|  |  |  |
| --- | --- | --- |
| **Data Types** | **Length (Bytes)** | **Range of values** |
| BYTEINT | 1 | -128 to +127 |
| SMALLINT | 2 | -32768 to +32767 |
| INTEGER | 4 | -2,147,483,648 to +2147,483,647 |
| BIGINT | 8 | -9,233,372,036,854,775,80 8 to +9,233,372,036,854,775,8 07 |
| DECIMAL | 1-16 |  |
| NUMERIC | 1-16 |  |
| FLOAT | 8 | IEEE format |
| CHAR | Fixed Format | 1-64,000 |
| VARCHAR | Variable | 1-64,000 |
| DATE | 4 | YYYYMMDD |
| TIME | 6 or 8 | HHMMSS.nnnnnn **or**HHMMSS.nnnnnn+HHMM |
| TIMESTAMP | 10 or 12 | YYMMDDHHMMSS.nnnnnn **or**YYMMDDHHMMSS.nnnnnn +HHMM |

Database

Database is a collection of logically related data. They are accessed by many users for different purposes. For example, a sales database contains entire information about sales which is stored in many tables.

Tables

Tables are the basic unit in RDBMS where the data is stored. A table is a collection of rows and columns. Following is an example of emp1 table.

CREATE TABLE emp1(

EmployeeNo INTEGER,

FirstName VARCHAR(20),

LastName VARCHAR(20),

BirthDate DATE FORMAT 'DD/MM/YYYY');

insert into emp1 values(101,'Mike','James','01/05/1980');

insert into emp1 values(102,'Alex','Stuart','11/06/1984');

insert into emp1 values(103,'Robert','Williams','03/05/1983');

insert into emp1 values(104,'Robert','James','12/01/1984');

insert into emp1 values(105,'Peter','Paul','04/10/1983');

insert into emp1 values(106,'Dev','Singh','12/**1**/1984');

\*\*\* Failure 3535 A character string failed conversion to a numeric value.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

insert into emp1 values(106,'Dev','Singh','12/**01**/1984');

insert into emp1 values(107,'Ram','Singh',current\_date);

insert into emp1 values(108,'Ravi','Verma',cast('25/10/1990' as date format 'dd/mm/yyyy'));

select \* from emp1 order by 1;

EmployeeNo FirstName LastName BirthDate

----------- -------------------- -------------------- ----------

101 Mike James 01/05/1980

102 Alex Stuart 11/06/1984

103 Robert Williams 03/05/1983

104 Robert James 12/01/1984

105 Peter Paul 04/10/1983

106 Dev Singh 12/01/1984

107 Ram Singh 20/08/2018

108 Ravi Verma 25/10/1990

Teradata supports the following table types to hold temporary data.

* Derived Table (Inline View in Oracle)
* Volatile Table (Local temporary table)
* Global Temporary Table (Same as Oracle)

## Derived Table

Derived tables are created, used and dropped within a query. These are used to store intermediate results within a query.

### Example

The following example builds a derived table EmpSal with records of employees with salary greater than 75000.

select empno, ename, sal from emp,(select \* from dept);

$

\*\*\* Failure 3707 Syntax error, expected something like a name or a Unicode

delimited identifier or an 'UDFCALLNAME' keyword between ')' and ';'.

Statement# 1, Info =58

\*\*\* Total elapsed time was 1 second.

**NOTE: Without table alias query will throw error.**

select empno, ename, sal from emp,(select \* from dept) **as dept**

where emp.deptno = dept.deptno;

\*\*\* Query completed. 11 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

empno ename sal

----------- ---------- -----------

7369 SMITH 800

7900 JAMES 950

7782 CLARK 2450

7567 JONES 2975

7698 BLAKE 2850

7839 KING 5000

7901 FORD 3000

7934 MILLER 1300

7902 FORD 3000

7788 SCOTT 3000

7566 JONES 2975

## Volatile Table (Temporary table in Oracle)

Volatile tables are created, used and dropped within a user session. Their definition is not stored in data dictionary. They hold intermediate data of the query which is frequently used. Following is the syntax.

### Syntax

CREATE [SET|MULTISET] VOALTILE TABLE tablename

<table definitions>

<column definitions>

<index definitions>

ON COMMIT [DELETE|PRESERVE] ROWS

### Example

CREATE VOLATILE TABLE dept\_stat(

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no)

ON COMMIT PRESERVE ROWS;

When the above query is executed, it produces the following output.

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

insert into dept\_stat values(10,5000,50000,3000);

insert into dept\_stat values(20,1000,80000,5000);

insert into dept\_stat values(30,3000,90000,5000);

Commit work;

$

\*\*\* Failure 3706 Syntax error: COMMIT WORK not allowed for a DBC/SQL session.

Statement# 1, Info =8

\*\*\* Total elapsed time was 1 second.

**NOTE**: Depending upon the setting of the Session Mode, the executer will have to write "COMMIT WORK" statement, or the Teradata database will implicitly commit each statement that is processed.

select \* from dept\_stat;

\*\*\* Query completed. 3 rows found. 4 columns returned.

\*\*\* Total elapsed time was 1 second.

dept\_noavg\_salarymax\_salarymin\_salary

----------- ----------- ----------- -----------

20 1000 80000 5000

30 3000 90000 5000

10 5000 50000 3000

.quit

Login again and check the table.

C:\Users\haryadav>bteq

Teradata BTEQ 15.10.01.01 for WIN32. PID: 4264

Copyright 1984-2016, Teradata Corporation. ALL RIGHTS RESERVED.

Enter your logon or BTEQ command:

.logon 10.74.161.248/td\_user3

.logon 10.74.161.248/td\_user3

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.10.07.12

\*\*\* Teradata Database Version is 13.10.07.12

\*\*\* Transaction Semantics are BTET.

\*\*\* Session Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

select \* from dept\_stat;

select \* from dept\_stat;

\*\*\* Failure 3807 Object 'dept\_stat' does not exist.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

**NOTE: Table got droped.**

**How to change the session mode.**

C:\Users\haryadav>bteq

Teradata BTEQ 15.10.01.01 for WIN32. PID: 812

Copyright 1984-2016, Teradata Corporation. ALL RIGHTS RESERVED.

Enter your logon or BTEQ command:

**.set session transaction ansi;**

.set session transaction ansi;

Teradata BTEQ 15.10.01.01 for WIN32. Enter your logon or BTEQ command:

.logon 10.74.161.248/td\_user3

.logon 10.74.161.248/td\_user3

Password:

\*\*\* Logon successfully completed.

\*\*\* Teradata Database Release is 13.10.07.12

\*\*\* Teradata Database Version is 13.10.07.12

**\*\*\* Transaction Semantics are ANSI.**

\*\*\* Session Character Set Name is 'ASCII'.

\*\*\* Total elapsed time was 1 second.

NOTE: After session mode change to ANSI you need to write commit work explicitly, even for DDL statement.

CREATE VOLATILE TABLE dept\_stat (

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no)

ON COMMIT PRESERVE ROWS;

CREATE VOLATILE TABLE dept\_stat (

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no)

ON COMMIT PRESERVE ROWS;

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

insert into dept\_stat values(10,5000,50000,3000);

insert into dept\_stat values(10,5000,50000,3000);

\*\*\* Error 3722 Only a COMMIT WORK or null statement is legal after a DDLStatement.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

**commit;**

commit;

\*\*\* COMMIT done.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

insert into dept\_stat values(10,5000,50000,3000);

insert into dept\_stat values(10,5000,50000,3000);

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

commit;

commit;

\*\*\* COMMIT done.

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

select \* from dept\_stat;

select \* from dept\_stat;

\*\*\* Query completed. One row found. 4 columns returned.

\*\*\* Total elapsed time was 1 second.

dept\_noavg\_salarymax\_salarymin\_salary

----------- ----------- ----------- -----------

10 5000 50000 3000

BTEQ -- Enter your SQL request or BTEQ command:

## Teradata Mode (BTET) vs. ANSI Mode

Teradata systems have the ability to use either of these two modes:

•     Teradata mode

•     ANSI mode

|  |  |
| --- | --- |
| Teradata Mode | ANSI Mode |
| Comparison are NOT CASE Specific | Comparison ARE CASE Specific |
| CREATE TABLE default to SET tables | CREATE TABLE default to MULTISET tables |
| Each Transaction is IMPLICIT automatically | Each Transaction is EXPLICIT automatically |
| BTET (Begin and End Transaction) | BTEQ (no Begin and End Transaction necess.) |

## Global Temporary Table

Global Temporary tables are those tables whose definition is stored in data dictionary and they can be used by many users/sessions. But the data loaded into global temporary table is retained only during the session. You can materialize up to 2000 global temporary tables per session. Following is the syntax.

### Syntax

CREATE [SET|MULTISET] GLOBAL TEMPORARY TABLE tablename

<table definitions>

<column definitions>

<index definitions>

### Example

CREATE SET GLOBAL TEMPORARY TABLE dept\_stat(

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no);

When the above query is executed, it produces the following output.

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

insert into hy\_dept\_stat values(10,5000,50000,3000);

insert into hy\_dept\_stat values(20,1000,80000,5000);

insert into hy\_dept\_stat values(30,3000,90000,5000);

select \* from dept\_stat;

select \* from dept\_stat;

**\*\*\* Query completed. No rows found.**

\*\*\* Total elapsed time was 1 second.

BTEQ -- Enter your SQL request or BTEQ command:

**Shocked? ...**   
Yes. You cannot see the records because Teradata becomes strict to GLOBAL TEMPORARY tables by incorporating ON COMMIT DELETE ROWS by default to add more meaning to TEMPORARY table.   
  
so, if you want to see the records, you have to explicitly write ON COMMIT PRESERVE ROWS immediately after CREATE TABLE syntax.

drop table dept\_stat;

drop table dept\_stat;

\*\*\* Table has been dropped.

\*\*\* Total elapsed time was 1 second.

CREATE SET GLOBAL TEMPORARY TABLE dept\_stat(

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no)

ON COMMIT PRESERVE ROWS;

insert into dept\_stat values(10,5000,50000,3000);

insert into dept\_stat values(20,1000,80000,5000);

insert into dept\_stat values(30,3000,90000,5000);

select \* from dept\_stat;

\*\*\* Query completed. 3 rows found. 4 columns returned.

\*\*\* Total elapsed time was 1 second.

dept\_noavg\_salarymax\_salarymin\_salary

----------- ----------- ----------- -----------

20 1000 80000 5000

30 3000 90000 5000

10 5000 50000 3000

## Table Types

Teradata supports different types of tables.

