21-9-22 INTRODUCTION & ARCHITECTURE OF TERADATA

What is Teradata?

Teradata is one of the RDBMS. Used to store huge amount of data in data warehousing application. It uses concept called parallelism. It supports automatic data distribution with help of AMP. Teradata has three systems they are,

* SMP (Symmetric Multi-processing)

Single server system

* MPP (Massive parallel processing)

This system is used now a days, where multi servers system

* Cloud environment

Architecture of Teradata: Is a Massive Parallel Processing (MPP).

PARSING ENGINE

MESSAGE PASSING LAYER (BYNET)

AMP4

AMP3

AMP2

AMP1

VDISK1 VDISK2 VDISK3 VDISK4

PARSING ENGINE:

* When user run a query first it gets connected to parsing engine (PE), which will generate a plan for data distribution and send it to AMP (Access module processor).

Sub-component of parsing engine:

* Parser: Checks syntax, symmetrical, and existence of objects.
* Session control: Checks security like log in, permission to execute query.
* Optimizer: Find out best plan to execute.
* Dispatcher: Forward it to AMP.

BYNET: BYNET is communicating channel between PE and AMP for communicating the plan to execute and decide which AMP receive the plan. Act as Message Passing Layer.

* BYNET (BAN YAN Network)
* There are two types of BYNET, BYNET 0(board) and BYNET 1(board less), if one is not working BYNET 0 will replace it. It is called fault tolerant
* BYNET allows point-to-point, (msg is routed to ONE SPECIFIC AMPs and PE on the system).
* multi-cast, (msg is routed to group of AMPs).
* broadcast (msg is routed to ALL AMPS or PEs on the sys) communications among the server(node).
* Load balancing, balance the load in the node equally
* higher scalability, increasing and decreasing the scalability of node. (Number of nodes needed and can remove the nodes)
* fault tolerant are the features of BYNET.

AMP (Access Module Processor):

* AMPs will perform all sort of operations like, joins, sorting, aggregate, DDL, DML etc.
* Each AMPs are stored in data directory
* When PE send the plan via BYNET, AMP receives the plan and finds rows and table in the disk.
* AMP stores and retrieve row to and from disk.
* AMP can lock a table and have all access on data distribution.

VDISK (Virtual-disk):

Teradata offer set vdisk to each AMPs. Each VDISK has one cylinder and inside it data blocks which stores data in. In that, data are classified into three types,

* Hot data
* Warm data
* Cold data

Retrieval architecture:

Step 1: The user raises query which is sent to PE.

Step 2: PE checks syntax and security and finds a plan to execute the query.

Step 3: BYNET will sent the plan to AMP with help of communication types.

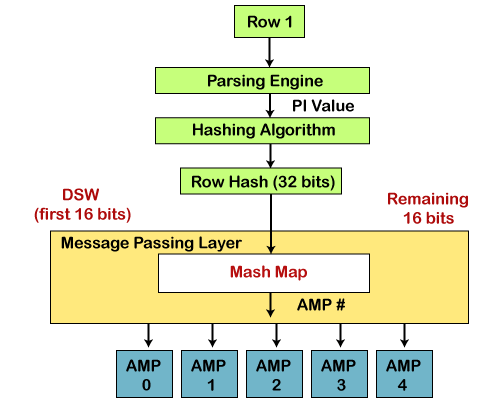
Step 4: The table row is distributed on the AMP, and data retrieval will happen from the disk.

Step 5: AMP send back the data to PE via BYNET.

Step 6: PE returns the data to the user.

22-9-22 INTRODUCTION & ARCHITECTURE OF TERADATA 2

Data Distribution:



* First, the client submits a query.
* Then, the parser receives the query and passes the PI value of the record to the hashing algorithm.
* The hashing algorithm hashes the primary index value and returns a 32-bit number, called Row Hash.
* The higher-order bits of the row hash (first 16 bits) are used to identify the hash map entry. The hash map contains one AMP #. Hash map is an array of buckets that contain specific AMP #.
* BYNET sends the data to the identified AMP.
* AMP uses the 32-bit Row hash to locate the row within its disk.
* If there is any record with the same row hash, it increments the uniqueness ID (+1), a 32-bit number.
* The combination of Row hash and Uniqueness ID is called Row ID.
* Row ID prefixes each record in the disk.
* Each table row in the AMP is logically sorted by their Row IDs.

