SQL: CREATE

Creating Tables in Oracle

Prerequisites

For a user to be able to create a table, he needs the **create table** system privilege, otherwise he'll receive the ORA-01031: insufficient privileges error message.
Additionally, the user needs to have enough quota on the tablespace where he wants to create the table.

How To Create a New Table?

If you want to create a new table in your own schema, you can log into the server with your account, and use the CREATE TABLE statement. The following scripts shows you how to create a table:

Creating Tables

CREATE TABLE

Purpose

To create a *table,* the basic structure to hold user data, specifying the following information:

- column definitions
- table organization definition
- column definitions using objects
- integrity constraints
- the table's tablespace
- storage characteristics

The typical usage is:

CREATE [TEMP[ORARY]] TABLE *[table name]*

(*[column definitions]*) *[table parameters]*.

Column Definitions: A comma-separated list consisting of any of the following

- Column definition:

[column name] [data type] {NULL | NOT NULL} {column options}

- Primary key definition:

PRIMARY KEY (*[comma separated column list]*)

- CONSTRAINTS:

{CONSTRAINT} [constraint definition]

Syntax:

```
CREATE TABLE  table_name
    ( column 1  data_type_for_column_1,
      column 2   data_type_for_column_2,
      ...
    );
```

For example, the command to create a table named **employees** with a few sample columns would be:

```
CREATE TABLE employees (
   id             INTEGER   PRIMARY KEY,
   first_name     CHAR(50)  null,
   last_name      CHAR(75)  not null,
   date_of_birth DATE       null
);
```

The following are examples of field types:

INTEGER	A whole number
VARCHAR2(10)	Up to 10 characters.
CHAR(10)	Fixed number of characters
DATE	A date
DATETIME	Date and time
FLOAT	Floating point numbers
Specific to Oracle	
CLOB	Allows large character fields.
NUMBER(10,2)	Up to 10 digits before the point, 2 after.

Heap tables

When we refer to tables we refer to heap tables. They are simple tables without constraints. We will learn about constratints later. A heap table is created as follows:

```
----------------------------------------------------------------------------------------
CREATE TABLE emp (

empno NUMBER(4) ,

ename VARCHAR2(10) ,

job VARCHAR2(9),

mgr NUMBER(4),

hiredate DATE ,

sal NUMBER(7,2),

comm NUMBER(7,2),

deptno NUMBER(2)

);
```

```
CREATE TABLE DEPT
        (DEPTNO NUMBER(2),
         DNAME VARCHAR2(14)
        );
```

```
-----------------------------------------------
CREATE TABLE SALGRADE
       ( GRADE NUMBER,
         LOSAL NUMBER,
         HISAL NUMBER );
```

```
-----------------------------------------------
CREATE TABLE BONUS
         (
         ENAME VARCHAR2(10),
         JOB VARCHAR2(9),
         SAL NUMBER,
         COMM NUMBER
         );
```

create the constraints

Create a table with primary key

It is possible to create the **constraints** together with the create statement. As a foreign key references a known type, it is not necessary to specify the foreign key's column type.

```
create table orders (
  order_id   number primary key
  order_dt   date,
  cust_id    references customers
);
```

A primary key needs to have an associated (unique) index.

```
create table orders (
  order_id number Primary Key,
  order_dt date,
  cust_id  references customer

 );
```

```
CREATE TABLE supplier
   ( supplier_id    number(10)        not null,
     supplier_name  varchar2(50)      not null,
     contact_name   varchar2(50)
     CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
    );
```

```
Table created.
```

```
desc supplier;
```

```
Name                 Null?                Type
-------------------  --------             --------------
SUPPLIER_ID          NOT NULL             NUMBER(10)
SUPPLIER_NAME        NOT NULL             VARCHAR2(50)
CONTACT_NAME                              VARCHAR2(50)
```

```
drop table supplier;
```

```
Table dropped.
```

Creating Table with combined primary key

```
create table employee_history
    (employee_id       number(6) not null,
     salary            number(8,2),
     hire_date         date default sysdate,
     termination_date  date,
     termination_desc varchar2(4000),
     constraint emphistory_pk
      primary key (employee_id, hire_date)
    );
```

```
Table created.
```

```
drop table employee_history;
```

```
Table dropped.
```

NOT NULL

```
CREATE TABLE Customer
(SID number(10) NOT NULL,
Last_Name varchar2 (30) NOT NULL,
First_Name varchar2(30));
```