* **Permanent Table** − This is the default table and it contains data inserted by the user and stores the data permanently.
* **Volatile Table** − The data inserted into a volatile table is retained only during the user session. The table and data is dropped at the end of the session. These tables are mainly used to hold the intermediate data during data transformation.
* **Global Temporary Table** − The definition of Global Temporary table are persistent but the data in the table is deleted at the end of user session.
* **Derived Table (Inline View)** − Derived table holds the intermediate results in a query. Their lifetime is within the query in which they are created, used and dropped.

### Set Versus Multiset

Teradata classifies the tables as SET or MULTISET tables based on how the duplicate records are handled. A table defined as SET table doesn’t store the duplicate records, whereas the MULTISET table can store duplicate records.

CREATE TABLE command is used to create tables in Teradata.

## Syntax

Following is the generic syntax of CREATE TABLE statement.

CREATE <SET/MULTISET> TABLE <Tablename>

<Table Options>

<Column Definitions>

<Index Definitions>;

* **Table Options** − Specifies the physical attributes of the table such as Journal and Fallback.
* **Column Definition** − Specifies the list of columns, data types and their attributes.
* **Index Definition** − Additional indexing options such as Primary Index, Secondary Index and Partitioned Primary Index.

## Example

.set width 32767

.set retlimit 30

SELECT \* FROM dbc.tables WHERE tablekind = 'T' and DatabaseName='tdhari';

\*\*\* Query completed. 18 rows found. 26 columns returned.

\*\*\* Total elapsed time was 1 second.

DatabaseName TableName Version Table

------------------------------ ------------------------------ ------------ -----

tdhari EMPLOYEE 5 T

tdhari emp 1 T

tdhari salary 1 T

tdhari dept 4 T

Creating a Table with a Unique Primary Index.

CREATE TABLE EMP1 (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

UNIQUE PRIMARY INDEX (EMP\_NO);

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

insert emp1 values(1111,10,'Ram','Singh',50000,'Delhi');

insert emp1 values(1112,20,'Dev','Singh',30000,'Pune');

insert emp1 values(1113,30,'Mark','Taylor',60000,'Chennai');

insert emp1 values(1114,20,'John','Brown',40000,'Chennai');

insert emp1 values(1115,10,'Pritam','Singh',50000,'Delhi');

select \* from emp1;

\*\*\* Query completed. 5 rows found. 6 columns returned.

\*\*\* Total elapsed time was 1 second.

EMP\_NO DEPT\_NO FIRST\_NAME LAST\_NAME SAL

----------- ----------- -------------------- -------------------- ---------

1115 10 Pritam Singh 50

1111 10 Ram Singh 50

1113 30 Mark Taylor 60

1112 20 Dev Singh 30

1114 20 John Brown 40

Creating a Table with a Unique Secondary Index

CREATE TABLE EMP1USI (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

UNIQUE PRIMARY INDEX(EMP\_NO)

UNIQUE INDEX(DEPT\_NO);

show table emp1usi;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

----------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1usi ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO )

UNIQUE INDEX ( DEPT\_NO );

CREATE TABLE EMP1NSI (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

UNIQUE PRIMARY INDEX(EMP\_NO)

INDEX(DEPT\_NO);

show table emp1nsi;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1nsi ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO )

INDEX ( DEPT\_NO );

Creating a Table with a Non-Unique Primary Index

CREATE TABLE EMP2 (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

PRIMARY INDEX(EMP\_NO);

show table emp2;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

----------------------------------------------------------------------

CREATE SET TABLE tdhari.emp2 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO );

Creating a Table with a Non-Unique Primary Index and Non-Unique Secondery Index

CREATE TABLE EMP2NPINSI (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

PRIMARY INDEX(EMP\_NO)

INDEX (DEPT\_NO);

show table emp2npinsi;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-----------------------------------------------------------------------

CREATE SET TABLE tdhari.emp2npinsi ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO )

INDEX ( DEPT\_NO );

Creating a Table without entering a Primary Index

CREATE TABLE EMP3 (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100));

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

show table emp3;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE SET TABLE TD\_BIM\_FR\_TRNG\_DB.emp3 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO );

NOTE: By default Non-Unique Primary Index on first column.

Creating a Table with NO Primary Index

CREATE TABLE EMP4 (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

NO PRIMARY INDEX;

show table emp4;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------------------------

CREATE MULTISET TABLE TD\_BIM\_FR\_TRNG\_DB.emp4 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

NO PRIMARY INDEX ;

Creating a Set Table

CREATE SET TABLE EMP5(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

PRIMARY INDEX(EMP\_NO);

show table emp5;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------------------------

CREATE SET TABLE TD\_BIM\_FR\_TRNG\_DB.emp5 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO );

insert into emp5 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

insert into emp5 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Failure 2802 Duplicate row error in TD\_BIM\_FR\_TRNG\_DB.EMP5.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

insert into emp5 values(**1001**,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

insert into emp5 values(1001,20,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

Creating a Multiset Table with Non-Unique Primary Index

CREATE MULTISET TABLE EMP6(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

PRIMARY INDEX(EMP\_NO);

show table emp6;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------

CREATE MULTISET TABLE TD\_BIM\_FR\_TRNG\_DB.emp6 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO );

insert into emp6 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

insert into emp6 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

**NOTE**: This table has created Non-Unique Primary Index, so duplicate value inserted in that column as well as entire rows can be duplicate.

Creating a Multiset Table with Unique Primary Index

CREATE MULTISET TABLE EMP7(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

UNIQUE PRIMARY INDEX(EMP\_NO);

show table emp7;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE MULTISET TABLE TD\_BIM\_FR\_TRNG\_DB.emp7 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

DEFAULT MERGEBLOCKRATIO

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

insert into emp7 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Insert completed. One row added.

\*\*\* Total elapsed time was 1 second.

insert into emp7 values(1000,10,'Ram','Singh',50000,'New Delhi');

\*\*\* Failure 2801 Duplicate unique prime key error in TD\_BIM\_FR\_TRNG\_DB.EMP7.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

**NOTE**: This table has created Unique Primary Index, so duplicate value is not inserted.

Creating a Table with a Multicolumn Primary Index

CREATE TABLE EMP8 (

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100))

PRIMARY INDEX(EMP\_NO,DEPT\_NO,SALARY);

show table emp8;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp8 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO ,DEPT\_NO ,SALARY );

Table options are:

Freespace % of freespace (0-75%) that will remain on a cylinder during subsequent loading operations

Datablocksize Size of data block in which row(s) of the table are stored

FALLBACK: FALLBACK is TERADATA mechanism to save table data in case of AMP Failure. If you define any table with FALLBACK option , a duplicate copy of table data is maintained in some other AMP. SO in case of AMP Failure, the FALLBACK AMP is used. For critical tables, it is recommended to use FALLBACK option. However it also comes with more Storage Space utilisation disadvantage. In TERADATA, default is NO FALLBACK.

PERMANENT JOURNALS:There are four types of Permanent Journals you can use: BEFORE, AFTER, DUAL BEFORE, DUAL AFTER. The BEFORE JOURNAL holds the image of impacted rows before any changes are made. AFTER JOURNAL holds the image of affected rows after changes are done. In DUAL BEFORE/AFTER Journal, two images are taken and are stored in two different AMP’s.

Column definitions

Data type Declare the column to be a character, byte, numeric, or graphic data type.

Data type Attributes Specify DEFAULT, FORMAT, TITLE, NULL,CASESPECIFIC, UPPERCASE.

Column Storage Compress NULL values or a specified value.

Column-level Constraint Specify the single column as a primary key or foreign key.

Specify the single column as unique (must be NOT NULL).

Specify constraint conditions on the column

Example :

CREATE SET TABLE emp ,

NO FALLBACK

(

EmpNo SMALLINT FORMAT '999' NOT NULL,

Name VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

Salary DECIMAL(8,2),

City VARCHAR(12) CHARACTER SET LATIN NOT CASESPECIFIC,

State VARCHAR(2) CHARACTER SET LATIN NOT CASESPECIFIC,

doj DATE FORMAT 'YYYY-MM-DD'

)

UNIQUE PRIMARY INDEX ( EmpNo );

CREATE TABLE EMP9

,NO FALLBACK

,DUAL BEFORE JOURNAL

,DUAL AFTER JOURNAL

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100)

)

UNIQUE PRIMARY INDEX(EMP\_NO) ;

\*\*\* Failure 3566 Database does not have a PERMANENT journal.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

MODIFY DATABASE tdhari AS DEFAULT JOURNAL TABLE = tdharijrnl;

\*\*\* Database/User has been modified.

\*\*\* Total elapsed time was 1 second.

CREATE TABLE EMP9

,NO FALLBACK

,DUAL BEFORE JOURNAL

,DUAL AFTER JOURNAL

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100)

)

UNIQUE PRIMARY INDEX(EMP\_NO) ;

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

Creating a Table with a Journal Keyword Alone.

CREATE TABLE EMP10

,NO FALLBACK

,JOURNAL

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100)

)

UNIQUE PRIMARY INDEX(EMP\_NO) ;

Creating a Table with Customization of the Data Block Size and FREESPACE Percent

CREATE TABLE EMP11

,NO FALLBACK

,JOURNAL

,DATABLOCKSIZE= 21248 BYTES

,FREESPACE = 25 PERCENT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20),

LAST\_NAME CHAR(20),

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100)

)

UNIQUE PRIMARY INDEX(EMP\_NO) ;

SHOW TABLE EMP11;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.EMP11 ,NO FALLBACK ,

BEFORE JOURNAL,

NOT LOCAL AFTER JOURNAL,

WITH JOURNAL TABLE = TDHARI.tdharijrnl ,

DATABLOCKSIZE = 21504 BYTES, FREESPACE = 25 PERCENT, CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

ALTER TABLE tdhari.emp11, MINIMUM DATABLOCKSIZE ;

\*\*\* Table has been modified.

\*\*\* Total elapsed time was 1 second.

SHOW TABLE EMP11;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.EMP11 ,NO FALLBACK ,

BEFORE JOURNAL,

NOT LOCAL AFTER JOURNAL,

WITH JOURNAL TABLE = TDHARI.tdharijrnl ,

DATABLOCKSIZE = 9216 BYTES, FREESPACE = 25 PERCENT, CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

ALTER TABLE tdhari.emp11, MAXIMUM DATABLOCKSIZE IMMEDIATE ;

SHOW TABLE EMP11;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.EMP11 ,NO FALLBACK ,

BEFORE JOURNAL,

NOT LOCAL AFTER JOURNAL,

WITH JOURNAL TABLE = TDHARI.tdharijrnl ,

DATABLOCKSIZE = 130560 BYTES, FREESPACE = 25 PERCENT, CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

**Creating and Dropping NUSI**

NUSI (Non Unique Secondary Indexes)

Used to improve performance of queries that use the column or columns in WHERE clause.

