DB2 - TABLES

Tables are logical structure maintained by Database manager. In a table each vertical block called as column *Tuple* and each horizontal block called as row *Entity*. The collection of data stored in the form of columns and rows is known as a table. In tables, each column has different data type. Tables are used to store persistent data.

Type of tables

- **Base Tables**: They hold persistent data. There are different kinds of base tables, including:
  - **Regular Tables**: General purpose tables, Common tables with indexes are general purpose tables.
  - **Multidimensional Clustering Table** *MDC*: This type of table physically clustered on more than one key, and it used to maintain large database environments. These type of tables are not supported in DB2 pureScale.
  - **Insert time clustering Table** *ITC*: Similar to MDC tables, rows are clustered by the time they are inserted into the tables. They can be partitioned tables. They too, do not support pureScale environment.
  - **Range-Clustered tables Table** *RCT*: These type of tables provide fast and direct access of data. These are implemented as sequential clusters. Each record in the table has a record ID. These type of tables are used where the data is clustered tightly with one or more columns in the table. This type of tables also do not support in DB2 pureScale.
  - **Partitioned Tables**: These type of tables are used in data organization schema, in which table data is divided into multiple storage objects. Data partitions can be added to, attached to and detached from a partitioned table. You can store multiple data partition from a table in one tablespace.
  - **Temporal Tables**: History of a table in a database is stored in temporal tables such as details of the modifications done previously.
- **Temporary Tables**: For temporary work of different database operations, you need to use temporary tables. The temporary tables *DGTTs* do not appear in system catalog, XML columns cannot be used in created temporary tables.
- **Materialized Query Tables**: MQT can be used to improve the performance of queries. These types of tables are defined by a query, which is used to determine the data in the tables.

Creating Tables

The following syntax creates table:

Syntax: [To create a new table]

```
db2 create table <schema_name>.<table_name>
(column_name column_type....) in <tablespace_name>
```

Example: We create a table to store “employee” details in the schema of “professional”. This table has “id, name, jobrole, joindate, salary” fields and this table data would be stored in tablespace “ts1”.

```
db2 create table professional.employee(id int, name
varchar(50),jobrole varchar(30),joindate date,
salary double) in ts1
```

Output:

```
DB20000I The SQL command completed successfully.
```

Listing table details

The following syntax is used to list table details:

Syntax: [To see the list of tables created with schemas]

```
db2 select tabname, tabschema, tbspace from syscat.tables
```

Example: [To see the list of tables in the current database]

```
db2 select tabname, tabschema, tbspace from syscat.tables
```

Output:

```
TABNAME      TABSCHEMA     TBSPACE
------------ ------------- --------
EMPLOYEE     PROFESSIONAL    TS1


 1 record(s) selected.
```

Listing columns in a table

The following syntax lists columns in a table:

Syntax: [To see columns and data types of a table]

```
db2 describe table <table_name>
```

Example: [To see the columns and data types of table ‘employee’]

```
db2 describe table professional.employee
```

Output:

```
                Data type                     Column
Column name     schema     Data type name     Length     Scale Nulls
------ ----- --------- ----------------- --------- ----- ------
ID              SYSIBM     INTEGER              4         0     Yes
NAME            SYSIBM     VARCHAR             50         0     Yes
JOBROLE         SYSIBM     VARCHAR             30         0     Yes
JOINDATE        SYSIBM     DATE                4         0     Yes
SALARY          SYSIBM     DOUBLE              8         0     Yes

  5 record(s) selected.
```

Hidden Columns

You can hide an entire column of a table. If you call “select * from” query, the hidden columns are not returned in the resulting table. When you insert data into a table, an “INSERT” statement without a column list does not expect values for any implicitly hidden columns. These type of columns are highly referenced in materialized query tables. These type of columns do not support to create temporary tables.

Creating table with hidden column

The following syntax creates table with hidden columns:

Syntax: [To create a table with hidden columns]

```
db2 create table <tab_name> (col1 datatype,col2 datatype
implicitly hidden)
```

Example: [To create a 'customer' table with hidden columns 'phone']

```
db2 create table professional.customer(custid integer not
null, fullname varchar(100), phone char(10)
implicitly hidden)
```

Inserting data values in table

The following syntax inserts values in the table:

Syntax: [To insert values into a table]

```
db2 insert into <tab_name>(col1,col2,...)
 values(val1,val2,..)
```

Example: [To insert values in 'customer' table]

```
db2 insert into professional.customer(custid, fullname, phone)
values(100,'ravi','9898989')


db2 insert into professional.customer(custid, fullname, phone)
values(101,'krathi','87996659')


db2 insert into professional.customer(custid, fullname, phone)
values(102,'gopal','768678687')
```

Output:

```
DB20000I  The SQL command completed successfully.
```

Retrieving values from table

The following syntax retrieves values from the table:

Syntax: [To retrieve values form a table]

```
db2 select * from &lttab_name>
```

Example: [To retrieve values from 'customer' table]

```
db2 select * from professional.customer
```

Output:

```
CUSTID      FULLNAME
----------- ------------------------
        100 ravi

        101 krathi

        102 gopal

  3 record(s) selected.
```

Retrieving values from a table including hidden columns

The following syntax retrieves values from selected columns:

Syntax: [To retrieve selected hidden columns values from a table]

```
db2 select col1,col2,col3 from <tab_name>
```

Example: [To retrieve selected columns values result from a table]

```
db2 select custid,fullname,phone from professional.customer
```

Output:

```
CUSTID  FULLNAME    PHONE
------- ---------   ------------
100     ravi        9898989

101     krathi      87996659

102     gopal       768678687

  3 record(s) selected.
```

If you want to see the data in the hidden columns, you need to execute “DESCRIBE” command.

Syntax:

```
db2 describe table <table_name> show detail
```

Example:

```
db2 describe table professional.customer show detail
```

Output:

```
Column name     Data type schema     Data type name  Column
           column    Partitionkey  code
                                        Length   Scale    Nulls
number     sequence       page     Hidden      Default
--------------- -------------------- --------------- -------- ----
---- -------- ---------- ------------- -------- ----------- ------
---
CUSTID          SYSIBM               INTEGER         4        0
No       0          0          0        No
FULLNAME        SYSIBM               VARCHAR         100      0
Yes      1          0        1208     No

PHONE           SYSIBM               CHARACTER       10       0
Yes      2          0             1208     Implicitly

3 record(s) selected.
```

Altering the type of table columns

You can modify our table structure using this “alter” command as follows:

Syntax:

```
db2 alter table <tab_name> alter column <col_name> set data type <data_type>
```

Example: [To modify the data type for column “id” from “int” to “bigint” for employee table]

```
db2 alter table professional.employee alter column id set data type bigint
```

Output::

```
DB20000I The SQL command completed successfully.
```

Altering column name

You can change column name as shown below:

Syntax: [To modify the column name from old name to new name of a table]

```
db2 alter table <tab_name> rename column <old_name> to <new_name>
```

Example: [To modify the column name from “fullname” to “custname” in “customers” table.]

```
db2 alter table professional.customer rename column fullname to custname
```

Dropping the tables

To delete any table, you need to use the “DROP” command as follows:

Syntax:

```
db2 drop table <tab_name>
```

Example: [To drop customer table form database]

```
db2 drop table professional.customers
```

To delete the entire hierarchy of the table *includingtriggersandrelation*, you need to use “DROP TABLE HIERARCHY” command.

Syntax:

```
db2 drop table hierarchy <tab_name>
```

Example: [To drop entire hierarchy of a table ‘customer’]

```
db2 drop table hierarchy professional.customers
```