

Storing and Accessing Data row

Lesson 02: Primary Index
Mechanics

Storing and Accessing Data Rows

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.

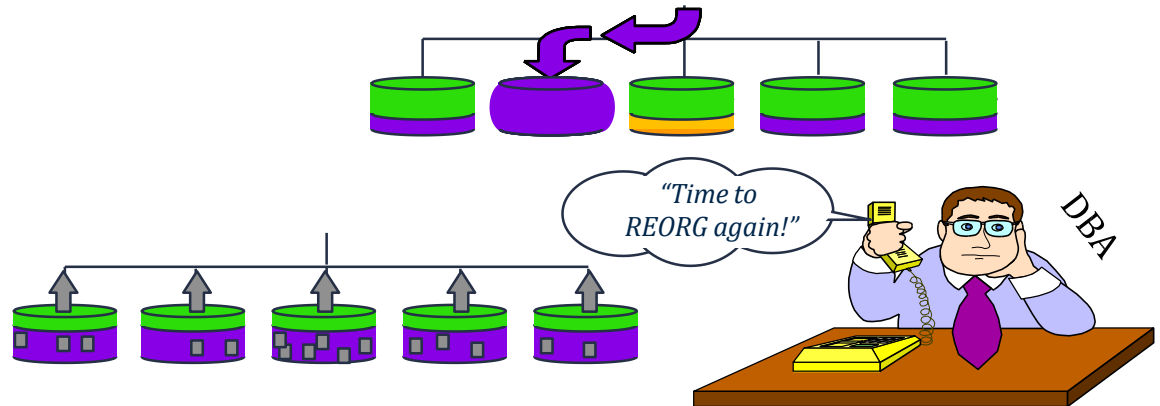
The benefits of automatic data placement include:

- 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);
```

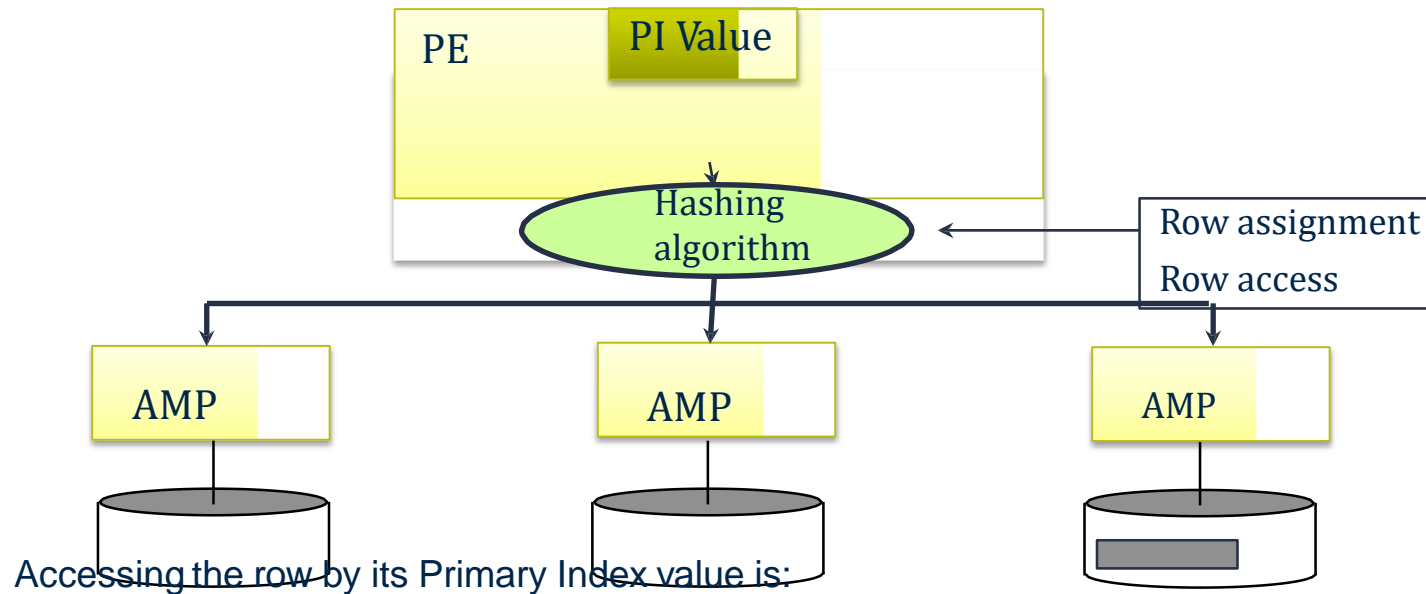
NUPI