NUSI can be created in following manners:

**When creating the table using CREATE TABLE syntax**

**Using CREATE INDEX after the table is created.**

**NUSI can be dropped at any moment of time.**

Syntax for creating index while creating table

CREATE SET TABLE EMPLOYEE ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT,

LOG

(

Employeeid INTEGER,

DepartmentNo INTEGER,

Salary DECIMAL(8,2),

Hiredate DATE FORMAT 'YYYY-MM-DD')

PRIMARY INDEX ( Employeeid )

INDEX(DEPARTMENTNO);

Syntax for creating index using CREATE INDEX

CREATE INDEX (DEPARTMENTNO) ON EMPLOYEE;

Syntax for dropping index using DROP INDEX

DROP INDEX (DEPARTMENTNO) ON EMPLOYEE;

Creating a Unique Secondary Index (USI) after a table is created

CREATE UNIQUE INDEX idx\_emp1\_empno(emp\_no) ON EMP1;

CREATE UNIQUE INDEX idx\_emp1\_empno(emp\_no) ON EMP1;

\*\*\* Failure 3534 Another index already exists, using the same columns and the same ordering.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

NOTE: emp\_no column already having UNIQUE primary index. But if table have non-unique primary index then unique secondary index can be created.

CREATE UNIQUE PRIMARY INDEX idx\_emp1\_deptno(DEPT\_NO) ON EMP1;

CREATE UNIQUE PRIMARY INDEX idx\_emp1\_deptno(DEPT\_NO) ON EMP1;

\*\*\* Failure 3525 The user cannot create or drop a PRIMARY index.

Statement# 1, Info =62

\*\*\* Total elapsed time was 1 second.

NOTE: A table cannot have two primary index.

Creating a Unique Secondary Index (USI) or (NUSI) after a table is created on dept\_no column.

CREATE UNIQUE INDEX idx\_emp1\_deptno(DEPT\_NO) ON EMP1;

CREATE UNIQUE INDEX idx\_emp1\_deptno(DEPT\_NO) ON EMP1;

\*\*\* Index has been created.

\*\*\* Total elapsed time was 1 second.

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO )

UNIQUE INDEX idx\_emp1\_deptno ( DEPT\_NO );

NOTE: You can't drop a Primary Index in Teradata.

drop unique primary index ( emp\_no ) on emp1;

drop UNIQUE PRIMARY INDEX ( EMP\_NO ) on emp1;

\*\*\* Failure 3525 The user cannot create or drop a PRIMARY index.

Statement# 1, Info =46

\*\*\* Total elapsed time was 1 second.

drop index idx\_emp1\_deptno on emp1;

\*\*\* Index has been dropped.

\*\*\* Total elapsed time was 1 second.

NOTE: You can change Unique Primary Index to Non-Unique Primary Index and vise versa.

ALTER TABLE emp1 MODIFY NOT UNIQUE PRIMARY INDEX (emp\_no);

\*\*\* Table has been modified.

\*\*\* Total elapsed time was 1 second.

NOTE: You cannot change Unique Index to Non-Unique Index and vise versa.

ALTER TABLE emp1 MODIFY UNIQUE INDEX (dept\_no);

$

\*\*\* Failure 3707 Syntax error, expected something like a 'PRIMARY' keyword between the 'UNIQUE' keyword and the 'INDEX' keyword.

Statement# 1, Info =39

\*\*\* Total elapsed time was 1 second.

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( EMP\_NO );

**NOTE: Now it Non-Unique Primary Index**

ALTER TABLE emp1 MODIFY UNIQUE PRIMARY INDEX (emp\_no);

\*\*\* Table has been modified.

\*\*\* Total elapsed time was 1 second.

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO );

**Creating a Value-Ordered NUSI**

CREATE INDEX(DEPT\_NO) ORDER BY VALUES ON EMP1 ;

\*\*\* Index has been created.

\*\*\* Total elapsed time was 1 second.

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO )

INDEX ( DEPT\_NO ) ORDER BY VALUES ( DEPT\_NO );

drop INDEX ( DEPT\_NO ) ORDER BY VALUES ( DEPT\_NO ) on emp1;

\*\*\* Index has been dropped.

\*\*\* Total elapsed time was 1 second.

OR

drop INDEX ( DEPT\_NO ) on emp1;

\*\*\* Index has been dropped.

\*\*\* Total elapsed time was 1 second.

CREATE INDEX idx\_emp1\_deptno(DEPT\_NO) ON EMP1;

\*\*\* Index has been created.

\*\*\* Total elapsed time was 1 second.

show table emp1;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp1 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EMP\_NO INTEGER,

DEPT\_NO INTEGER,

FIRST\_NAME VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

LAST\_NAME CHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

SALARY DECIMAL(10,2),

ADDRESS VARCHAR(100) CHARACTER SET LATIN NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX ( EMP\_NO )

INDEX idx\_emp1\_deptno ( DEPT\_NO );

DROP INDEX idx\_emp1\_deptno ON EMP1;

\*\*\* Index has been dropped.

\*\*\* Total elapsed time was 1 second.

Creating duplicate table with data.

insert into emp1 values(1000,10,'Ram','Singh',50000,'Pune');

insert into emp1 values(2000,10,'Dev','Singh',50000,'Pune');

insert into emp1 values(3000,20,'Mark','Taylor',50000,'Pune');

insert into emp1 values(5000,30,'Gita','Singh',50000,'Pune');

select \* from emp1;

\*\*\* Query completed. 4 rows found. 6 columns returned.

\*\*\* Total elapsed time was 1 second.

EMP\_NO DEPT\_NO FIRST\_NAME LAST\_NAME SAL

----------- ----------- -------------------- -------------------- ---------

3000 20 Mark Taylor 50

5000 30 Gita Singh 50

2000 10 Dev Singh 50

1000 10 Ram Singh 50

create table emp1\_dup1 as emp1 with data;

create table emp1\_dup2 as emp1 with no data;

create table emp1\_dup3 as (select \* from emp1) with data;

create table emp1\_dup4 as (select emp\_no, first\_name, salary from emp1) with no data;

create table emp1\_dup5 as emp1 with data and stats;

create table emp1\_dup6 as emp1 with no data and stats;

create VOLATILE table emp1\_dup4

as

(select empno, ename || ' Kumar Singh' as full\_name, job, sal, deptno from emp1

union all

select emp\_no, first\_name , '' , salary, dept\_no from emp2 order by 5 )

with data

ON COMMIT PRESERVE ROWS;

select \* from emp1\_dup1;

EMP\_NO DEPT\_NO FIRST\_NAME LAST\_NAME SAL

----------- ----------- -------------------- -------------------- ---------

3000 20 Mark Taylor 50

5000 30 Gita Singh 50

2000 10 Dev Singh 50

1000 10 Ram Singh 50

select \* from emp1\_dup2;

\*\*\* Query completed. No rows found.

\*\*\* Total elapsed time was 1 second.

The following example creates a table called employee with FALLBACK option. The table contains 5 columns with EmployeeNo as the Unique Primary Index.

CREATE SET TABLE EMPLOYEE,

FALLBACK

(

EmployeeNo INTEGER,

FirstName VARCHAR(15),

LastName VARCHAR(15),

DOB DATE FORMAT 'YYYY-MM-DD',

JoinedDate DATE FORMAT 'YYYY-MM-DD',

DepartmentNo BYTEINT )

UNIQUE PRIMARY INDEX (EmployeeNo);

Once the table is created, you can use SHOW TABLE command to view the Definition of the table.

SHOW TABLE Employee;

SHOW TABLE Employee;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.Employee ,FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EmployeeNo INTEGER,

FirstName VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,

LastName VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC,

DOB DATE FORMAT 'YYYY-MM-DD',

JoinedDate DATE FORMAT 'YYYY-MM-DD',

DepartmentNo BYTEINT)

UNIQUE PRIMARY INDEX ( EmployeeNo );

ALTER TABLE command is used to add or drop columns from an existing table. You can also use ALTER TABLE command to modify the attributes of the existing columns.

## Syntax

Following is the generic syntax for ALTER TABLE.

ALTER TABLE <tablename>

ADD <columnname><column attributes>

DROP <columnname>;

## Example

The following example drops the column DOB and adds a new column BirthDate.

ALTER TABLE employee ADD BirthDate DATE FORMAT 'YYYY-MM-DD', DROP DOB;

ALTER TABLE employee ADD firstname VARCHAR(20);

\*\*\* Table has been modified.

\*\*\* Total elapsed time was 1 second.

NOTE: You can increase the column width but cannot decrease.

alter table employee add bonus decimal(10,2);

alter table employee rename bonus to bonus\_oct;

alter table employee drop bonus\_oct;

alter table employee add projname char(15);

\*\*\* Table has been modified.

\*\*\* Total elapsed time was 1 second.

alter table employee add projname char(20);

\*\*\* Failure 3558 Cannot alter the specified attribute(s) for PROJNAME.

Statement# 1, Info =0

\*\*\* Total elapsed time was 1 second.

**NOTE: You cannot change the size for CHAR data type both increase or decrease but for varchar, you increase but cannot decrease.**

Here are the results while applying above syntax for column's data type change:

VARCHAR ( from VARCHAR(20) TO VARCHAR(22)) – successful

VARCHAR ( from VARCHAR(22) TO VARCHAR(20)) - error

INTEGER ( FROM INTEGER TO DECIMAL(15,0)) - syntax error - 3558

CHAR ( FROM CHAR (5) TO CHAR (7)) - - syntax error - 3558

TIMESTAMP ( FROM TIMESTAMP(0) TO TIMESTAMP(1) )- syntax error - 3558

DECIMAL (FROM DECIMAL(15,0) TO DECIMAL(15,1) )- syntax error - 3558

**To Change the data type of a column.**

create table test (a integer, b decimal(8,0));

insert into test values(55,55.55);

insert into test values(66,66.66);

show table test;

alter table test add b\_new decimal(15,0);

update test set b\_new = b;

alter table test drop b, rename b\_new to b;

show table test;

drop table test;

DROP TABLE command is used to drop a table. When the DROP TABLE is issued, data in the table is deleted and the table is dropped.

## Syntax

Following is the generic syntax for DROP TABLE.

DROP TABLE <tablename>;

## Example

The following example drops the table ‘employee’.

DROP TABLE employee;

If you run the SHOW TABLE command after this, you will get an error message stating that the table doesn’t exist.

SHOW TABLE employee;

\*\*\* Failure 3807 Object 'employee' does not exist.

Statement# 1, Info = 0

\*\*\* Total elapsed time was 1 second.

## Insert Records

INSERT INTO statement is used to insert records into the table.

### Syntax

Following is the generic syntax for INSERT INTO.

INSERT INTO <tablename>

(column1, column2, column3,…)

VALUES

(value1, value2, value3 …);

### Example

The following example inserts records into the employee table.

