Intro to Teradata

In this lecture:

- High-level view of Teradata features.
- Storage architecture.
- Retrieval architecture.

What is Teradata?

- A global leader in enterprise DWH and analytic technologies.
- Founded in 1979, presence in 60+ countries.
- Newer products that support relational and non-relational data.
- **Main competitors:** Netezza, SQL Server and Oracle.

What are Teradata features?

- A **massively parallel** processing system to ensure that tasks are processed quickly.
- A **shared nothing** architecture.
- **Linear scalability** in all dimensions of a database system workload (i.e., data volume, breadth, number of users, complexity of queries). Can scale up to 2048 nodes.

What are Teradata features (cont.)?

- **Connectivity:** Channel-attached systems such as mainframes or network-attached systems.
- **Mature Optimizer:** Refined for each release.
- **SQL:** Industry standard SQL. In addition to this, it provides its own extension.
- **Robust Utilities:** for import/export data from/to such as `FastLoad`, `MultiLoad`, `FastExport` and `TPT`.
- **Automatic Distribution:** No manual intervention for data redistribution.

Trivia: How do databases work?

Logical RDBMS Architecture

- Data is stored in **tables**, which consist of **rows** and **columns**.
- Tables can communicate with each other, so that combined information can be retrieved and analyzed.
- The way the tables and their relationships are organized is called a **schema**.

Logical RDBMS Architecture (cont.)

- **Primary Key:** Primary key is used to uniquely identify a row in a table.
  - No duplicate values are allowed in a primary key column and they cannot accept NULL values.
- **Foreign Key:** Foreign keys are used to build a relationship between the tables.
  - A foreign key in a child table is defined as the primary key in the parent table.
  - A table can have more than one foreign key.
  - It can accept duplicate values and also NULL values.
  - Foreign keys are optional in a table.

Teradata Architecture

Physical components of Teradata

- **Node:** It is the basic unit in Teradata System. Each individual server in a Teradata system is referred as a Node.
- A node consists of its own operating system, CPU, memory, own copy of Teradata RDBMS software and disk space.
- A node can contain several AMPs.
- A **cabinet** consists of one or more Nodes.

Components of Teradata

- **Parsing Engine:** Parsing Engine is responsible for receiving queries from the client and preparing an efficient execution plan.

Components of Teradata (cont.)

- The responsibilities of parsing engine are:
  - Receive the SQL query from the client.
  - Parse the SQL query check for syntax errors.
  - Check if the user has required privilege against the objects used in the SQL query.
  - Check if the objects used in the SQL actually exists.
  - Prepare the execution plan to execute the SQL query and pass it to BYNET.
  - Receives the results from the AMPs and send to the client.

Components of Teradata (cont.)

- **Message Passing Layer:** Message Passing Layer called **BYNET**, is the networking layer in Teradata system.
- BYNET allows the communication between PE and AMP and also between the nodes.
- It receives the execution plan from PE and sends to AMP.
- Similarly, receives the results from the AMPs and sends to PE.

Components of Teradata (cont.)

- **Access Module Processor (AMP):** AMPs, called as Virtual Processors (vprocs) are the one that actually stores and retrieves the data.
- AMPs receive the data and execution plan from Parsing Engine, performs any data type conversion, aggregation, filter, sorting and stores the data in the disks associated with them.

Components of Teradata (cont.)

- Records from the tables are evenly distributed among the AMPs in the system.
- Each AMP is associated with a set of disks on which data is stored. Only that AMP can read/write data from the disks.

Storage Architecture

Row Storage - Hashing algorithm

- A row is assigned to a particular AMP based on the primary index value. Teradata uses a **hashing algorithm** to determine which AMP gets the row.
- A hash function is any function that can be used to map data of arbitrary size onto data of a fixed size.
- Hash functions are used in **hash tables** or **hash maps** to quickly locate a data record.
- They are often random and suffer of **collisions**: multiple values are mapped to the same hash value.

Storage - Steps

- The client submits a query.
- 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) is used to identify the hash map entry.
- The hash map contains one AMP number.
- The hash map is an array of buckets which contains specific AMP number.

Storage - steps (cont.)

- 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 same row hash, then it increments the uniqueness ID which is a 32 bit number. For new row hash, uniqueness ID is assigned as 1 and incremented whenever a record with same row hash is inserted.
- The combination of Row hash and Uniqueness ID is called as **Row ID**.
- Row ID prefixes each record in the disk.
- Each table row in the AMP is logically sorted by their Row IDs.