23-9-22 DATA TYPES AND BUILT-IN FUNCTIONS

DATA TYPES:

* BYTEINT

Serial number, ID, age etc. Range between -128 to +128

CREATE TABLE Education

  (Id CHAR(9)

  ,LastName CHAR(26)

  ,EdLev BYTEINT FORMAT 'Z9'

   CHECK (EdLev BETWEEN 0 AND 22) NOT NULL);

* SMALLINT (Range between -32768 to +32767)

 CREATE TABLE Departments

     (DeptNo SMALLINT FORMAT '999' BETWEEN 100 AND 900

     ,ManagerName CHAR(26)

     ,ManagerID CHAR(9));

* INTEGER

Phone number, account number, all number between -2,147,483,648 to +2147,483,647

CREATE TABLE Contact

     (Id CHAR(9)

     ,LastName CHAR(26)

     ,TelNo INTEGER);

* BIGINT (Range between -9,233,372,036,854,775,80 8 to +9,233,372,036,854,775,8 07)

CREATE TABLE RelevantNumbers

     (Id CHAR(9)

     ,LastSummary INTEGER

     ,Total BIGINT);

* DECIMAL

CREATE TABLE Salaries

     (Id CHAR(9)

     ,Salary DECIMAL(8,2) FORMAT 'ZZZ,ZZ9.99'

      CHECK (Salary BETWEEN 1.00 AND 999000.00) );

* NUMERIC
* FLOAT

CREATE TABLE Salaries

     (Id CHAR(9)

     ,SalaryFactor FLOAT BETWEEN .1 AND 1E1 );

* CHAR

CREATE TABLE PersonalData

     (Id INTEGER

     ,Age INTEGER

     ,Sex CHARACTER NOT NULL UPPERCASE

     ,Frgn\_Lang CHARACTER(7) NULL UPPERCASE );

* VARCHAR

 CREATE TABLE InfoTable

     (InfoKey VARCHAR(10) NOT NULL

     ,InfoData VARCHAR(16384) )

   UNIQUE PRIMARY INDEX ( InfoKey );

* DATE (YYYYYMMDD)
* TIME (HHMMSS.nnnnnn **or** HHMMSS.nnnnnn+HHMM)
* TIMESTAMP (YYMMDDHHMMSS.nnnnnn **or** YYMMDDHHMMSS.nnnnnn +HHMM)

There are more data types in Teradata. These are commonly used data type in Teradata.

Built-in functions:

* SELECT DATE;
* SELECT CURRENT\_DATE
* SELECT TIME
* SELECT CURRENT\_TIME
* SELECT CURRENT\_TIMESTAMP (comes with today’s date, time)
* SELECT DATABASE (Teradata user)

26-9-22 PRIMARY INDEX AND SECONDARY INDEX

What is Indexes: Indexes is a term used in Teradata system which allows query to retrieve data from AMP effectively.

Unique Primary Index

Non-Unique Primary Index

Primary index

Indexes

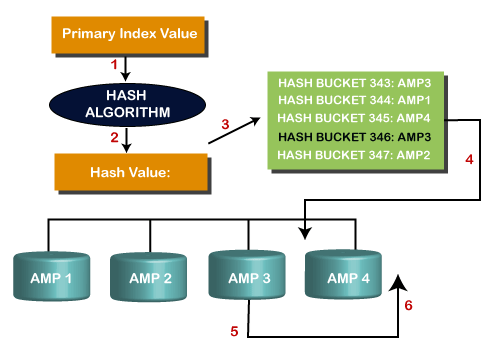
Unique Secondary Index

Secondary index

Non-Unique Secondary Index

Primary Index:

* PI used to specify where the data resides in AMP via Hashing algorithm



* Hash algorithm provides a Hash Value, the hash value will move to the hash bucket with respect to AMPs Number
* The hash map will point to the row in AMP.
* For example, User writes a query of table where he mentions EMP\_ID as PI
* PI sends data to PE, PE will provide PI value to hashing algorithm, which will provide row hash value to the data.
* Each row hash value of data assign to each AMPs.
* So, the data will be placed in the AMPs according to its row hash value which is called as Row Id.
* When retrieval data happens, the data in the respective disk will send it to AMP, AMP send it to PE via BYNET and PE will send data to the user.