INSERT INTO Employee VALUES (101,'Mike','James','1980-01-05',01,'2005-03-27');

insert into employee values(102,'Ram','Singh',current\_date,2,'1980-06-10');

insert into employee values(103,'Dev','Singh',current\_date-500,3,'1975-08-19');

insert into employee values(104,'Sita','Kumari',current\_date-1000,4,'1985-10-10');

insert into employee values(105,'Mohan','Kumar',current\_date-5000,5,'1975-10-10');

select \* from employee;

EmployeeNo FirstName LastName JoinedDate DepartmentNo BirthDate

----------- --------------- --------------- ---------- ------------ ----------

101 Mike James 2005-03-27 1 1980-01-05

104 Sita Kumari 2014-01-13 4 1985-10-10

102 Ram Singh 2016-10-09 2 1980-06-10

105 Mohan Kumar 2003-01-31 5 1975-10-10

103 Dev Singh 2015-05-28 3 1975-08-19

## Insert from Another Table

INSERT SELECT statement is used to insert records from another table.

### Syntax

Following is the generic syntax for INSERT INTO.

INSERT INTO <tablename>

(column1, column2, column3,…)

SELECT

column1, column2, column3…

FROM

<source table>;

### Example

The following example inserts records into the employee table. Create a table called Employee\_Bkup with the same column definition as employee table before running the following insert query.

CREATE TABLE Employee\_Bkup as Employee with no data;

INSERT INTO Employee\_Bkup(EmployeeNo,FirstName,LastName,BirthDate,JoinedDate,DepartmentNo)

SELECT EmployeeNo,FirstName,LastName,BirthDate,JoinedDate,DepartmentNo

FROM Employee;

When the above query is executed, it will insert all records from the employee table into employee\_bkup table.

### Rules

* The number of columns specified in the VALUES list should match with the columns specified in the INSERT INTO clause.
* Values are mandatory for NOT NULL columns.
* If no values are specified, then NULL is inserted for nullable fields.
* The data types of columns specified in the VALUES clause should be compatible with the data types of columns in the INSERT clause.

## Update Records

UPDATE statement is used to update records from the table.

### Syntax

Following is the generic syntax for UPDATE.

UPDATE <tablename>

SET <columnnamme> = <new value>

[WHERE condition];

### Example

The following example updates the employee dept to 03 for employee 101.

UPDATE Employee

SET DepartmentNo=03

WHERE EmployeeNo=101;

In the following output, you can see that the DepartmentNo is updated from 1 to 3 for EmployeeNo 101.

SELECT Employeeno,DepartmentNo FROM Employee;

\*\*\*Query completed.One row found.2 columns returned.

\*\*\*Total elapsed time was 1 second.

EmployeeNo DepartmentNo

---------------- --------

101 3

### Rules

* You can update one or more values of the table.
* If WHERE condition is not specified then all rows of the table are impacted.
* You can update a table with the values from another table.

## Delete Records

DELETE FROM statement is used to update records from the table.

### Syntax

Following is the generic syntax for DELETE FROM.

DELETE FROM <tablename>

[WHERE condition];

### Example

The following example deletes the employee 101 from the table employee.

DELETE FROM Employee

WHERE EmployeeNo=101;

In the following output, you can see that employee 101 is deleted from the table.

SELECT EmployeeNo FROM Employee;

\*\*\* Query completed. No rows found.

\*\*\* Total elapsed time was 1 second.

### Rules

* You can delete one or more records of the table.
* If WHERE condition is not specified then all rows of the table are deleted.
* You can delete a table with the values from another table.

SELECT statement is used to retrieve records from a table.

### Syntax

Following is the basic syntax of SELECT statement.

SELECT

column 1, column 2, .....

FROM

tablename;

### Example

Consider the following employee table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmployeeNo** | **FirstName** | **LastName** | **JoinedDate** | **DepartmentNo** | **BirthDate** |
| 101 | Mike | James | 3/27/2005 | 1 | 1/5/1980 |
| 102 | Robert | Williams | 4/25/2007 | 2 | 3/5/1983 |
| 103 | Peter | Paul | 3/21/2007 | 2 | 4/1/1983 |
| 104 | Alex | Stuart | 2/1/2008 | 2 | 11/6/1984 |
| 105 | Robert | James | 1/4/2008 | 3 | 12/1/1984 |

Following is an example of SELECT statement.

.set width 32767

.set retlimit 30

SELECT EmployeeNo,FirstName,LastName

FROM Employee;

When this query is executed, it fetches EmployeeNo, FirstName and LastName columns from the employee table.

EmployeeNoFirstNameLastName

----------- ------------------------------ ---------------------------

101 Mike James

104 Alex Stuart

102 Robert Williams

105 Robert James

103 Peter Paul

If you want to fetch all the columns from a table, you can use the following command instead of listing down all columns.

SELECT \* FROM Employee;

The above query will fetch all records from the employee table.

## WHERE Clause

WHERE clause is used to filter the records returned by the SELECT statement. A condition is associated with WHERE clause. Only, the records that satisfy the condition in the WHERE clause are returned.

### Syntax

Following is the syntax of the SELECT statement with WHERE clause.

SELECT \* FROM tablename

WHERE [condition];

### Example

The following query fetches records where EmployeeNo is 101.

SELECT \* FROM Employee

WHERE EmployeeNo=101;

When this query is executed, it returns the following records.

EmployeeNoFirstNameLastName

----------- ------------------------------ -----------------------------

101 Mike James

## ORDER BY

When the SELECT statement is executed, the returned rows are not in any specific order. ORDER BY clause is used to arrange the records in ascending/descending order on any columns.

### Syntax

Following is the syntax of the SELECT statement with ORDER BY clause.

SELECT \* FROM tablename

ORDER BY column 1, column 2..;

### Example

The following query fetches records from the employee table and orders the results by FirstName.

SELECT \* FROM Employee

ORDER BY FirstName;

When the above query is executed, it produces the following output.

EmployeeNoFirstNameLastName

----------- ------------------------------ -----------------------------

104 Alex Stuart

101 Mike James

103 Peter Paul

102 Robert Williams

105 Robert James

## GROUP BY

GROUP BY clause is used with SELECT statement and arranges similar records into groups.

### Syntax

Following is the syntax of the SELECT statement with GROUP BY clause.

SELECT column 1, column2 …. FROM tablename

GROUP BY column 1, column 2..;

### Example

The following example groups the records by DepartmentNo column and identifies the total count from each department.

SELECT DepartmentNo,Count(\*) FROM Employee

GROUP BY DepartmentNo;

When the above query is executed, it produces the following output.

DepartmentNoCount(\*)

------------ -----------

3 1

1 1

2 3

Teradata supports the following logical and conditional operators. These operators are used to perform comparison and combine multiple conditions.

|  |  |
| --- | --- |
| **Syntax** | **Meaning** |
| **>** | Greater than |
| **<** | Less than |
| **>=** | Greater than or equal to |
| **<=** | Less than or equal to |
| **=** | Equal to |
| **BETWEEN** | If values within range |
| **IN** | If values in <expression> |
| **NOT IN** | If values not in <expression> |
| **IS NULL** | If value is NULL |
| **IS NOT NULL** | If value is NOT NULL |
| **AND** | Combine multiple conditions. Evaluates to true only if all conditions are met |
| **OR** | Combine multiple conditions. Evaluates to true only if either of the conditions is met. |
| **NOT** | Reverses the meaning of the condition |

## BETWEEN

BETWEEN commands is used to check if a value is within a range of values.

### Example

Consider the following employee table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmployeeNo** | **FirstName** | **LastName** | **JoinedDate** | **DepartmentNo** | **BirthDate** |
| 101 | Mike | James | 3/27/2005 | 1 | 1/5/1980 |
| 102 | Robert | Williams | 4/25/2007 | 2 | 3/5/1983 |
| 103 | Peter | Paul | 3/21/2007 | 2 | 4/1/1983 |
| 104 | Alex | Stuart | 2/1/2008 | 2 | 11/6/1984 |
| 105 | Robert | James | 1/4/2008 | 3 | 12/1/1984 |

The following example fetches records with employee numbers in the range between 101,102 and 103.

SELECT EmployeeNo,FirstName FROM

Employee

WHERE EmployeeNo BETWEEN 101 AND 103;

When the above query is executed, it returns the employee records with employee no between 101 and 102.

\*\*\* Query completed. 3 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoFirstName

----------- ------------------------------

101 Mike

102 Robert

103 Peter

## IN

IN command is used to check the value against a given list of values.

### Example

The following example fetches records with employee numbers in 101, 102 and 103.

SELECT EmployeeNo,FirstName FROM

Employee

WHERE EmployeeNo in(101,102,103);

The above query returns the following records.

\*\*\* Query completed. 3 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoFirstName

----------- ------------------------------

101 Mike

102 Robert

103 Peter

## NOT IN

NOT IN command reverses the result of IN command. It fetches records with values that don’t match with the given list.

### Example

The following example fetches records with employee numbers not in 101, 102 and 103.

SELECT \* FROM

Employee

WHERE EmployeeNo not in(101,102,103);

The above query returns the following records.

\*\*\* Query completed. 2 rows found. 6 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoFirstNameLastName

----------- ------------------------------ -----------------------------

104 Alex Stuart

105 Robert James

SET operators combine results from multiple SELECT statement. This may look similar to Joins, but joins combines columns from multiple tables whereas SET operators combines rows from multiple rows.

### Rules

* The number of columns from each SELECT statement should be same.
* The data types from each SELECT must be compatible.
* ORDER BY should be included only in the final SELECT statement.

## UNION

UNION statement is used to combine results from multiple SELECT statements. It ignores duplicates.

### Syntax

Following is the basic syntax of the UNION statement.

SELECT col1, col2, col3…

FROM

<table 1>

[WHERE condition]

UNION

SELECT col1, col2, col3…

FROM

<table 2>

[WHERE condition];

### Example

Consider the following employee table and salary table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmployeeNo** | **FirstName** | **LastName** | **JoinedDate** | **DepartmentNo** | **BirthDate** |
| 101 | Mike | James | 3/27/2005 | 1 | 1/5/1980 |
| 102 | Robert | Williams | 4/25/2007 | 2 | 3/5/1983 |
| 103 | Peter | Paul | 3/21/2007 | 2 | 4/1/1983 |
| 104 | Alex | Stuart | 2/1/2008 | 2 | 11/6/1984 |
| 105 | Robert | James | 1/4/2008 | 3 | 12/1/1984 |

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeNo** | **Gross** | **Deduction** | **NetPay** |
| 101 | 40,000 | 4,000 | 36,000 |
| 102 | 80,000 | 6,000 | 74,000 |
| 103 | 90,000 | 7,000 | 83,000 |
| 104 | 75,000 | 5,000 | 70,000 |

The following UNION query combines the EmployeeNo value from both Employee and Salary table.