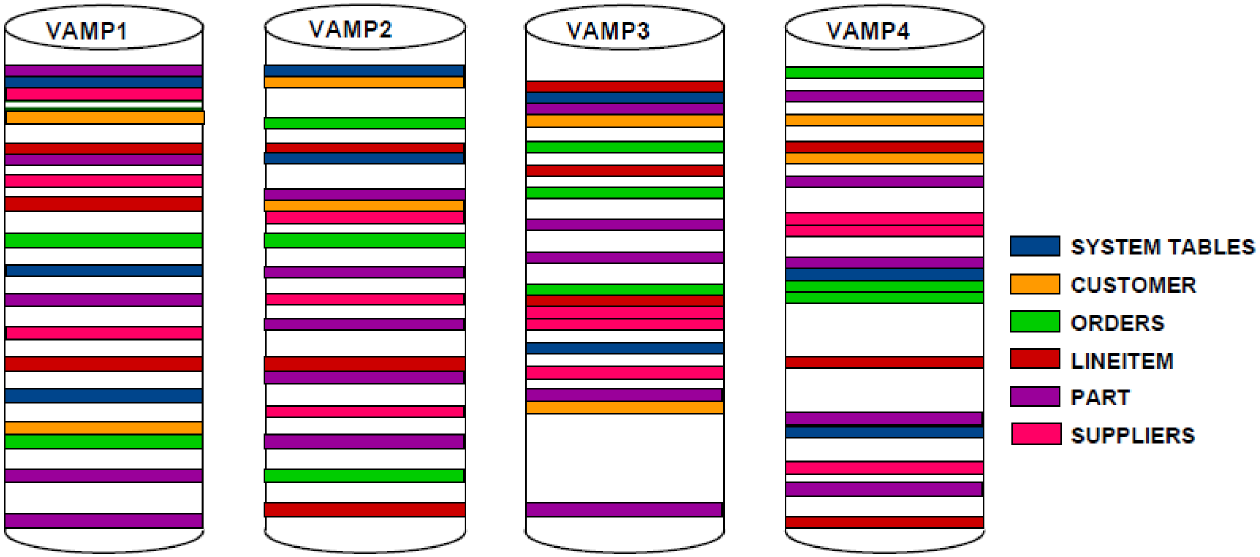
How Tables are Stored

- Tables are sorted by their Row ID (Row hash + uniqueness id) and then stored within the AMPs. Row ID is stored with each data row.

RowHash	UniquenessID	EmployeeNo	Name
2A01 2611	0000 0001	101	Mike James
2A01 2612	0000 0001	104	Alex Stuart
2A01 2613	0000 0001	102	Robert Williams
2A01 2614	0000 0001	105	Robert James
2A01 2615	0000 0001	103	Peter Paul

How does Teradata store rows?

- **Random, <u>automatic</u> data distribution & placement**
- **Real-time, automatic data reorganization**



With Teradata there is no sense of <u>ORDER</u>, therefore there is no sense of <u>DISORDER</u>, eliminating the need to <u>REORDER!</u>

Teradata
a division of NCR

Space

- **Permanent Space:** the maximum amount of space available for the user/database to hold data rows. Permanent tables, journals, fallback tables and secondary index sub-tables use permanent space. Not pre-allocated for user/database, rather defined as an upper bound.
- **Spool Space:** Spool space is the unused permanent space which is used by the system to keep the intermediate results of the SQL query. Users without spool space cannot execute any query.
- **Temp Space:** Temp space is the unused permanent space which is used by Global Temporary tables.

Retrieval Architecture

Retrieval Architecture

- When the client runs queries to retrieve records, Parsing Engine sends a request to BYNET.
- BYNET sends the retrieval request to appropriate AMPs.
- Then AMPs search their disks in parallel and identify the required records and sends to BYNET.
- BYNET then sends the records to Parsing Engine which in turn will send to the client.

Question:

- Suppose you are visiting a friend in an unknown city. How would you find their place faster:
    - Knocking on all the doors in the city.
    - Using a XII century map.
    - Using a new, high quality map?