```
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

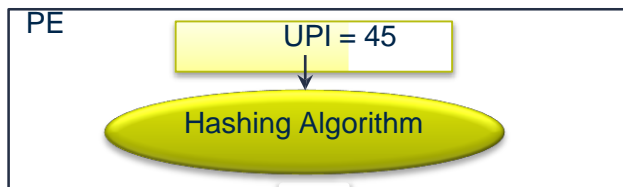
Other table access techniques:

- Secondary index access
- Full table scans

Adding New Row

```
CREATE TABLE Customer
(Cust INT
,Name CHAR(10)
,Phone CHAR(8) )
UNIQUE PRIMARY INDEX
(Cust);
```

Insert into Customer Table values (45, 'Adams', '444-6666');



BYNET

AMP 1

AMP 2

AMP 3

AMP 4

Base
Table

Base
Table

Base
Table

Base
Table

Cust UPI	Name	Phone
45	Adams	444-6666
51	Marsh	888-2222

Cust UPI	Name	Phone
84	Rice	666-5555
77	Jones	777-6666

UPI Cust	Name	Phone
12	Young	777-4444
74	Smith	555-6666
98	Smith	333-9999
31	Brown	111-2222
	Adams	

Cust	Name	Phone
27	Jones	222-8888
72	Adams	666-7777
40	White	222-3333
37	Smith	555-4444

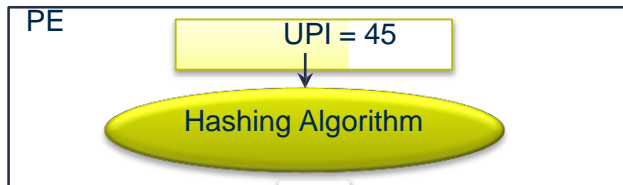
CUSTOMER table

Cust	Name	Phone
PK UPI		
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
51	Adams	111-2222
31	Black	444-5555
62		

Accessing Via a Unique Primary Index

```
CREATE TABLE Customer
(Cust INT
,Name CHAR(10)
,Phone CHAR(8) )
UNIQUE PRIMARY INDEX
(Cust);
```

```
SELECT * FROM customer WHERE cust =
45;
```



CUSTOMER table

Cust	Name	Phone
PK		
UPI		
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adams	444-6666
84	Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
31	Adams	111-2222
62	Black	444-5555

BYNET

AMP 1

AMP 2

AMP 3

AMP 4

Base
Table

Base
Table

Base
Table

Base
Table

Name	Phone
45 Adams 444-6666	
51 Marsh 888-2222	

Cust	Name	Phone
84	Rice	666-5555
77	Jones	777-6666

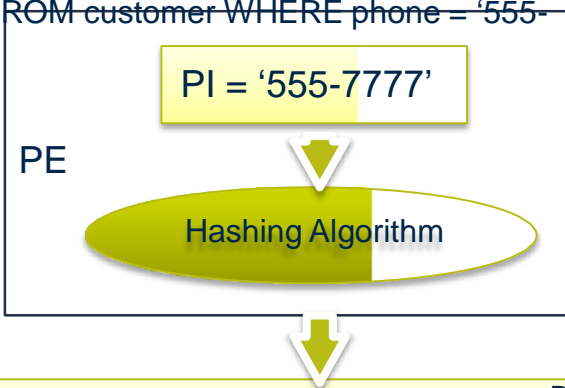
UPI	Cust	Name	Phone
12		Young	777-4444
74		Smith	555-6666
98		Brown	333-9999
31		Adams	111-2222