TYPES OF PI:

* UPI

When table has UPI, it should not have duplicate values, if there it will be rejected.

Eg,

CREATE SET TABLE Student

(

   Roll\_no int,

   First\_name varchar2(10),

   Last\_name varchar2(10),

)

UNIQUE PRIMARY INDEX(Roll\_no);

* NUPI

It will consider many duplicate values in the same table.

In NUPI data will be send to hashing algorithm and provide hash value as incremented by one for uniqueness.

CREATE SET TABLE Employee

(

   Employee\_Id int,

   Name varchar2 (10),

   Department varchar2 (10),

)

PRIMARY INDEX(City);

* Multi-column primary index

Teradata allows more than one column can be assign as PI.

Multi-column primary index can be UPI or NUPI.

For example, In a table one or more columns can be a PI which has one hash value and stored in one AMP.

SECONDARY INDEX:

* A Table can contain two or more secondary index.
* SI is not involved in data distribution.
* It allows accessing the data without having a full-table scan, Gives particular table row.
* Unique Secondary Index and Non-Unique secondary Index are the types of secondary Index.
* AMP will create a sub table when secondary index is queried.
* Two purpose of USI is, enforces uniqueness and speed up access to a row

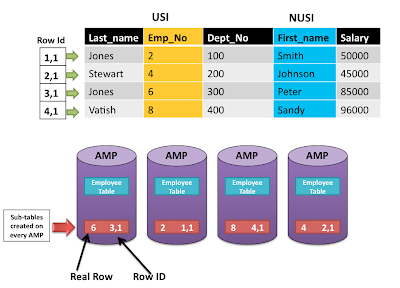
USI:

SYNTAX: CREATE UNIQUE SECONDARY INDEX ON [COL]

When A USI is created Teradata will immediately build a secondary index subtable on each AMP.

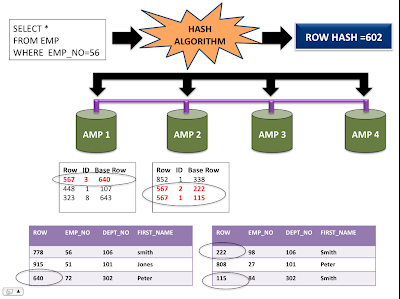
Each AMP will then hash the secondary index value for each of their rows in the base table.

Each sub table consists of Real row (which is declared as USI)and Row ID(Hash Value).



NUSI:

SYNTAX: CREATE INDEX [ COL] ON;



* When query is fired, hashing algorithm calculate hash value for that index and send it to AMP.
* AMP find row id for that respective row hash value.
* AMP whose sub table contain the row id will be participating others will be rejected.
* AMP locates hash value plus uniqueness id in the sub table, it can be from different AMP like AMP 1, AMP 2.
* AMP collects the data and send it back PE, PE send it to user.

27-9-22 PARTITION PRIMARY INDEX & JOIN INDEX

Partition primary index [PPI]:

It is one of the types of indexes. It helps to sort the data within AMP much faster than primary index. For example, Range-based query are performed much faster. It avoids full table scan.

Assume there is a costumer table, if we wanted to run a query and extract order\_date but order\_date is a PI. So PE send request to AMP for the data retrieval, it will take hash value and it will a full table scan.

Here, the concept of PPI comes to the point, order\_date is a PPI. PE sends a request to AMP; it will not take hash value instead they are partitioned by order\_date and avoid full table scan.

Example for creating a table in PPI:

CREATE SET TABLE Orders (

StoreNo SMALLINT,

OrderNo INTEGER,

OrderDate DATE FORMAT 'YYYY-MM-DD',

OrderTotal INTEGER

)

PRIMARY INDEX(OrderNo)

PARTITION BY RANGE\_N (

OrderDate BETWEEN DATE '2010-01-01' AND '2016-12-31' EACH INTERVAL '1' DAY

);

PI is added on OrderNo and PPI is added on OrderDate, but rows will go to the same AMP.

