


Storing and Accessing Data row

Lesson 02: Primary Index
Mechanics

Storing and Accessing Data Rows



Copyright © Capgemini 2016. All Rights Reserved 2

How Does Teradata Store Rows?

- Teradata uses hash partitioning and distribution to randomly and evenly distribute data
- across all AMPs.
- The rows of every table are distributed among all AMPs - and ideally will be evenly distributed among all AMPs.
- Each AMP is responsible for a subset of the rows of each table. Evenly distributed tables result in evenly distributed workloads. The data is not placed in any particular order

The benefits of unordered data include:

- ☐ No maintenance needed to preserve order, and
- ☐ It is independent of any query being submitted.

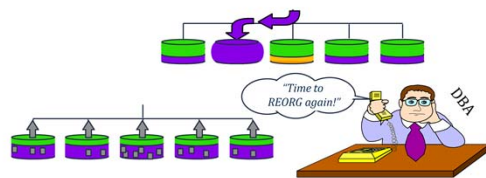
The benefits of automatic data placement include:

- ☐ Distribution is the same regardless of data
- ☐ Distribution is based on row content, not data

How do other databases Store Rows

- Many use range distribution Creates intensive maintenance for DBA
DBA's must consider:
 - How to partition the data
 - How large to make the partitions Where is there data contention How are users accessing the data

Placing all data into a single partition creates bottlenecks for all queries against that data.



Teradata DBAs never need to do costly reorganizations!

Primary Indexes

- The mechanism used to assign a row to an AMP
- A table must have a Primary Index
- The Primary Index cannot be changed

UPI UPI's guarantee even data distribution and eliminate duplicate row checking.

- ☐ If the index choice of column(s) is unique, we call this a *UPI* (Unique Primary Index).
- ☐ A UPI choice will result in even distribution of the rows of the table across all AMPs.

NUPI

- ☐ If the index choice of column(s) isn't unique, we call this a *NUPI* (Non-Unique Primary Index).
- ☐ A NUPI choice will result in even distribution of the rows of the table proportional to the degree of uniqueness of the index.

Creating a Primary Index

- A Primary Index is defined at table creation.
- It may consist of a single column or a combination of up to 16 columns.

UPI

```
CREATE TABLE sample_1  
  (col_a INT  
   ,col_b INT  
   ,col_c INT)  
  UNIQUE PRIMARY INDEX (col_b);
```

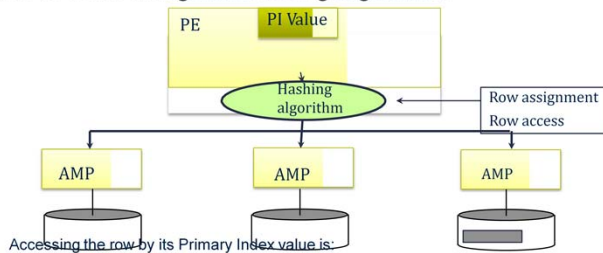
NUPI

```
CREATE TABLE sample_2  
  (col_x INT  
   ,col_y INT  
   ,col_z INT)  
  PRIMARY INDEX (col_x);
```

Note: Changing the Primary Index requires dropping and recreating the table.

Primary Index Values

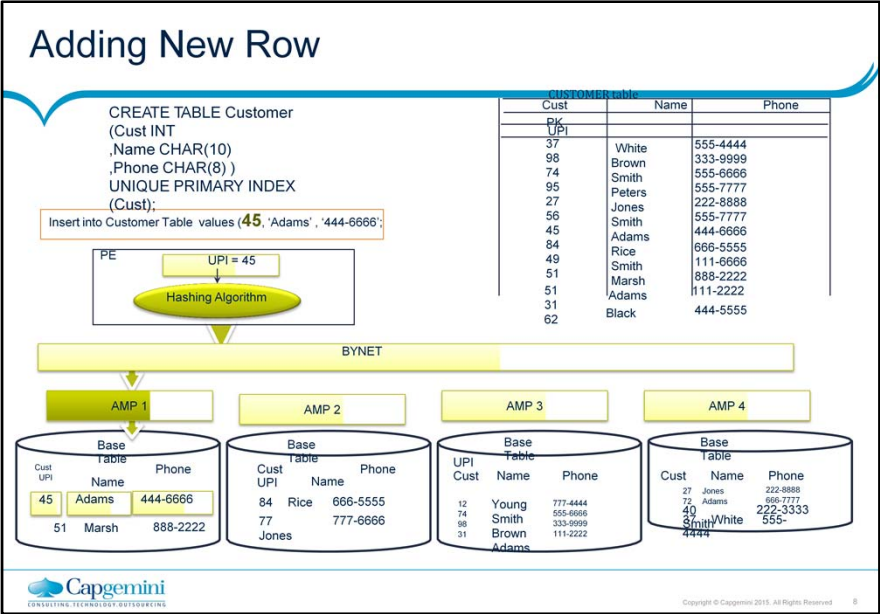
- The value of the Primary Index for a specific row determines its AMP assignment.
- This is done using the hashing algorithm.