Cust	Name	Phone
27	Jones	222-8888
72	Adams	666-7777
37	Smith	222-3333
98	White	555-4444

Accessing Via a Non Unique Primary Index

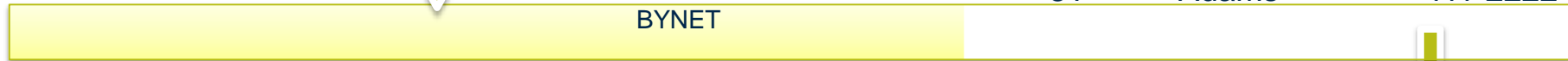
```
CREATE TABLE Customer
(Cust INT
,Name CHAR(10)
,Phone CHAR(8) )
PRIMARY INDEX
(Phone);
```

```
SELECT * FROM customer WHERE phone = '555-7777';
```



CUSTOMER

Cust	Name	Phone
37	White	555-4444
98	Brown	333-9999
74	Smith	555-6666
95	Peters	555-7777
27	Jones	222-8888
56	Smith	555-7777
45	Adam	444-6666
84	s Rice	666-5555
49	Smith	111-6666
51	Marsh	888-2222
31	Adams	111-2222



AMP 1

Cust UPI	Name	Phone
45	Adams	444-6666
51	Marsh	888-2222

AMP 2

Cust UPI	Name	Phone
84	Rice	666-5555
77	Jones	777-6666

AMP 3

UPI Cust	Name	Phone
12	Young	777-4444
74	Smith	555-6666
98	Brown	333-9999
31	Adams	111-2222

AMP 4

Cust	Name	Phone
56	Smith	555-7777
77	Jone	777-6666
95	SPeters	555-7777

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

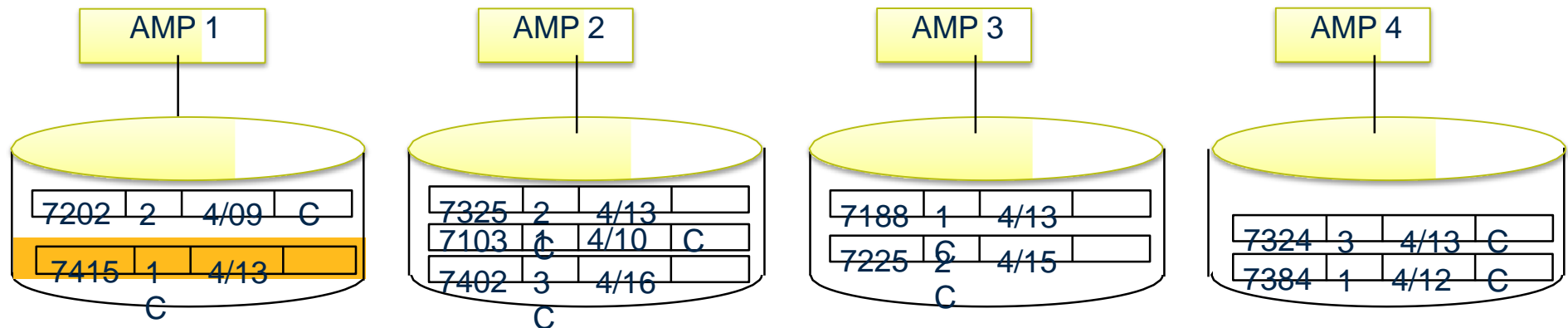
Physical mechanism for access and storage
Each table must have exactly one
16-column limit
Defined in CREATE TABLE statement
May be unique or non-unique
Used to place and locate each row on an AMP
Values may be changed (Del+ Ins)
May be NULL
Defines most efficient access path
Chosen for physical performance

- 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	O
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



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.

Resulting row distribution among AMPs is uniform.