CREATE TABLE salary(

EmployeeNo integer,

Gross integer,

Deduction integer,

NetPay integer);

show table salary;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-----------------------------------------------

CREATE SET TABLE TDHARI.salary ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

EmployeeNo INTEGER,

Gross INTEGER,

Deduction INTEGER,

NetPay INTEGER)

PRIMARY INDEX ( EmployeeNo );

Insert into salary values (101, 40000, 4000, 36000);

Insert into salary values (102, 80000, 6000, 74000);

Insert into salary values (103, 90000, 7000, 83000);

Insert into salary values (104, 75000, 5000, 70000);

select \* from salary;

EmployeeNo Gross Deduction NetPay

----------- ----------- ----------- -----------

101 40000 4000 36000

104 75000 5000 70000

102 80000 6000 74000

103 90000 7000 83000

SELECT EmployeeNo

FROM Employee

UNION

SELECT EmployeeNo

FROM Salary;

When the query is executed, it produces the following output.

EmployeeNo

-----------

101

102

103

104

105

## UNION ALL

UNION ALL statement is similar to UNION, it combines results from multiple tables including duplicate rows.

### Syntax

Following is the basic syntax of the UNION ALL statement.

SELECT col1, col2, col3…

FROM

<table 1>

[WHERE condition]

UNION ALL

SELECT col1, col2, col3…

FROM

<table 2>

[WHERE condition];

### Example

Following is an example for UNION ALL statement.

SELECT EmployeeNo

FROM Employee

UNION ALL

SELECT EmployeeNo

FROM Salary ORDER BY 1 ;

When the above query is executed, it produces the following output. You can see that it returns the duplicates also.

EmployeeNo

-----------

101

104

102

105

103

101

104

102

103

## INTERSECT

INTERSECT command is also used to combine results from multiple SELECT statements. It returns the rows from the first SELECT statement that has corresponding match in the second SELECT statements. In other words, it returns the rows that exist in both SELECT statements.

### Syntax

Following is the basic syntax of the INTERSECT statement.

SELECT col1, col2, col3…

FROM

<table 1>

[WHERE condition]

INTERSECT

SELECT col1, col2, col3…

FROM

<table 2>

[WHERE condition];

### Example

Following is an example of INTERSECT statement. It returns the EmployeeNo values that exist in both tables.

SELECT EmployeeNo

FROM Employee

INTERSECT

SELECT EmployeeNo

FROM Salary ORDER BY 1 ;

When the above query is executed, it returns the following records. EmployeeNo 105 is excluded since it doesn’t exist in SALARY table.

EmployeeNo

-----------

101

104

102

103

## MINUS/EXCEPT

MINUS/EXCEPT commands combine rows from multiple tables and returns the rows which are in first SELECT but not in second SELECT. They both return the same results.

### Syntax

Following is the basic syntax of the MINUS statement.

SELECT col1, col2, col3…

FROM

<table1>

[WHERE condition]

MINUS

SELECT col1, col2, col3…

FROM

<table2>

[WHERE condition];

### Example

Following is an example of MINUS statement.

SELECT EmployeeNo

FROM Employee

MINUS

SELECT EmployeeNo

FROM Salary ORDER BY 1;

When this query is executed, it returns the following record.

EmployeeNo

-----------

105

SELECT EmployeeNo

FROM Employee

EXCEPT

SELECT EmployeeNo

FROM Salary ORDER BY 1;

EmployeeNo

-----------

105

Teradata provides several functions to manipulate the strings. These functions are compatible with ANSI standard.

|  |  |
| --- | --- |
| **Sr.No** | **String Function & Description** |
| 1 | **||**  Concatenates strings together |
| 2 | **SUBSTR**  Extracts a portion of a string (Teradata extension) |
| 3 | **SUBSTRING**  Extracts a portion of a string (ANSI standard) |
| 4 | **INDEX**  Locates the position of a character in a string (Teradata extension)  SELECT INDEX('India is my country', 'my'); |
| 5 | **POSITION**  Locates the position of a character in a string (ANSI standard)  SELECT POSITION('my' in 'India is my country'); |
| 6 | **TRIM**  Trims blanks from a string |
| 7 | **UPPER**  Converts a string to uppercase |
| 8 | **LOWER**  Converts a string to lowercase |

## Example

Following table lists some of the string functions with the results.

|  |  |
| --- | --- |
| **String Function** | **Result** |
| SELECT SUBSTRING('warehouse' FROM 1 FOR 4); | Ware |
| SELECT SUBSTR('warehouse',1,4); | Ware |
| SELECT 'data' || ' ' || 'warehouse'; | data warehouse |
| SELECT UPPER('data'); | DATA |
| SELECT LOWER('DATA'); | Data |

Teradata supports common aggregate functions. They can be used with the SELECT statement.

* **COUNT** − Counts the rows
* **SUM** − Sums up the values of the specified column(s)
* **MAX** − Returns the large value of the specified column
* **MIN** − Returns the minimum value of the specified column
* **AVG** − Returns the average value of the specified column

Example

Consider the following Salary Table.

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeNo** | **Gross** | **Deduction** | **NetPay** |
| 101 | 40,000 | 4,000 | 36,000 |
| 104 | 75,000 | 5,000 | 70,000 |
| 102 | 80,000 | 6,000 | 74,000 |
| 105 | 70,000 | 4,000 | 66,000 |
| 103 | 90,000 | 7,000 | 83,000 |

COUNT

The following example counts the number of records in the Salary table.

SELECT count(\*) from Salary;

Count(\*)

-----------

5

MAX

The following example returns maximum employee net salary value.

SELECT max(NetPay) from Salary;

Maximum(NetPay)

---------------------

83000

MIN

The following example returns minimum employee net salary value from the Salary table.

SELECT min(NetPay) from Salary;

Minimum(NetPay)

---------------------

36000

AVG

The following example returns the average of employees net salary value from the table.

SELECT avg(NetPay) from Salary;

Average(NetPay)

---------------------

65800

SUM

The following example calculates the sum of employees net salary from all records of the Salary table.

SELECT sum(NetPay) from Salary;

Sum(NetPay)

-----------------

329000

This chapter explains the CASE and COALESCE functions of Teradata.

## CASE Expression

CASE expression evaluates each row against a condition or WHEN clause and returns the result of the first match. If there are no matches then the result from ELSE part of returned.

### Syntax

Following is the syntax of the CASE expression.

CASE <expression>

WHEN <expression> THEN result-1

WHEN <expression> THEN result-2

ELSE

Result-n

END

### Example

Consider the following Employee table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmployeeNo** | **FirstName** | **LastName** | **JoinedDate** | **DepartmentNo** | **BirthDate** |
| 101 | Mike | James | 3/27/2005 | 1 | 1/5/1980 |
| 102 | Robert | Williams | 4/25/2007 | 2 | 3/5/1983 |
| 103 | Peter | Paul | 3/21/2007 | 2 | 4/1/1983 |
| 104 | Alex | Stuart | 2/1/2008 | 2 | 11/6/1984 |
| 105 | Robert | James | 1/4/2008 | 3 | 12/1/1984 |

The following example evaluates the DepartmentNo column and returns value of 1 if the department number is 1; returns 2 if the department number is 3; otherwise it returns value as invalid department.

SELECT EmployeeNo, CASE DepartmentNo

WHEN 1 THEN 'Admin'

WHEN 2 THEN 'IT'

WHEN 3 THEN 'FINANCE'

WHEN 4 THEN 'SALES'

ELSE 'Invalid Dept'

END AS Department

FROM Employee;

When the above query is executed, it produces the following output.

\*\*\* Query completed. 5 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNo Department

----------- ------------

101 Admin

104 SALES

102 IT

105 Invalid Dept

103 FINANCE

The above CASE expression can also be written in the following form which will produce the same result as above.

SELECT EmployeeNo, CASE

WHEN DepartmentNo = 1 THEN 'Admin'

WHEN DepartmentNo = 2 THEN 'IT'

WHEN DepartmentNo = 3 THEN 'FINANCE'

WHEN DepartmentNo = 4 THEN 'SALES'

ELSE 'Invalid Dept'

END AS Department

FROM Employee;

## COALESCE

COALESCE is a statement that returns the first non-null value of the expression. It returns NULL if all the arguments of the expression evaluates to NULL. Following is the syntax.

### Syntax

COALESCE(expression 1, expression 2, ....)

### Example

SELECT EmployeeNo, COALESCE(DepartmentNo, 'Department not found', 30+null, null\*null, null/null) FROM employee;

## NULLIF

NULLIF statement returns NULL if the arguments are equal.

### Syntax

Following is the syntax of the NULLIF statement.

NULLIF(expression 1, expression 2)

### Example

The following example returns NULL if the DepartmentNo is equal to 3. Otherwise, it returns the DepartmentNo value.

SELECT EmployeeNo, NULLIF(DepartmentNo,3) AS department

FROM Employee;

The above query returns the following records. You can see that employee 105 has department no. as NULL.

\*\*\* Query completed. 5 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNo department

----------- ------------------

101 1

104 2

102 2

105 ?

103 2

Join is used to combine records from more than one table. Tables are joined based on the common columns/values from these tables.

There are different types of Joins available.

* Inner Join
* Left Outer Join
* Right Outer Join
* Full Outer Join
* Self Join
* Cross Join
* Cartesian Production Join

## INNER JOIN

Inner Join combines records from multiple tables and returns the values that exist in both the tables.

### Syntax

Following is the syntax of the INNER JOIN statement.

SELECT col1, col2, col3….

FROM

Table-1

INNER JOIN

Table-2

ON (col1 = col2)

<WHERE condition>;

### Example

Consider the following employee table and salary table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EmployeeNo** | **FirstName** | **LastName** | **JoinedDate** | **DepartmentNo** | **BirthDate** |
| 101 | Mike | James | 3/27/2005 | 1 | 1/5/1980 |
| 102 | Robert | Williams | 4/25/2007 | 2 | 3/5/1983 |
| 103 | Peter | Paul | 3/21/2007 | 2 | 4/1/1983 |
| 104 | Alex | Stuart | 2/1/2008 | 2 | 11/6/1984 |
| 105 | Robert | James | 1/4/2008 | 3 | 12/1/1984 |

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeeNo** | **Gross** | **Deduction** | **NetPay** |
| 101 | 40,000 | 4,000 | 36,000 |
| 102 | 80,000 | 6,000 | 74,000 |
| 103 | 90,000 | 7,000 | 83,000 |
| 104 | 75,000 | 5,000 | 70,000 |

The following query joins the Employee table and Salary table on the common column EmployeeNo. Each table is assigned an alias A & B and the columns are referenced with the correct alias.

SELECT A.EmployeeNo,A.DepartmentNo,B.NetPay

FROM Employee A

INNER JOIN Salary B

ON (A.EmployeeNo= B.EmployeeNo);

When the above query is executed, it returns the following records. Employee 105 is not included in the result since it doesn’t have matching records in the Salary table.

