# INDEX Constraint

DBMS-LAB -9

The INDEX is used to create and retrieve data from the database very quickly. Index can be created by using single or group of columns in a table. When index is created, it is assigned a ROWID for each row before it sorts out the data.
For example, the following SQL creates a new table called CUSTOMERS and adds five columns:

```
CREATE TABLE CUSTOMERS(
       ID   INT              NOT NULL,
       NAME VARCHAR (20)     NOT NULL,
       AGE  INT              NOT NULL,
       ADDRESS  CHAR (25) ,
       SALARY   DECIMAL (18, 2),
       PRIMARY KEY (ID)
);
```

Now, you can create index on single or multiple columns using the following syntax:

```
CREATE INDEX index_name
    ON table_name ( column1, column2.....);
```

To create an INDEX on AGE column, to optimize the search on customers for a particular age, following is the SQL syntax:

```
CREATE INDEX idx_age
    ON CUSTOMERS ( AGE );
```

## DROP an INDEX Constraint:

To drop an INDEX constraint, use the following SQL:

```
ALTER TABLE CUSTOMERS
   DROP INDEX idx_age;
```

## Index in Oracle

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. By default, Oracle creates B-tree indexes.

## Create an Index

## Syntax

The syntax for creating an index in Oracle/PLSQL is:

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (column1, column2, ... column_n)
  [ COMPUTE STATISTICS ];
```

UNIQUE
: It indicates that the combination of values in the indexed columns must be unique.

index_name
: The name to assign to the index.

# INDEX Constraint

table_name
: The name of the table in which to create the index.

column1, column2, ... column_n
: The columns to use in the index.

COMPUTE STATISTICS
: It tells Oracle to collect statistics during the creation of the index. The statistics are then used by the optimizer to choose a "plan of execution" when SQL statements are executed.

## Example

Let's look at an example of how to create an index in Oracle/PLSQL.

For example:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name);
```

In this example, we've created an index on the supplier table called supplier_idx. It consists of only one field - the supplier_name field.

We could also create an index with more than one field as in the example below:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name, city);
```

We could also choose to collect statistics upon creation of the index as follows:

```
CREATE INDEX supplier_idx
  ON supplier (supplier_name, city)
  COMPUTE STATISTICS;
```

## Create a Function-Based Index

In Oracle, you are not restricted to creating indexes on only columns. You can create function-based indexes.

## Syntax

The syntax for creating a function-based index in Oracle/PLSQL is:

```
CREATE [UNIQUE] INDEX index_name
  ON table_name (function1, function2, ... function_n)
  [ COMPUTE STATISTICS ];
```

UNIQUE
: It indicates that the combination of values in the indexed columns must be unique.

index_name
: The name to assign to the index.

table_name
: The name of the table in which to create the index.

function1, function2, ... function_n
: The functions to use in the index.

# INDEX Constraint

COMPUTE STATISTICS
: It tells Oracle to collect statistics during the creation of the index. The statistics are then used by the optimizer to choose a "plan of execution" when SQL statements are executed.

## Example

Let's look at an example of how to create a function-based index in Oracle/PLSQL.

For example:

```
CREATE INDEX supplier_idx
  ON supplier (UPPER(supplier_name));
```

In this example, we've created an index based on the uppercase evaluation of the *supplier_name* field.

However, to be sure that the Oracle optimizer uses this index when executing your SQL statements, be sure that UPPER(supplier_name) does not evaluate to a NULL value. To ensure this, add **UPPER(supplier_name) IS NOT NULL** to your WHERE clause as follows:

```
SELECT supplier_id, supplier_name, UPPER(supplier_name)
FROM supplier
WHERE UPPER(supplier_name) IS NOT NULL
ORDER BY UPPER(supplier_name);
```

## Rename an Index

## Syntax

The syntax for renaming an index in Oracle/PLSQL is:

```
ALTER INDEX index_name
  RENAME TO new_index_name;
```

index_name
: The name of the index that you wish to rename.

new_index_name
: The new name to assign to the index.

## Example

Let's look at an example of how to rename an index in Oracle/PLSQL.

For example:

```
ALTER INDEX supplier_idx
  RENAME TO supplier_index_name;
```

In this example, we're renaming the index called *supplier_idx* to *supplier_index_name*.

# INDEX Constraint

DBMS-LAB -9

## Collect Statistics on an Index

If you forgot to collect statistics on the index when you first created it or you want to update the statistics, you can always use the ALTER INDEX command to collect statistics at a later date.

## Syntax

The syntax for collecting statistics on an index in Oracle/PLSQL is:

```
ALTER INDEX index_name
  REBUILD COMPUTE STATISTICS;
```

index_name
: The index in which to collect statistics.

## Example

Let's look at an example of how to collect statistics for an index in Oracle/PLSQL.

For example:

```
ALTER INDEX supplier_idx
  REBUILD COMPUTE STATISTICS;
```

In this example, we're collecting statistics for the index called supplier_idx.

## Drop an Index

## Syntax

The syntax for dropping an index in Oracle/PLSQL is:

```
DROP INDEX index_name;
```

index_name
: The name of the index to drop.

## Example

Let's look at an example of how to drop an index in Oracle/PLSQL.

For example:

```
DROP INDEX supplier_idx;
```

In this example, we're dropping an index called supplier_idx.

**Question 1:** Assume customer order system with the following three tables:

**Customer:** Contains the details of various customers such that each customer has been assigned a

# INDEX Constraint

## DBMS-LAB -9

UNIQUE Identification named Customer_Id.

**Product:** Contains the details of various products manufactured in the company. Each product has a unique identification number called Product_Code.

**Order:** This table contains the various information about the various orders received from customers. The first two charaters of product code indicate the type the type of product. For example CD indicate CD Player.

Customer(Customer_Id, Customer_Name, City, Pincode, State, Balance_Due);
Product(Product_Code, Product_Name, Oty_Availabel, Cost_Price, Selling_Price);
Order(Order_No, Order_Date, Customer_Id, Product_Code, Quantity);

a. Create Unique index on customer_id of Customer table.
b. Create Unique index on Order_No, Customer_Id and Product CD of Order table.

**Question 2: Assume employees table**

```
CREATE TABLE employees
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
         CONSTRAINT emp_last_name_nn_demo NOT NULL
    , email          VARCHAR2(25)
         CONSTRAINT emp_email_nn_demo     NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE  DEFAULT SYSDATE
         CONSTRAINT emp_hire_date_nn_demo  NOT NULL
    , job_id         VARCHAR2(10)
       CONSTRAINT     emp_job_nn_demo  NOT NULL
    , salary         NUMBER(8,2)
       CONSTRAINT     emp_salary_nn_demo  NOT NULL
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
    , dn             VARCHAR2(300)
    , CONSTRAINT     emp_salary_min_demo
                     CHECK (salary > 0)
    , CONSTRAINT     emp_email_uk_demo
                     UNIQUE (email)
    ) ;
```

Create an index by executing following queury:

```
CREATE INDEX income_ix
   ON employees(salary + (salary*commission_pct));
```

Use **income_ix** index to dispaly first name and last name of employees where **salary*commission_pct) + salary > 15000** from employees table.