Row Distribution Using an NUPI

Order

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

AMP 1

7325	2	4/13	O
7202	2	4/09	C
7225	2	4/15	C

AMP 2

7384	1	4/12	C
7103	1	4/10	C
7415	1	4/13	C
7188	1	4/13	C

AMP 3

7402	3	4/16	C
7324	3	4/13	O

AMP 4

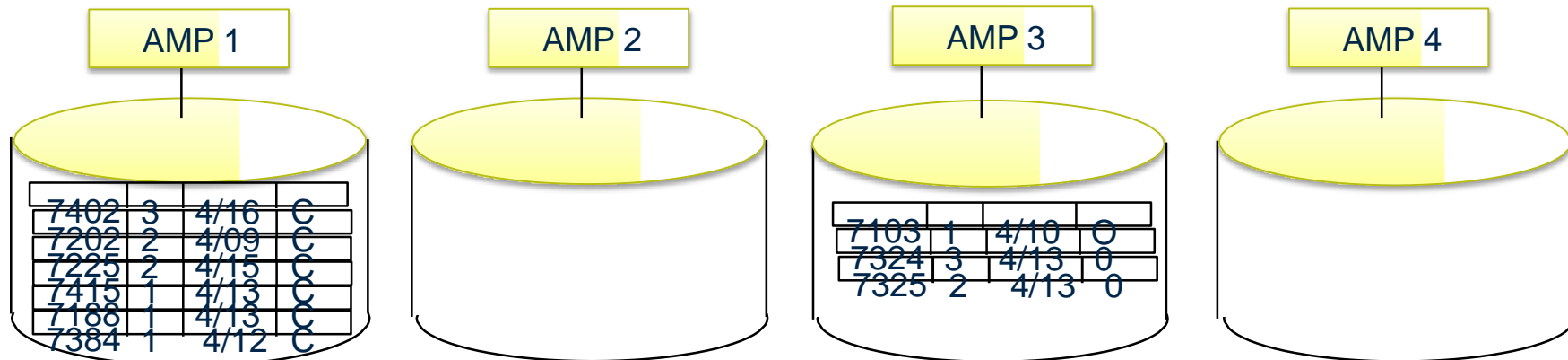
7402	3	4/16	C
7324	3	4/13	O

- 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	Customer Number	Order Date	Order Status
PK			
			NUPI
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



- 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.

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:

- ☐ Primary index access (*one*-AMP access)
- ☐ Secondary index access (*two*-or *all*-AMP access)
- ☐ Full Table Scan (*all*-AMP 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.

Unique Secondary Index (USI) Access

Create USI

```
CREATE UNIQUE INDEX (cust)
on customer;
```

Access via USI

```
SELECT *
FROM customer
WHERE cust = 56;
```

Customer table Id = 100

USI Value = 56

Hashing Algorithm

Table ID	Row Hash	USI Value
100	602	56

BYNET

AMP 1

AMP 2

AMP 3

AMP 4

USI Subtable			
RowID	Subtable	Cust	RowID
244, 1	74	884, 1	
505, 1	77	639, 1	
744, 4	27	388, 1	

USI Subtable			
RowID	Subtable	Cust	RowID
296, 1	84	536, 5	
602, 1	56	778, 7	
969, 1	49	147, 1	

USI Subtable			
RowID	Subtable	Cust	RowID
288, 1	31	638, 1	
339, 1	40	640, 1	

USI Subtable			
RowID	Subtable	Cust	RowID
175, 1	37	107, 1	
489, 1	72	717, 2	

Table ID	Row Hash	Unique Val
100	Hash 778	7

BYNET

AMP 1

AMP 2

AMP 3

AMP 4

Base Table			
RowID	Cust	Name	Phone NUPI
107, 1	37	Rice	555-4444
638, 1	31	Adams	111-2222

Base Table			
RowID	Cust	Name	Phone NUPI
471, 1	345	Adams	444-
6666	72	Adams	666-
717, 2	72	Adams	666-
7777			