Types of PPI:

* Case partitioning
* Range-based partitioning

Here, query will be partitioned by range only.

* Multi-level partitioning

EG;, PRIMARY INDEX ( emp\_no )

PARTITION BY (

RANGE\_N(hire\_year BETWEEN 1980 AND 1988 EACH 1),

RANGE\_N(hire\_month BETWEEN 1 AND 12 Each 1,

NO RANGE OR UNKNOWN ));

HERE, query was first partitioned by year and then by month. This is called multi-level partitioning.

* Character- based partitioning

**CREATE** **TABLE** accounts (

       cust\_id INTEGER,

       last\_name VARCHAR(30) CHARACTER **SET** **UNICODE** **NOT** CASESPECIFIC,

       first\_name VARCHAR(30),

       city  VARCHAR(50))

     PRIMARY **INDEX** (cust\_id)

**PARTITION** **BY** CASE\_N (last\_name **LIKE** ‘A%’,

                          last\_name **LIKE** ‘B%’,

**NO** **CASE**,

                          UNKNOWN);

Here, CHARACTER is partitioned by CASE\_N

Partitioning Rules:

* A table can have up to 65,535 Partitions.
* Partitioning never determines which AMP gets row.
* Partitioning determines how an AMP will sort the row on its own.
* Table can have up to 15 levels of partitions.
* A table cannot have an UPI as the Primary Index if the Partition table does not include PI.
* Total 3 forms of Partitioning Simple, RANGE and CASE.

JOIN INDEXES:

The JOIN INDEX joins two table together and keep result set in the permanent space of Teradata.

User never query the join index directly. The PE will decide which result set to take.

Types of join indexes:

* Single Table Join Index (STJI)
* Multi Table Join Index (MTJI)
* Aggregated Join Index (AJI)

Single Table Join Index (STJI):

* For joining one table with other table, we use join, but we cannot change the PI. To avoid this, Teradata brings STJI, STJI duplicates a single table, but changes PI.
* User can only query the base table; PE only will decide to use JOIN INDEX.

CREATE JOIN INDEX EMP\_SNAP

AS

SELECT EMP\_NO, EMP\_NAME, EMO\_DEPT

FROM EMPLOYEE\_TABLE

PRIMARY INDEX(EMP\_DEPT);

* Reason to use this STJI, it will faster the performance of joins because no redistribution or duplicates occurs.

Multi Table Join Index (MTJI):

* Here joins index will store result of two tables.

CREATE JOIN INDEX EMP\_DEPT

AS

SELECT EMP\_NO, EMP\_NAME, EMP\_DEPT, EMP\_SAL, EMP\_MGR

FROM EMPLOYEE\_TABLE EMP

INNER JOIN DEP\_TABLE DEP

ON EMP.EMP\_DEPT = DEP.DEPT\_NO

UNIQUE PRIMARY INDEX (EMP\_NO);

Aggregated Join Index (AJI):

An aggregate JOIN index will allow the tracking of Averages SUM and COUNT on any table. This JOIN index is basically used if we need to perform any aggregate function in the data of the table.

CREATE JOIN INDEX Employee\_Salary\_JI

AS

SELECT a.DepartmentNo,SUM(b.NetPay) AS TotalPay

FROM Employee a

INNER JOIN Salary b

ON (a.EmployeeNo = b.EmployeeNo)

GROUP BY a.DepartmentNo

Primary Index (DepartmentNo);

28-9-21 TABLES & SPACES

Teradata classifies the table as SET and MULTISET. SET tables not allows duplicates but MULTISET allows duplicate values.

CREATE [SET|MULTISET] VOALTILE TABLE tablename

<table definitions>

<column definitions>

<index definitions>

ON COMMIT [DELETE|PRESERVE] ROWS

Types of tables:

* Derived table
* Volatile table
* Global Temporary table

Derived table:

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

Eg;, SELECT

Emp.EmployeeNo,

Emp.FirstName,

Empsal.NetPay

FROM

Employee Emp,

(select EmployeeNo , NetPay

from Salary

where NetPay >= 75000) Empsal

where Emp.EmployeeNo = Empsal.EmployeeNo;

Volatile table:

Volatile tables are created, used, and dropped within a user session.