\*\*\* Query completed. 4 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoDepartmentNoNetPay

----------- ------------ -----------

101 1 36000

102 2 74000

103 2 83000

104 2 70000

## OUTER JOIN

LEFT OUTER JOIN and RIGHT OUTER JOIN also combine the results from multiple tables.

* **LEFT OUTER JOIN** returns all the records from the left table and returns only the matching records from the right table.
* **RIGHT OUTER JOIN** returns all the records from the right table and returns only matching rows from the left table.
* **FULL OUTER JOIN** combines the results from both LEFT OUTER and RIGHT OUTER JOINS. It returns both matching and non-matching rows from the joined tables.

### Syntax

Following is the syntax of the OUTER JOIN statement. You need to use one of the options from LEFT OUTER JOIN, RIGHT OUTER JOIN or FULL OUTER JOIN.

SELECT col1, col2, col3….

FROM

Table-1

LEFT OUTER JOIN/RIGHT OUTER JOIN/FULL OUTER JOIN

Table-2

ON (col1 = col2)

<WHERE condition>;

### Example

Consider the following example of the LEFT OUTER JOIN query. It returns all the records from Employee table and matching records from Salary table.

SELECT A.EmployeeNo,A.DepartmentNo,B.NetPay

FROM Employee A

LEFT OUTER JOIN Salary B

ON (A.EmployeeNo= B.EmployeeNo)

ORDER BY A.EmployeeNo;

When the above query is executed, it produces the following output. For employee 105, NetPay value is NULL, since it doesn’t have matching records in Salary table.

\*\*\* Query completed. 5 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoDepartmentNoNetPay

----------- ------------ -----------

101 1 36000

102 2 74000

103 2 83000

104 2 70000

105 3 ?

SELECT A.EmployeeNo,A.DepartmentNo,B.NetPay

FROM Employee A

FULL OUTER JOIN Salary B

ON (A.EmployeeNo= B.EmployeeNo)

ORDER BY A.EmployeeNo;

\*\*\* Query completed. 5 rows found. 3 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNo DepartmentNo NetPay

----------- ------------ -----------

101 1 36000

102 2 74000

103 3 83000

104 4 70000

105 5 ?

## CROSS JOIN

Cross Join joins every row from the left table to every row from the right table.

### Syntax

Following is the syntax of the CROSS JOIN statement.

SELECT A.EmployeeNo, A.DepartmentNo, B.EmployeeNo,B.NetPay

FROM Employee A

CROSS JOIN Salary B

WHERE A.EmployeeNo = 101

ORDER BY B.EmployeeNo;

When the above query is executed, it produces the following output. Employee No 101 from Employee table is joined with each and every record from Salary Table.

\*\*\* Query completed. 4 rows found. 4 columns returned.

\*\*\* Total elapsed time was 1 second.

EmployeeNoDepartmentNoEmployeeNoNetPay

----------- ------------ ----------- -----------

101 1 101 36000

101 1 104 70000

101 1 102 74000

101 1 103 83000

Teradata provides built-in functions which are extensions to SQL. Following are the common built-in functions.

|  |  |
| --- | --- |
| **Function** | **Result** |
| SELECT DATE; | Date --------  16/01/01 |
| SELECT CURRENT\_DATE; | Date -------- 16/01/01 |
| SELECT TIME; | Time -------- 04:50:29 |
| SELECT CURRENT\_TIME; | Current Time(0)  ---------------  13:03:25+00:00 |
| SELECT CURRENT\_TIMESTAMP; | Current TimeStamp(6) -------------------------------- 2016-01-01 04:51:06.990000+00:00 |
| SELECT DATABASE; | Database ------------------------------ TDUSER |
| SELECT USER; | User  -------------------  TDUSER |

### Column Level attributes

**Teradata:**

|  |  |
| --- | --- |
| UPPERCASE | --> Stores the data entered in upper case. |
| CASESPECIFIC | --> Treats data as case specific for comparisons and sorting. By default in teradata 'A' and 'a' would mean the same and hence they will sort in any sequence. However when we make it case specific the sorting differs. |
| FORMAT | --> Used to control display format of a field. |
| TITLE | --> Used to provide default titles. |
| NAMED/ AS | --> Uses Default column name. |
| COMPRESS | --> Used to compress NULL's to take no physical space. |
| COMPRESS NULL | --> Used to compress NULL's to take no physical space. |
| COMPRESS value | --> Used to compress a particular value and NULL's to take no physical space |
| WITH DEFAULT | --> Uses System default values. |
| DEFAULT DATE | --> Uses today's date as default date. |
| DEFAULT TIME | --> Uses Current Time as default time. |

CREATE TABLE emp11

,NO FALLBACK

,NO BEFORE JOURNAL

,NO AFTER JOURNAL

(empno integer,

ename varchar(10) UPPERCASE,

salary decimal(10) format '99999999.99',

join\_date date default current\_date,

join\_time time default current\_time)

PRIMARY INDEX ( empno );

insert into emp11(empno, ename, salary) values(1000,'Ram',50000);

insert into emp11(empno, ename, salary) values(2000,'Shyam',80000);

select \* from emp11;

empno ename salary join\_date join\_time

----------- ---------- ----------- --------- ---------------

2000 SHYAM 00080000.00 16/10/09 23:31:20.000000

1000 RAM 00050000.00 16/10/09 23:31:20.000000

# GRANT REFERENCES, GRANT UPDATE, GRANT INDEX

V2R2 introduced some new privileges including REFERENCES, INDEX and expanded UPDATE privileges.

To create references to existing tables, a REFERENCES privilege is needed.

GRANT REFERENCES ON employee TO hari; /\* On all columns \*/

GRANT REFERENCES (employee\_number) ON employee TO hari; /\* On employee\_number only \*/

GRANT REFERENCES (ALL BUT employee\_number) ON employee TO hari; /\* On all except employee\_number \*/

GRANT UPDATE ON employee TO hari;

GRANT UPDATE (salary\_amount) ON employee TO hari; /\* On salary\_amount only \*/

GRANT UPDATE (ALL BUT salary\_amount) ON employee TO hari; /\* On All except salary\_amount \*/

Above grant commands can be used to allow or disallow updates on specific columns

GRANT INDEX ON employee TO hari; --> To CREATE or DROP indexes on a table

**Primary Key And Primary Index Difference:**

|  |  |
| --- | --- |
| **Primary Key** | **Primary Index** |
| One or More column used to uniquely identify table rows | One or more columns used to Distribute and locate rows on the AMPs |
| Used along with Foreign Key to Define table relationships. | Used to affect Distribution, access and Performance |
| Pk is always Unique **(ND-No duplicates)** | Primary index can be Unique or Non Unique (UPI and NUPI) |
| PK cannot be NULL **(NN- Not NULL)** | Primary index can contain NULL value. If its UPI then only one row can have null for PI and if its NUPI, multiple rows can have NULL. |
| PK value cannot be changed **( NC- No change)** | Primary Index value can be changed. This will lead to redistribution of data. |
| PK can be dropped and recreated using ALTER TABLE command | Primary index cannot be recreated without dropping and recreating the table. |

CREATE TABLE dept (

Deptno integer NOT NULL CONSTRAINT pk\_dept\_deptno PRIMARY KEY,

Dname varchar(15) NOT NULL UPPERCASE,

Loc varchar(15) UPPERCASE);

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

CREATE TABLE emp12 (

Empno integer NOT NULL CONSTRAINT pk\_emp12\_empno PRIMARY KEY,

Ename varchar(15) NOT NULL UPPERCASE,

Mobile numeric(10) NOT NULL CONSTRAINT uk\_emp12\_mobile UNIQUE,

Salary decimal(10,2) CONSTRAINT chk\_emp12\_salary CHECK(salary >= 25000),

Deptno integer CONSTRAINT fk\_emp12\_deptno REFERENCES dept(deptno));

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

show table dept;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.dept ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Deptno INTEGER NOT NULL,

Dname VARCHAR(15) CHARACTER SET LATIN UPPERCASE NOT CASESPECIFIC NOT

NULL,

Loc VARCHAR(15) CHARACTER SET LATIN UPPERCASE NOT CASESPECIFIC)

UNIQUE PRIMARY INDEX pk\_dept\_deptno ( Deptno );

show table emp12;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE SET TABLE TDHARI.emp12 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN UPPERCASE NOT CASESPECIFIC NOT

NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

CONSTRAINT chk\_emp12\_salary CHECK ( salary >= 25000 ),

CONSTRAINT fk\_emp12\_deptno FOREIGN KEY ( Deptno ) REFERENCES TDHARI.dept (

Deptno ))

UNIQUE PRIMARY INDEX pk\_emp12\_empno ( Empno )

UNIQUE INDEX uk\_emp12\_mobile ( Mobile );

help constraint dept.pk\_dept\_deptno;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Name Type Unique? Index Id Column Name

------------------------------ --------------- ------- -------- -----------

PK\_DEPT\_DEPTNO PRIMARY KEY Y 1 DEPTNO

help constraint emp12.pk\_emp12\_empno;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Name Type Unique? Index Id Column Name

------------------------------ --------------- ------- -------- -----------

PK\_EMP12\_EMPNO PRIMARY KEY Y 1 EMPNO

help constraint emp12.uk\_emp12\_mobile;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Name Type Unique? Index Id Column Name

------------------------------ --------------- ------- -------- -----------

UK\_EMP12\_MOBILE UNIQUE Y 4 MOBILE

help constraint emp12.chk\_emp12\_salary;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Name Type Constraint

------------------------------ --------------- ----------------------------

CHK\_EMP12\_SALARY CHECK CONSTRAINT chk\_emp12\_salary

help constraint emp12.fk\_emp12\_deptno;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Name Type State Index Id Foreig

------------------------------ --------------- ------------ -------- ------

FK\_EMP12\_DEPTNO REFERENCE VALID 0 DEPTNO

Help index dept;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Primary

or

Unique? Secondary? Column Names

------- ---------- -------------------------------------

Y P Deptno

Help index emp12;

\*\*\* Help information returned. 2 rows.

\*\*\* Total elapsed time was 1 second.

Primary

or

Unique? Secondary? Column Names

------- ---------- -------------------------------------

Y P Empno

Y S Mobile

Following are the points to note:

The PRIMARY KEY is implemented as unique primary index.

The UNIQUE constraint is implemented as a unique secondary index.

Up to 64 foreign keys may be defined for a child table.

Up to 64 references may reference a parent table.

The table creator must have the REFERENCES privilege on referenced table.

The referenced column must be defined as NOT NULL.

The referenced column must be defined unique using one of the following methods:

· USI

· UPI

· PRIMARY KEY constraint

· UNIQUE constraint

NOTE: The referenced column need not always be a PRIMARY KEY. Basic requirement is that the column should be NOT NULL and UNIQUE.)