UNIQUE

```
CREATE TABLE Customer
(SID number(10) Unique,
Last_Name varchar2 (30),
First_Name varchar2(30));
```

CHECK

```
CREATE TABLE Customer
(SID number(10) CHECK (SID > 0),
Last_Name varchar2 (30),
First_Name varchar2(30));
```

Default Values

```
CREATE TABLE customer
(First_Name varchar2(50),
Last_Name varchar2(50),
Address varchar2(50) default
'Unknown', City char(50) default
'Ankara', Country char(25),
Birth_Date  date
);
```

Create table with foreign key

```
CREATE TABLE supplier
   ( supplier_id    number(10)        not null,
     supplier_name  varchar2(50)      not null,
     contact_name   varchar2(50)
     CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
   );
```

```
Table created.
```

```
CREATE TABLE products
        (      product_id     number(10)        not null,
               supplier_id    number(10)        not null,
               CONSTRAINT fk_supplier
                 FOREIGN KEY (supplier_id)
                 REFERENCES supplier(supplier_id)
        );
```

```
Table created.
```

```
desc products;
```

```
 Name Null?    Type
 ---------------

 PRODUCT_ID        NOT NULL           NUMBER(10)
 SUPPLIER_ID       NOT NULL           NUMBER(10)
```

```
desc supplier;
```

```
 Name              Null?              Type
 ---------------   ---------------    -----------------
 SUPPLIER_ID       NOT NULL           NUMBER(10)
 SUPPLIER_NAME     NOT NULL           VARCHAR2(50)
 CONTACT_NAME                         VARCHAR2(50)
```

```
drop table products cascade constraints;
```

```
Table dropped.
```

```
drop table supplier cascade constraints;
```

```
Table dropped.
```

create as select

SQL: CREATE Table from another table

You can also create a table from an existing table by copying the existing table's columns.

It is important to note that when creating a table in this way, the new table will be populated with the records from the existing table (based on the SELECT Statement).

Syntax #1 - Copying all columns from another table

The basic syntax is:

```
CREATE TABLE new_table
    AS (SELECT * FROM old_table);
```

Example:

```
CREATE TABLE suppliers
  AS (SELECT *
        FROM companies
        WHERE id > 1000);
```

This would create a new table called **suppliers** that included all columns from the **companies** table.

If there were records in the **companies** table, then the new suppliers table would also contain the records selected by the SELECT statement.

Syntax #2 - Copying selected columns from another table

The basic syntax is:

```
CREATE TABLE new_table
  AS (SELECT column_1, column2, ... column_n FROM old_table);
```

Example:

```
CREATE TABLE suppliers
  AS (SELECT id, address, city, state, zip
        FROM companies
        WHERE id > 1000);
```

This would create a new table called **suppliers**, but the new table would only include the specified columns from the **companies** table.

Again, if there were records in the **companies** table, then the new suppliers table would also contain the records selected by the SELECT statement.

Syntax #3 - Copying selected columns from multiple tables

The basic syntax is:

```
CREATE TABLE new_table
  AS (SELECT column_1, column2, ... column_n
        FROM old_table_1, old_table_2, ... old_table_n);
```

Example:

```
CREATE TABLE suppliers
  AS (SELECT companies.id, companies.address, categories.cat_type
        FROM companies, categories
        WHERE companies.id = categories.id
        AND companies.id > 1000);
```

This would create a new table called **suppliers** based on columns from both the **companies** and **categories** tables.

```
create table t1 as
   select
     store_name,
     avg(quantity) qty
   from
     store join sales using (store_key)
   group by store_name;
```

```
Table created.
```

```
desc t1;
```

```
Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 STORE_NAME                                         VARCHAR2(40)
 QTY                                                NUMBER
```

Practice 7

1. Create the MY_DEPT table based on the following table instance chart. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	Number	VARCHAR2
Length	7	25

2. Create the MY_EMP table based on the following table instance chart. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	Number	VARCHAR2	VARCHAR2	Number
Length	7	25	25	7

3. Modify the EMP table to allow for longer employee last names. Confirm your modification.

4. Create the MY_EMP2 table based on the structure of the MY_EMP table. Include the ID, FIRST_NAME, LAST_NAME, and DEPT_ID columns.

5. Drop the MY_EMP table

6. RENAME the table MY_EMP2 TO MY_EMP;

7. Add a table-level PRIMARY KEY constraint to the MY_EMP table on the ID column. The constraint should be named at creation. Name the constraint my_emp_id_pk

8. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my_deptid_pk.

9. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my_emp_dept_id_fk.