EG;,

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;

Global Temporary table:

 Global Temporary table is stored in data dictionary and they can be used by many users/sessions.

CREATE SET GLOBAL TEMPORARY TABLE dept\_stat (

dept\_no INTEGER,

avg\_salary INTEGER,

max\_salary INTEGER,

min\_salary INTEGER

)

PRIMARY INDEX(dept\_no);

TYPES OF SPACES IN TERADATA:

* Permanent space
* Spool space
* Temporary space

**Permanent space:**

Permanent space is the maximum space allocated to the user to hold the data.

Permanent space is connected to AMPs, whenever data stored is full in AMPs it will generate a error message.

**Spool space:**

User without spool space cannot execute the query.

Space used by system to keep SQL query.

Spool space is divided among the number of AMPs. Whenever per AMP limit exceeds the allocated space, the user will get a spool space error.

**Temporary space:**

Temporary space is the unused permanent space which is used by Global Temporary tables.

Temp space is also divided by the number of AMPs.

Temp space is allocated at the database or user level, but not the table level.

4-10-22 OLAP function in Teradata

What is OLAP function:

* OLAP is defined as Ordered analytical function which support many common operations as analytical operations and aggregate operations.
* OLAP function result in row not a single value

Some types of OLAP function:

* RANK
* DENSE\_RANK
* ROW\_NUMBER
* PARTITION\_BY
* CSUM

RANK:

Rank Function in Teradata will help in the ordering of the records based on the columns provided.

The RANK function also helps in filtering the number of records returned based on the rank.

The RANK function will provide the flexibility to evaluate each column and compare them based on the highest or lowest order.

**Syntax of Rank function:**

**RANK() OVER**

**([ORDER BY col] [ORDER BY col ] [DESC / ASC]**

Simply rank by its order from the records like rank for student.

If two or more students got 1st rank, in this case; we should use **partition\_by function.**

**EXAMPLE QUERY:**

SELECT EmployeeNo, JoinedDate,RANK()

OVER(ORDER BY JoinedDate) as Seniority

FROM Employee;

**RESULT QUERY:**

EmployeeNo JoinedDate Seniority

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

101 2005-03-27 1

103 2007-03-21 2

102 2007-04-25 3

105 2008-01-04 4

104 2008-02-01 5

**PARTITION\_BY :**

Which groups the data by column and perform OLAP function within each group.

*Example; if two student got same rank but I have to rank the student rank columns*

**Eample query:**

SELECT EmployeeNo, JoinedDate,RANK()

OVER(PARTITION BY DeparmentNo ORDER BY JoinedDate) as Seniority

FROM Employee;

RESULT:

EmployeeNo DepartmentNo JoinedDate Seniority

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

101 1 2005-03-27 1

103 2 2007-03-21 1

102 2 2007-04-25 2

104 2 2008-02-01 3

105 3 2008-01-04 1

DENSE\_RANK:

This function is used to find rank within categories or sub-categories.

Dense Rank function will not miss next number(rank) if there are 2 records with same value.

This is also sometime called as class rank.

EXAMPLE QUERY:

SELECT emp\_name,salary,dept\_no, DENSE\_RANK() over (PARTITION BY dept\_no ORDER BY salary ) As Dense\_Rank\_by\_salary FROM tutorial\_db.employee;

RESULT:

EMP NAME DEPT NO SALARY DR BY SALARY

XXX 100 1300 1 IT’S ONE CATEGORIES

YYY 100 2000 2

TTT 200 6000 1

OOO 200 7000 2 IT’S ANOTHER CATEGORIES

EEE 100 4000 3

ROW\_NUMBER:

It will rank continues number, so it is called as row\_number.

Example query:

SELECT name, company, power, ROW\_NUMBER() OVER(ORDER BY power DESC) AS Row Rank FROM Cars

Result:

Name company power row rank

Fff hjjk 6000 1

Ggg jkbj 5900 2

CSUM: Returns the cumulative (or running) sum of a value expression for each row in a partition, assuming the rows in the partition are sorted by the *sort\_expression* list.