**Adding constraints on existing table.**

CREATE TABLE emp13 (

Empno integer NOT NULL,

Ename varchar(15) NOT NULL,

Mobile numeric(10) NOT NULL,

Salary decimal(10,2),

Deptno integer)

NO PRIMARY INDEX;

HELP INDEX EMP13;

\*\*\* Empty HELP information returned.

\*\*\* Total elapsed time was 1 second.

**Adding/Modifying/Dropping CHECK constraints:**

1. *Adding unnamed constraint:*

ALTER TABLE emp13 ADD CHECK (salary >= 25000);

ALTER TABLE emp13 ADD CHECK (deptno IN (10,20,30,40,50));

ALTER TABLE emp13 ADD COMM DECIMAL(8,2);

ALTER TABLE emp13 ADD CHECK (COMM >= 1000);

SHOW TABLE emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2),

CHECK ( COMM >= 1000 ),

CHECK ( deptno IN (10 ,20 ,30 ,40 ,50 ) ),

CHECK ( salary >= 25000 ))

NO PRIMARY INDEX ;

1. *Dropping unnamed constraint:* ALTER TABLE emp13 DROP  salary CHECK; (Not working)
2. *Dropping all unnamed check constraints:*ALTER TABLE emp13 DROP  CHECK;

SHOW TABLE EMP13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.EMP13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX ;

Note that with the above syntax will drop all unnamed constraints but Named ones will not be dropped.

1. *Adding named constraint:*

ALTER TABLE emp13 ADD CONSTRAINT chk\_emp13\_salary CHECK(salary >= 25000);

1. *Dropping named constraint:*

ALTER TABLE emp13 DROP CONSTRAINT chk\_emp13\_salary ;

**Note:** Only named constraints can be modified. For unnamed constraints we need to drop the constraint and recreate it.

**This is the advantage of named constraints.**

Also if the data in the table does not conform to the new check constraint then an error is returned.

1. Modifying the constraint :

ALTER TABLE emp13 ADD CONSTRAINT chk\_emp13\_salary CHECK(salary >= 25000);

ALTER TABLE emp13  **modify constraint** chk\_emp13\_salary CHECK (salary >= 30000);

**Adding/Dropping UNIQUE constraint:**

Caution before adding UNIQUE constraint:

* Columns must be defined as **NOT NULL** before uniqueness constraint is applied to them.
* Data on the column must be **unique** , otherwise the constraint will be rejected

1. ***Adding unnamed UNIQUE constraints:***

ALTER TABLE emp13 ADD UNIQUE(mobile);

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX

**UNIQUE INDEX ( Mobile );**

1. *Dropping unnamed UNIQUE constraints:*

**Note that UNIQUE constraints are implemented as unique indexes- primary or secondary.**

So to drop a UNIQUE constraint we need to drop the index using DROP INDEX syntax:

This is the only way to drop unnamed UNIQUE indexes

DROP INDEX(mobile) ON emp13;

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX ;

1. ***Adding named UNIQUE constraints:***

ALTER TABLE emp13 ADD CONSTRAINT uk\_emp13\_mobile UNIQUE(mobile);

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX

**UNIQUE INDEX uk\_emp13\_mobile ( Mobile );**

1. *Dropping Named UNIQUE constraints :*

ALTER TABLE emp13 DROP CONSTRAINT uk\_emp13\_mobile;

As we know that UNIQUE is always implemented as an index, we can also use the drop index command to drop a constraint.

DROP INDEX (mobile) ON emp13;

**Adding / Dropping Primary Key Constraint:**

**Note that unlike Primary index , Primary key can be dropped and added.**

**There can be only 1 primary key per table.**

Primary keys are implemented as unique secondary indexes or Unique primary indexes.

Primary key can be dropped by using the DROP CONSTRAINT syntax for named primary keys and drop index syntax for unnamed. **However its very important to note that if a primary key is implemented as Unique Primary index then it cannot be dropped.**

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX ;

1. *Adding unnamed primary key:*

ALTER TABLE emp13 ADD PRIMARY KEY(empno);

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX

**UNIQUE INDEX ( Empno );**

1. *Dropping unnamed primary key :*

DROP INDEX (empno) ON emp13;

Note that this technique of dropping the key can work only if the primary key has not been implemented as UPI.

1. *Adding Named primary key constraint:*

ALTER TABLE emp13 ADD CONSTRAINT pk\_emp13\_empno PRIMARY KEY (empno);

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

-------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2))

NO PRIMARY INDEX

**UNIQUE INDEX pk\_emp13\_empno ( Empno );**

1. *Dropping Named primary key constraint :*

ALTER TABLE emp13 DROP CONSTRAINT pk\_emp13\_empno;

This primary key can also be dropped using the DROP INDEX syntax.

DROP UNIQUE INDEX pk\_emp13\_empno ON emp13;

DROP INDEX pk\_emp13\_empno ON emp13;

**Adding / Dropping Foreign key Constraint :**

Just like all other indexes Foreign key constraints can also be named or unnamed.

Rules for adding FK are:

1. The referenced column (column in the parent table ) must be defined as unique and not NULL
2. Referenced and Referencing columns must match in data type
3. There should be no values in the referencing table that are not present on the referenced table.

ALTER TABLE emp13 ADD CONSTRAINT fk\_emp13\_deptno FOREIGN KEY(deptno) REFERENCES dept(deptno);

show table emp13;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

---------------------------------------------------------------------------

CREATE MULTISET TABLE TDHARI.emp13 ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Empno INTEGER NOT NULL,

Ename VARCHAR(15) CHARACTER SET LATIN NOT CASESPECIFIC NOT NULL,

Mobile DECIMAL(10,0) NOT NULL,

Salary DECIMAL(10,2),

Deptno INTEGER,

COMM DECIMAL(8,2),

**CONSTRAINT fk\_emp13\_deptno FOREIGN KEY ( Deptno ) REFERENCES TDHARI.dept ( Deptno ))**

NO PRIMARY INDEX ;

ALTER TABLE emp13 DROP CONSTRAINT fk\_emp13\_deptno;

1. *Adding unnamed Foreign key :*

ALTER TABLE emp13 ADD FOREIGN KEY(deptno) REFERENCES dept(deptno);

1. *Dropping unnamed Foreign Key :*

ALTER TABLE emp13 DROP FOREIGN KEY (deptno) REFERENCES dept;

### Correlated Sub-query

First we create sample data for understanding the correlated queries using following queries:

DATABASE tdhari;

CREATE TABLE EMPLOYEE(

Employeeid integer,

DepartmentNo integer,

Salary decimal(8,2),

Hiredate date);

Insert into EMPLOYEE VALUES (1,100,1000.00,DATE '1999-01-01');

Insert into EMPLOYEE VALUES (2,200,2000.00,DATE '1998-01-01');

Insert into EMPLOYEE VALUES (3,300,3000.00,DATE '1997-01-01');

Insert into EMPLOYEE VALUES (4,400,4000.00,DATE '1996-01-01');

Insert into EMPLOYEE VALUES (5,100,5000.00,DATE '1998-02-01');

Insert into EMPLOYEE VALUES (6,200,6000.00,DATE '1997-03-01');

Insert into EMPLOYEE VALUES (7,300,7000.00,DATE '1996-06-01');

Insert into EMPLOYEE VALUES (8,400,8000.00,DATE '1992-01-01');

Insert into EMPLOYEE VALUES (9,100,9000.00,DATE '1999-04-01');

Insert into EMPLOYEE VALUES (10,200,10000.00,DATE '1998-01-01');

Insert into EMPLOYEE VALUES (11,300,11000.00,DATE '1999-05-01');

Insert into EMPLOYEE VALUES (12,400,12000.00,DATE '1999-08-01');

Insert into EMPLOYEE VALUES (13,500,16000.00, '1999-08-01');

Correlated subquery is a query where the outer query is processed a row at a time, against the subquery result. The subquery is executed for each row processed by the outer query.

**Example 1 :** Find the employee with maximum salary within his department. Note that here we are not finding the total maximum, instead we are finding departmental maximums.

select DepartmentNo, max(salary) from employee group by DepartmentNo order by 1;

DepartmentNo Maximum(Salary)

------------ ---------------

100 9000.00

200 10000.00

300 11000.00

400 12000.00

500 16000.00

Select \* from employee E where salary =(select max(salary) from employee EX where E.DepartmentNo = Ex.departmentNo) order by DepartmentNo;

Employeeid DepartmentNo Salary Hiredate

----------- ------------ ---------- --------

9 100 9000.00 99/04/01

10 200 10000.00 98/01/01

11 300 11000.00 99/05/01

12 400 12000.00 99/08/01

13 500 16000.00 99/08/01

Following is how the query works:

1. Get 1 row from the employee table
2. Using the departmentNo run the inner query to find the maximum salary for the department.
3. Compare the salary in the outer row with the maximum salary from the subquery.
4. If the values are equal the row is output.
5. If the values are not equal then the row is not output.
6. Repeat the same for next row

Thus we can see that we have got employees with maximum salaries in their respective departments.

**Example 2:** Find employees whose salary is greater than their departmental averages.

select DepartmentNo, avg(salary) from employee group by DepartmentNo order by 1;

DepartmentNo Average(Salary)

------------ ---------------

100 5000.00

200 6000.00

300 7000.00

400 8000.00

500 16000.00

Select \* from employee e where salary > (select avg(salary) from employee ex where e.departmentno=ex.departmentno) order by DepartmentNo;

Employeeid DepartmentNo Salary Hiredate

----------- ------------ ---------- --------

9 100 9000.00 99/04/01

10 200 10000.00 98/01/01

11 300 11000.00 99/05/01

12 400 12000.00 99/08/01

**Example 3:** Find out the employee who joined latest in each department

Select \* from employee e where hiredate = (select max(hiredate) from employee ex where e.departmentno=ex.departmentno);

Employeeid DepartmentNo Salary Hiredate

----------- ------------ ---------- --------

2 200 2000.00 98/01/01

9 100 9000.00 99/04/01

10 200 10000.00 98/01/01

11 300 11000.00 99/05/01

12 400 12000.00 99/08/01

13 500 16000.00 99/08/01

Note that there are two rows output for departmentno = 200. The reason being that the maximum hire date for this department is 1/1/1998 which is the hire date for two of its employees and hence both of them qualify in the output.