Easier said than done

- Suppose you are visiting a friend in an unknown city. How would you find their place faster:
  - Knocking on all the doors in the city. **Full table scan**
  - Using a XII century map. **Outdated statistics**
  - Using a new, high quality map? **Good index/updated stats**

Primary Indexes

- The mechanism used to assign a row to an AMP.
- A table ~~must~~ have a **Primary Index** that cannot be changed.
  - From Teradata 13.00 tables may *not* have a primary index. Rows are randomly distributed to AMPs.
  - No PI tables are typically used as staging tables for initial load by FastLoad or TPump Array Inserts (because load is faster).
- Primary Index can be unique (**UPI**) or non-unique (**NUPI**).
- Primary Indexes are not the same as primary keys

Quick quiz:

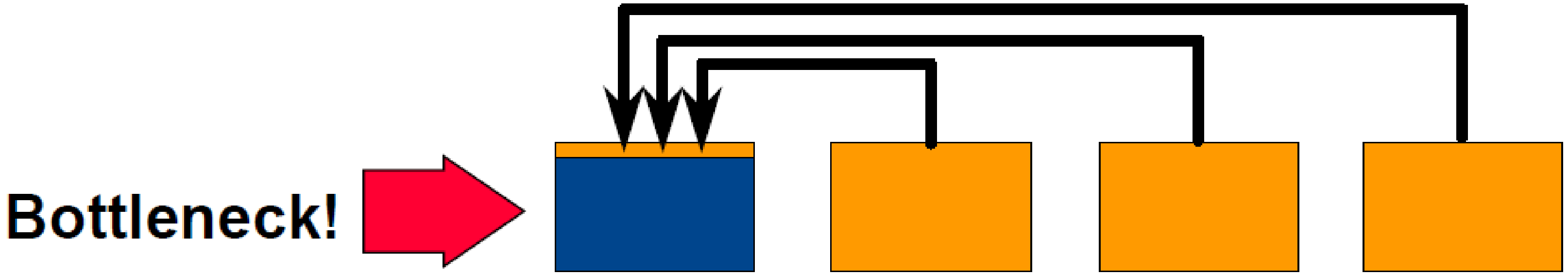
- Consider a transaction table with columns `OrderNumber` , `CustomerId` , `OrderDate` and `Total` .
- How would you assign primary keys and primary indices to make your life happier?
- **Happiness** is defined as fast running queries.

Solution

- **It depends**.
- Rows can be distributed using a `UPI` (in this case, `OrderNumber`, which is also the PK) or a `NUPI` (in this case, `CustomerId` or `OrderDate`).
- In the first case, the distribution of the rows is non-skewed across AMPs, while in the second case we will have a less even row distribution.
- But the catch is on **what level of granularity do you need for analysis**.

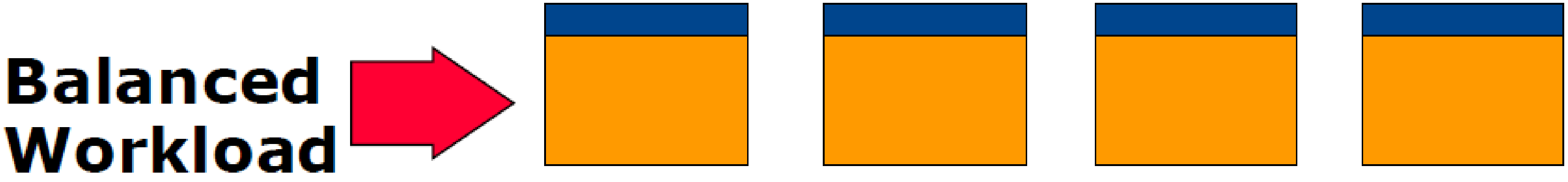
Why this matters?

- A sane index structure determines greatly the speed of our queries.
    - Indices determine storage, storage determines retrieval speed.
- In order to process records in a `JOIN`, they have to be on the same processing unit
- The records have to be sent very often to a single processing unit, to perform the `JOIN`



Avoiding bottlenecks

- The `JOIN` is performed balanced on all nodes if the primary index is chosen properly.
- Each node performs a smaller part of the `JOIN`.
- Similar bottlenecks (and solutions) apply for `GROUP BY` and `ORDER` queries.



Points to remember

- **Shared nothing**.
- Data is distributed across AMPs
- How this storage happes is determined by indices.
- Retrieval *efficiency* depends greatly on indices.