- The advantage of the marking column as "unused" and then dropping them is that marking the columns is much faster process than dropping the columns.
- You can refer to the data dictionary table USER_UNUSED_COL_TABS to get information regarding the tables with columns marked as unused.

Dropping Column



- Directly dropping the columns.

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

Ways for “Dropping” a column (contd.):

2. DROP COLUMN:

- This feature allows directly dropping the column.
- For DROP COLUMN command shown in the slide, to work successfully, the table should be exclusively locked by the user by giving the command.
 - All “indexes” defined on any of the target columns are also dropped.
 - All “constraints” that reference a target column are removed.
- Note that this command should be used with caution.
- The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
- The CASCADE CONSTRAINTS clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
- The CASCADE CONSTRAINTS clause also drops all multicolumn constraints defined on the dropped columns

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;
```

Deleting Database Objects: Tables

- Deleting objects that exist in the database is an easy task. Just say:

```
DROP Obj_Type obj_name;
```

- A table that is dropped cannot be recovered. When a table is dropped, dependent objects such as indexes are automatically dropped. Synonyms and views created on the table remain, but give an error if they are referenced.
- You cannot delete a table that is being referenced by another table. To do so use the following:

```
DROP TABLE table-name 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;
```

Renaming Database Objects: Tables

- Renaming objects that exist in the database is an easy task. Just say:

```
RENAME Existing_TableName TO new_tablename;
```

- You can RENAME a table that is being referenced by another table.

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 ;
```

Truncating Database Objects: Tables

- Truncating objects that exist in the database is an easy task. Just say:

```
TRUNCATE TABLE Existing_TableName
```

7.9: Tips and Tricks



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

7.9: Tips and Tricks



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



In this lesson, you have learnt:

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



Review – Questions



Question 1: Indexes can be created _____ or _____