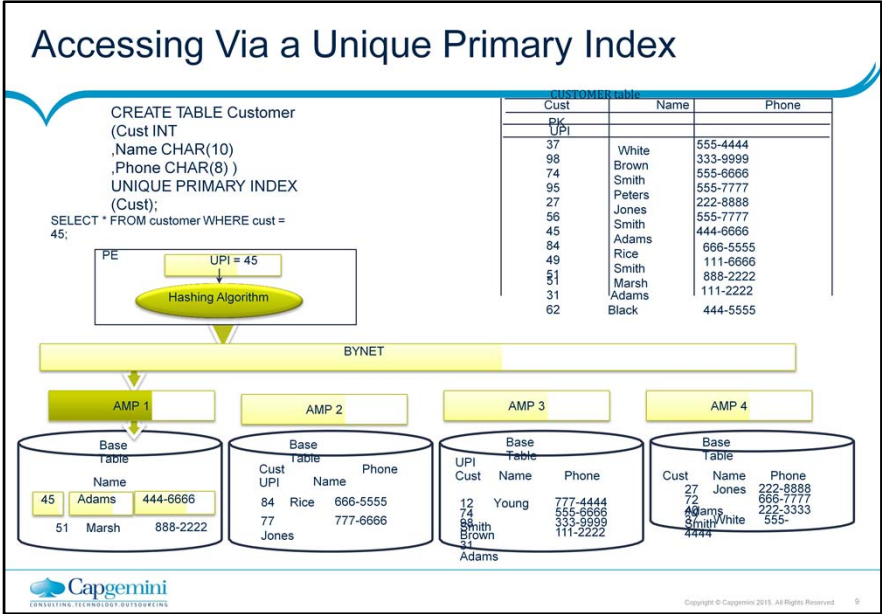


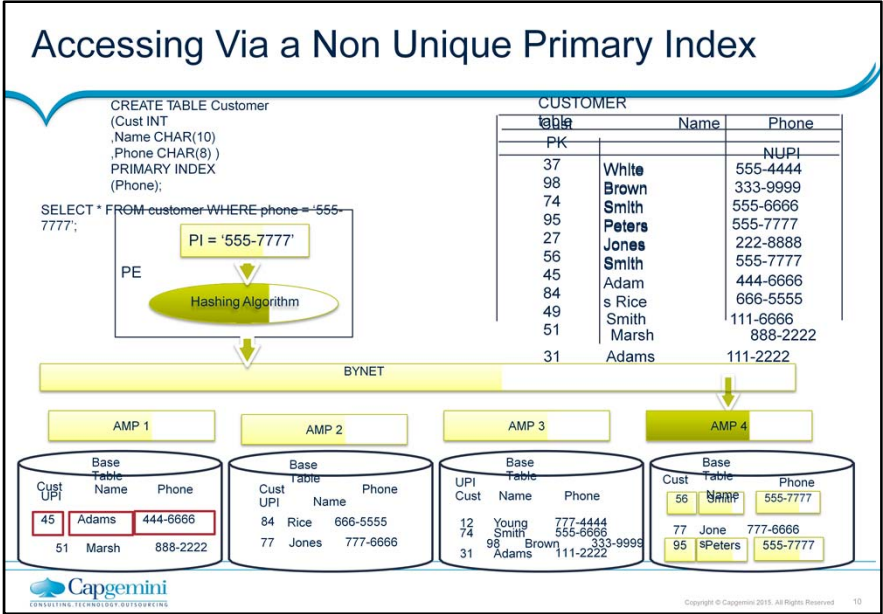
- ☐ Always a *one-AMP* operation
- ☐ The most efficient way to access a row

Other table access techniques:

- Secondary index access
- Full table scans







Primary Keys and Primary Indexes

- Indexes are conceptually different from keys:
 - A PK is a relational modeling convention which allows each row to be uniquely identified.
 - A PI is a Teradata convention which determines how the row will be stored and accessed.

Primary Key

Logical concept of data modeling
Teradata doesn't need to recognize
No limit on column numbers
Documented in data model (Optional in CREATE TABLE)
Must be unique
Uniquely identifies each row
Values should not change
May not be NULL—requires a value
Does not imply an access path
Chosen for logical correctness

Primary Index

Physical mechanism for access and storage
Each table must have exactly one
16-column limit
Defined in CREATE TABLE statement
May be unique or non-unique
Used to place and locate each row on an AMP
Values may be changed (Del+ Ins)
May be NULL
Defines most efficient access path
Chosen for physical performance

- A significant percentage of tables may use the same columns for both the PK and the PI.
- A well-designed database will use a PI that is different from the PK for some tables.