**NOT IN and EXISTS:**

Given a query where we need to find employees without a valid department we would normally use a traditional subquery as follows:

SELECT \* FROM EMPLOYEE WHERE DEPARTMENTNO NOT IN (select DISTINCT DEPARTMENTNO FROM DEPARTMENT);

We can get similar result using the EXISTS with correlated Subqueries as follows:

SELECT \* FROM EMPLOYEE E WHERE NOT EXISTS (SELECT \* FROM DEPARTMENT D WHERE E.DEPARTMENTNO=D.DEPARTMENTNO);

### CASE expression

SELECT

EMPLOYEEID,

CASE DEPARTMENTNO

WHEN 100 THEN 'SUPPORT'

WHEN 200 THEN 'SALES'

WHEN 300 THEN 'FINANCE'

WHEN 400 THEN 'DEVELOPMENT'

ELSE 'UNKNOWN'

END AS DEPERTMENT\_NAME

FROM EMPLOYEE

ORDER BY 1;

\*\*\* Query completed. 13 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

Employeeid DEPERTMENT\_NAME

----------- ---------------

1 SUPPORT

2 SALES

3 FINANCE

4 DEVELOPMENT

5 SUPPORT

6 SALES

7 FINANCE

8 DEVELOPMENT

9 SUPPORT

10 SALES

11 FINANCE

12 DEVELOPMENT

13 UNKNOWN

SELECT

EMPLOYEEID,

CASE

WHEN DEPARTMENTNO=100 THEN 'SUPPORT'

WHEN DEPARTMENTNO=200 AND Employeeid=2 THEN 'SALES\_SPECIAL' /\*multiple columns used in Case\*/

WHEN DEPARTMENTNO=200 AND Employeeid <> 2 THEN 'SALES'

WHEN DEPARTMENTNO=300 THEN 'FINANCE'

WHEN DEPARTMENTNO=400 THEN 'DEVELOPMENT'

ELSE 'UNKNOWN'

END AS DEPERTMENT\_NAME

FROM EMPLOYEE

ORDER BY 1;

\*\*\* Query completed. 13 rows found. 2 columns returned.

\*\*\* Total elapsed time was 1 second.

Employeeid DEPERTMENT\_NAME

----------- ---------------

1 SUPPORT

2 SALES\_SPECIAL

3 FINANCE

4 DEVELOPMENT

5 SUPPORT

6 SALES

7 FINANCE

8 DEVELOPMENT

9 SUPPORT

10 SALES

11 FINANCE

12 DEVELOPMENT

13 UNKNOWN

### Index Covering

**What is Index Covering:**

If the query references only those columns that are contained within a given index, then the query is said to be covering.

Such cases are more efficient because the optimizer has to access only the index subtable and does not need to access the base table rows at all.

Covering index will be considered for any query that references only columns defined in a given NUSI. These columns can be specified anywhere in the query including the:

* SELECT list
* WHERE clause
* aggregate functions
* GROUP BY
* expressions

However having all the indexed columns does not guarantee that the optimizer will use index to cover the query. Whether or not the index is used we can check after looking at the explain plan

**Example 1:**

show table employee;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

--------------------------------------------------

CREATE SET TABLE TDHARI.employee ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Employeeid INTEGER,

DepartmentNo INTEGER,

Salary DECIMAL(8,2),

Hiredate DATE FORMAT 'YYYY-MM-DD')

PRIMARY INDEX ( Employeeid );

create index (departmentno) order by values on employee;

show table employee;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------

CREATE SET TABLE TDHARI.employee ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

Employeeid INTEGER,

DepartmentNo INTEGER,

Salary DECIMAL(8,2),

Hiredate DATE FORMAT 'YYYY-MM-DD')

PRIMARY INDEX ( Employeeid )

INDEX ( DepartmentNo ) ORDER BY VALUES ( DepartmentNo );

EXPLAIN select departmentno from employee where departmentno <> 400;

Explain select departmentno from employee where departmentno <> 400

Explanation

---------------------------------------------------------------------------

1) First, we lock a distinct TDHARI."pseudo table" for read on a

RowHash to prevent global deadlock for TDHARI.employee.

2) Next, we lock TDHARI.employee for read.

3) We do an all-AMPs RETRIEVE step from TDHARI.employee by way of a

traversal of index # 4 without accessing the base table with a

residual condition of ("(TDHARI.employee.DepartmentNo >= 401) OR

(TDHARI.employee.DepartmentNo <= 399)") into Spool 1 (group\_amps),

which is built locally on the AMPs. The size of Spool 1 is

estimated with no confidence to be 2 rows (64 bytes). The

estimated time for this step is 0.03 seconds.

4) Finally, we send out an END TRANSACTION step to all AMPs involved

in processing the request.

-> The contents of Spool 1 are sent back to the user as the result of

statement 1. The total estimated time is 0.03 seconds.

Note the line where explain plan explicitly says that it would not access the base table.

This index is covering the query.

### JOIN indexes

**JOIN Index:**

* JOIN index is a technique that is used to improve the performance of certain types of queries.
* With NUSI the columns involved in the index can belong only to one table. With Join index the index structure contains columns from one or more tables.
* Note that once this is created , it is available only to optimizer. It’s the optimizer who decides whether to use join index or not. This index can never be directly accessed by the user.
* JOIN index helps in joining tables by providing the data needed by using index itself and also by avoiding redistribution of data in many cases.
* Following are the types of JOIN indexes:

1. Multiple table Join index: This type of index is used to pre-join the tables, which can help prevent redistribution of data
2. Single table Join index: This type of Join index is used to rehash and redistribute the rows of a single table based on specified columns
3. Aggregate Join index: Aggregate join index is used to create summary table.

* Example of Creating a **multi table join index**

create table department (

DEPARTMENTNO integer,

departmentname varchar(20),

departmentloc varchar(20));

\*\*\* Table has been created.

\*\*\* Total elapsed time was 1 second.

show table department;

\*\*\* Text of DDL statement returned.

\*\*\* Total elapsed time was 1 second.

------------------------------------------------------------------------

CREATE SET TABLE TDHARI.department ,NO FALLBACK ,

NO BEFORE JOURNAL,

NO AFTER JOURNAL,

CHECKSUM = DEFAULT

(

DEPARTMENTNO INTEGER,

departmentname VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC,

departmentloc VARCHAR(20) CHARACTER SET LATIN NOT CASESPECIFIC)

PRIMARY INDEX ( DEPARTMENTNO );

**CREATE JOIN INDEX JNINDEX**

**AS** SELECT departmentname, d.DEPARTMENTNO, employeeid, salary, hiredate

FROM department d

Join

employee e

on d.departmentno = e.departmentno

**primary index (departmentno);**

HELP INDEX JNINDEX;

\*\*\* Help information returned. One row.

\*\*\* Total elapsed time was 1 second.

Primary

or

Unique? Secondary? Column Names

------- ---------- -------------------------

N P DEPARTMENTNO

Above query can also be written as

**CREATE JOIN INDEX JNINDEX1**

**AS** SELECT **(**departmentname, d.DEPARTMENTNO**)**, **(**employeeid, salary, Hiredate**)**

FROM department d

Join

employee e

on d.departmentno = e.departmentno

**primary index (departmentno);**

Functionality wise both the syntax mean the same , however in the 2nd syntax join index is comprised of a 'fixed' portion (first parenthesis) and a 'repeatable' portion (second parenthesis). This represents a denormalization of the data. Difference is internally.

Explain of the below query shows that the join index was used to cover the query , and hence result will be generated without accessing the base rows.

Explain SELECT departmentname,employeeid,salary

from department d Join employee e

on d.departmentno=e.departmentno

  1) First, we lock a distinct EDW\_RESTORE\_TABLES."pseudo table" for

     read on a RowHash to prevent global deadlock for

     EDW\_RESTORE\_TABLES.JNINDEX.

  2) Next, we lock EDW\_RESTORE\_TABLES.JNINDEX for read.

  3) **We do an all-AMPs RETRIEVE step from EDW\_RESTORE\_TABLES.JNINDEX by**

     way of an all-rows scan with no residual conditions into Spool 1

     (group\_amps), which is built locally on the AMPs.  The size of

     Spool 1 is estimated with low confidence to be 1,440 rows (

     139,680 bytes).  The estimated time for this step is 0.01 seconds.

  4) Finally, we send out an END TRANSACTION step to all AMPs involved

     in processing the request.

  -> The contents of Spool 1 are sent back to the user as the result of

     statement 1.  The total estimated time is 0.01 seconds.

* JOIN index once created we don’t need to maintain the index , RDBMS does that automatically which means that when the base rows change the join index is also changed automatically.
* Note that when creating the JOIN index we have specified  a primary index. The primary index gets assigned irrespective of whether we explicitly specify one or not. Primary index is used to redistribute the index rows across the AMP's.

**note :** JOIN index can only have a NUPI and never UPI

* The index rows on the  AMP's are sequenced by the hash value of the primary index of the join index. However this type of sequencing is not beneficial in range processing. Hence we have **a option to use a ORDER BY clause** to override the default sequencing.

Syntax:

**CREATE JOIN INDEX JNINDEX**

**AS** SELECT

departmentname,

d.DEPARTMENTNO,

employeeid,

salary,

hiredate

FROM

department d

Join

employee e

on d.departmentno=e.departmentno

**ORDER BY HIREDATE**

**primary index (departmentno);**

The rules for the column in ORDER BY are  similar to those of the value ordered NUSI's

1. Single Column
2. The Column used for ordering should be  part of or all of the fixed portion index definition.
3. The column must be numeric column.
4. Column should not be greater then 4 bytes in length - INT, SMALLINT, BYTEINT, DATE, DEC with storage length no greater than 4 bytes are valid.

* JOIN indexes with NUSI defined.  
    
  NUSI can be defined on a Join index to improve performance. We earlier saw that index  rows can be ordered by a particular column to facilitate range processing . But what if there are multiple columns that are used for range processing.

This can be solved by making use of join index and value ordering it.

NUSI's on Join Indexes can be built

1. As part of the CREATE JOIN INDEX statement,

or

1. they can be added after Join Index creation using the CREATE INDEX statement.

Example:

**CREATE JOIN INDEX JNINDEX**

**AS SELECT**

departmentname,

d.DEPARTMENTNO,

employeeid,

salary,

hiredate

FROM

department d

Join

employee e

on d.departmentno=e.departmentno

**primary index (departmentno)**

**INDEX(hiredate) ORDER BY VALUES(hiredate);**

Or

CREATE INDEX(hiredate) ORDER BY VALUES (hiredate) on JNINDEX;

* Example of **Single table indexes**

Basic purpose single table Join indexes is to rehash and redistribute the rows of the table by column other than the primary index.

Assume a scenario where we join a two table and one of the two table needs to get distributed on the join column so that join can be performed. This would be time consuming if the table is very huge. However we can create a single table join index on this table with the column used for redistribution as the primary index of the join index. Thus rows will be pre-distributed and hence there wont be any re distribution while performing the join and thus will speed up the join.

CREATE JOIN INDEX emp\_deptno

AS SELECT employee\_number, department\_number

FROM employee

PRIMARY INDEX (department\_number)

;

drop join index jnindex1;