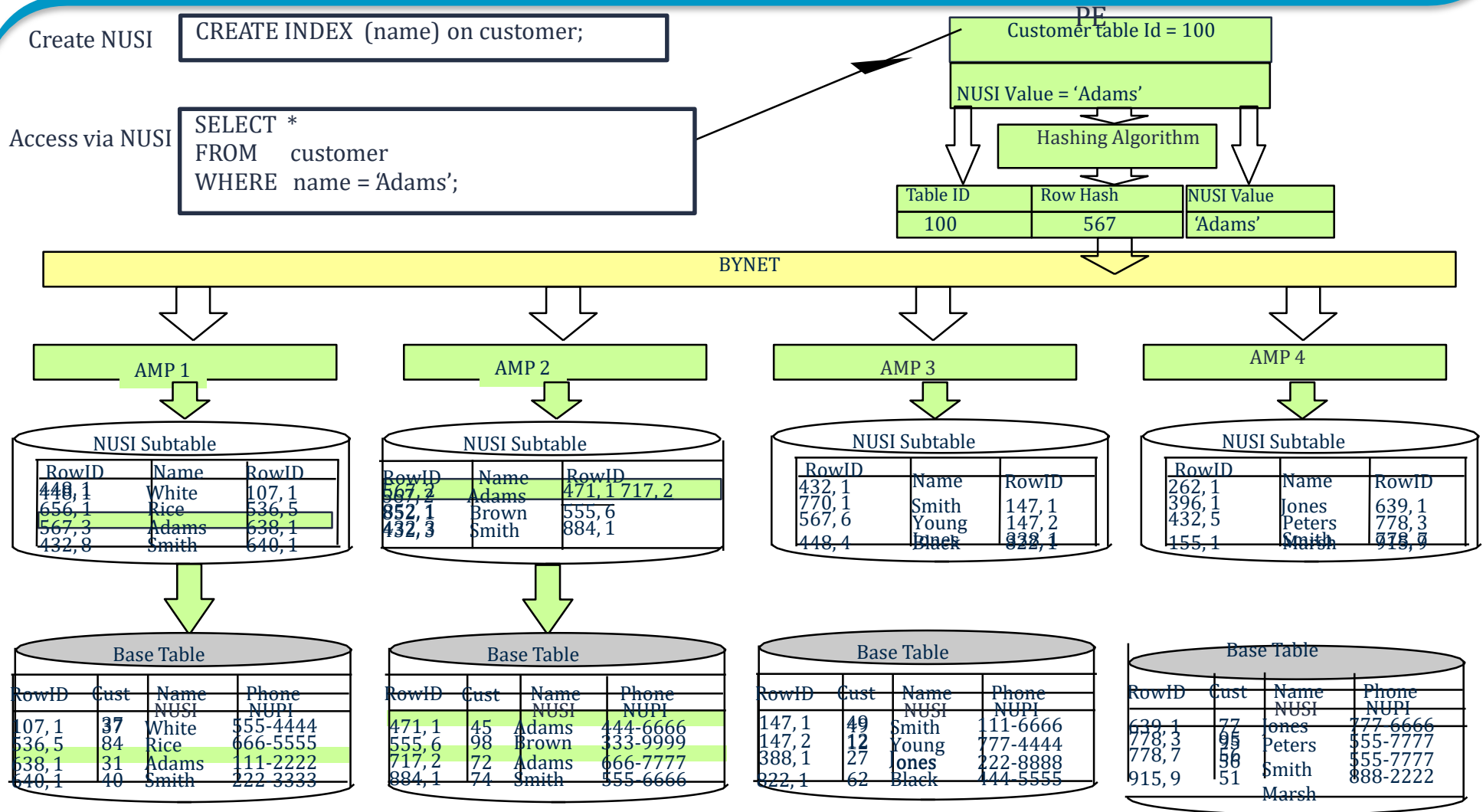
Base Table			
RowID	Cust	Name	Phone NUPI
147, 1	49	Smith	111-
388, 1	27	ones	222-
		8888	

Base Table			
RowID	Cust	Name	Phone NUPI
639, 1	77	Peters	777-6666
778, 7	56	Smith	555-
		7777	

Non-Unique Secondary Index (NUSI) Access

Create NUSI `CREATE INDEX (name) on customer;`

Access via NUSI `SELECT *
FROM customer
WHERE name = 'Adams';`



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;
```