Row Distribution Using an UPI

Order

Order Number	Customer Number	Order Date	Order Status
PK			
UPI			
7325	2	4/13	O
7324	3	4/13	O
7415	1	4/13	C
7103	1	4/10	C
7225	2	4/15	C
7384	1	4/12	C
7402	3	4/16	C
7188	1	4/13	C
7202	2	4/09	C

AMP 1

7202	2	4/09	C
7415	1	4/13	C

AMP 2

7325	2	4/13	
7103	1	4/10	C
7402	3	4/16	C

AMP 3

7188	1	4/13	
7225	2	4/15	

AMP 4


7324	3	4/13	C
7384	1	4/12	C

The PK column(s) will often be used as a UPI.
PI values for Order_Number are known to be unique (it's a PK).

☐ Teradata will distribute different index values evenly across all AMPs.

☐ AMPs.

Resulting row distribution among AMPs is uniform.



Capgemini

FORGING TECHNOLOGY OUTCOMES

Copyright © Capgemini 2016. All Rights Reserved

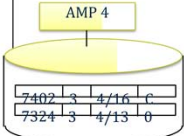
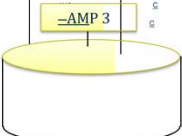
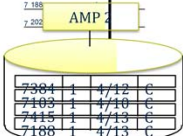
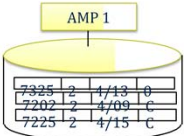
12

Page 02-12

Row Distribution Using an NUPI

Order

Order Number	Customer Number	Order Date	Order Status
PK		NUPI	
7325	2	4/13	Q
7384	3	4/13	Q
7415	1	4/13	G
7103	1	4/10	Q
7225	2	4/15	G
7284	1	4/12	G
7402	3	4/16	G
7188			G
7202			G



- Customer_Number may be the preferred access column for ORDER table, thus a good index candidate.
- Values for Customer_Number are non-unique and therefore a NUPI.
- Rows with the same PI value distribute to the same AMP causing row distribution to be less uniform or skewed.

Row Distribution Using a Highly Non Unique Index

Order

Order Number PK	Customer Number	Order Date	Order Status
7325	2	4/13	NUPI
7324	3	4/13	O O
7415	1	4/13	C O
7103	1	4/10	C C
7225	2	4/15	C C
7354	2	4/12	C
7402	3	4/15	
7185	1	4/13	
7202	2	4/09	

AMP 1

7402	3	4/15	O
7202	2	4/09	C
7452	2	4/13	C
7418	1	4/13	C
7185	1	4/13	C
7354	1	4/12	C


AMP 2

AMP 3

7103	1	4/10	O
7324	3	4/13	O
7325	2	4/13	O

AMP 4

- Values for Order_Status are highly non-unique, and therefore, it is a NUPI.
- Only two values exist, so only two AMPs will ever be used for this table.
- This table will not perform well in parallel operations.
- Highly non-unique columns are poor PI choices.
- The degree of uniqueness is critical to efficiency.



Capgemini
FORGING THE TECHNOLOGY FUTURE

Copyright © Capgemini 2016. All Rights Reserved 14

Secondary Indexes

Secondary Indexes

- A secondary index is an alternate path to the rows of a table.
- A table can have from 0 to 32 secondary indexes. Secondary indexes:
 - Do not affect table distribution.
 - Add overhead, both in terms of disk space and maintenance.
 - May be added or dropped dynamically as needed. Are chosen to improve table performance.

There are three general ways to access a table:

- | | |
|---|---------------------------------|
| <input type="checkbox"/> Primary index access | (<i>one-AMP</i> access) |
| <input type="checkbox"/> Secondary index access | (<i>two-or all-AMP</i> access) |
| <input type="checkbox"/> Full Table Scan | (<i>all-AMP</i> access) |

Choosing a Secondary Index

A secondary index may be defined:

- >At table creation (CREATE TABLE)
- > Following table creation (CREATE INDEX)
- > Using up to 16 columns

USI

- ☐ If the index choice of column(s) is unique, it is called a USI (unique secondary index).
- ☐ Accessing a row via a USI typically requires 2 AMPs.

NUS

- ☐ If the index choice of column(s) is non-unique, it is called a NUSI (non-unique secondary index).
- ☐ Accessing a row via a NUSI requires all AMPs.