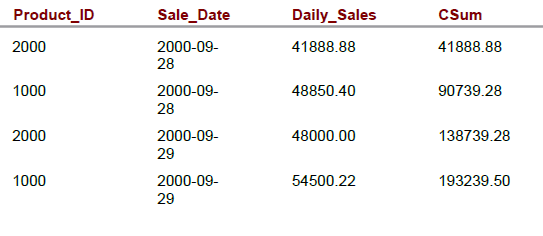
SYNTAX:

SELECT CSUM( <column-name>, <sort-key> [ASC | DESC][, <sort-key> [ASC | DESC] ... )

FROM <table-name>

[GROUP BY <column-name> [,<column-number> ... ] ];

EXAMPLE RESULT FOR CSUM



8-10-22 Teradata Utilities

What are utilities?

Utilities are a set of tools to manages DB/data.

Transferring / loading/ unloading of large amount data are done the various Teradata utilities

Type of utilities;

* BTEQ – used to interact with one or more Teradata database systems & used to rum DML, DDL function)
* TPT (Teradata parallel transporter)

[ It perform exporting, loading data from TD to file / files to TD / TD to TD/ TD to other DB]

* FASTLOAD – load a large amount of data into a empty table.
* FASTEXPORT – (transfer large amount of data from table & views of the tD to client applications )/ (used to export data from Teradata table into flat files)
* Tpump (continuously move data from data sources into Teradata tables without locking the affected table)
* MULTILOAD- Can load multi table at a time into DB.

8-10-22 JOIN STRATEGY & PERFOMANCE TUNING

SQL EXCUTION LAWS:

SELECT 5th

FROM executed 1st

WHERE 2nd

GROUP BY 3rd

HAVING 4th

ORDER BY 6th

JOIN STRATEGY:

There are three strategy works internally when use performing types of joins in the query.

There are:

* **Merge join –** one AMP operation
* **Nested join -** two AMP operation
* **Product join –** all AMP operation

Performance tuning:

Collect Strat:

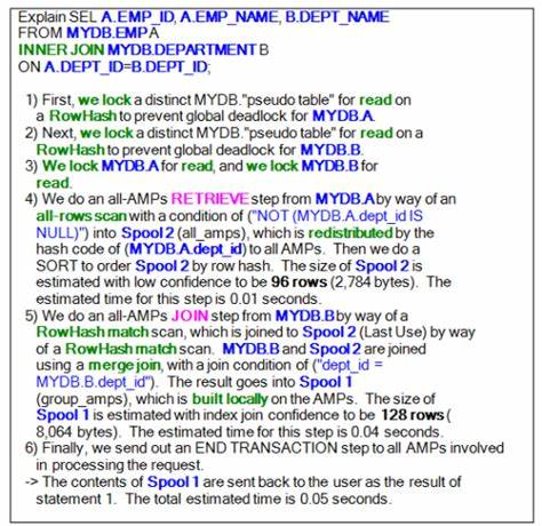
Collect Stats is used to gather the statistics on various columns as per the requirement. Thereafter the Teradata's optimizer creates an execution strategy that is based on these statistics collected by the COLLECT STATS command.​

Data Demographics​

1. The number of rows in the table.​
2. The average row size.​
3. The range of values in the column(s) where statistics are collected.​
4. Information about indexes in the table.​
5. The number of rows per value for the column where statistics are collected.​

EXPLAIN PLAN:

EXPLAIN command returns the execution plan of parsing engine​



8-10-22 PPI & join index + Macros & Stored procedures

MACROS:

Macros stores and execute SQL statements

Macro characteristics:

* Can be shared across multiple users and can be secured by providing execute access to required users.
* Can be parameterized (arguments can be passed) or non-parameterized.
* All statements within macro are processed as one transaction and if one fails that means others will be ROLLBACKED.

Syntax:

CREATE PROCEDURE <procedurename> ( [parameter 1 data type, parameter 2 data type..] )

BEGIN

<SQL or SPL statements>;

END;

SP :

A stored procedure contains a set of SQL statements and procedural statements.

They may contain only procedural statements.

The definition of stored procedure is stored in database and the parameters are stored in data dictionary tables.