USI

CREATE UNIQUE INDEX

(employee-number) on employee

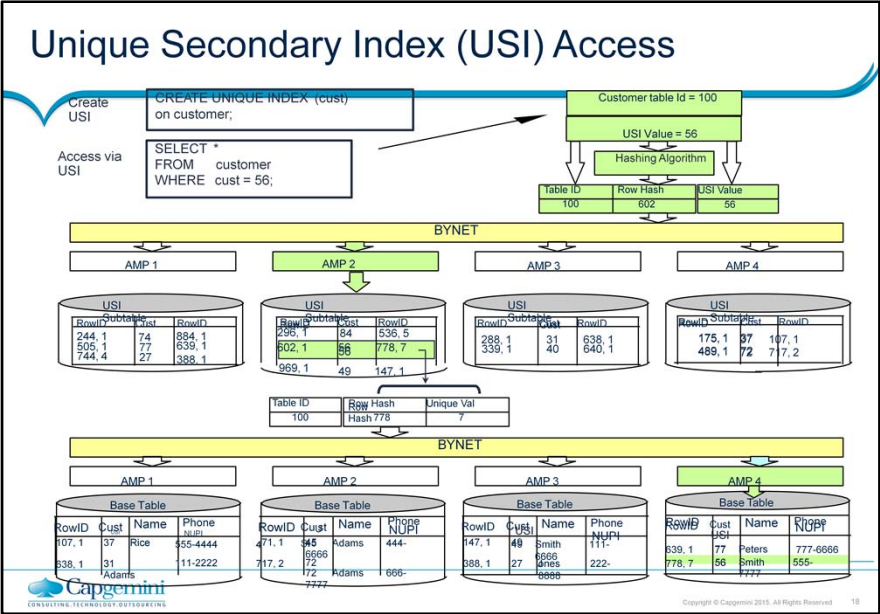
NUS

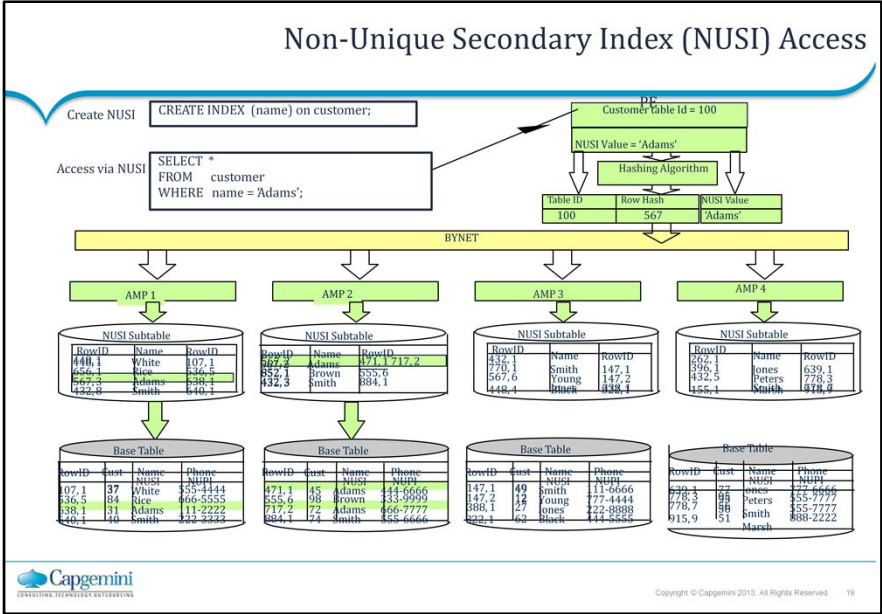
CREATE INDEX (last-name) on employee

CREATE INDEX (last-name, first-name) on employee

Note:

- ☐ Secondary indexes cause an internal sub-table to be built.
- ☐ Dropping the index causes the sub-table to be deleted.





Comparison of Primary and Secondary Indexes

Index Feature	Primary	Secondary
Required?	Yes	No
Number per Table	1	0-32
Max Number of Columns	16	16
Unique or Non-Unique?	Both	Both
Affects Row Distribution	Yes	No
Created/Dropped Dynamically	No	Yes
Improves Access	Yes	Yes
Multiple Data Types	Yes	Yes
Separate Physical Structure	None	Sub-table
Extra Processing Overhead	No	Yes

Full – Table Scans

- Every row of the table must be read.
- All AMPs scan their portion of the table in parallel. Primary Index choice affects FTS performance.
- Full-table scans typically occur when either:
 - The index columns are not used in the query
 - An index is used in a non-equality test
 - A range of values is specified for the primary index

CUSTOMER

Cust_ID	Cust_Name	Cust_Phone
USI	NUSI	NUPI

Examples of Full-Table Scans:

```
SELECT * FROM customer WHERE Cust_Phone LIKE '524-____';  
SELECT * FROM customer WHERE Cust_Name <> 'Davis';  
SELECT * FROM customer WHERE Cust_ID > 1000;
```