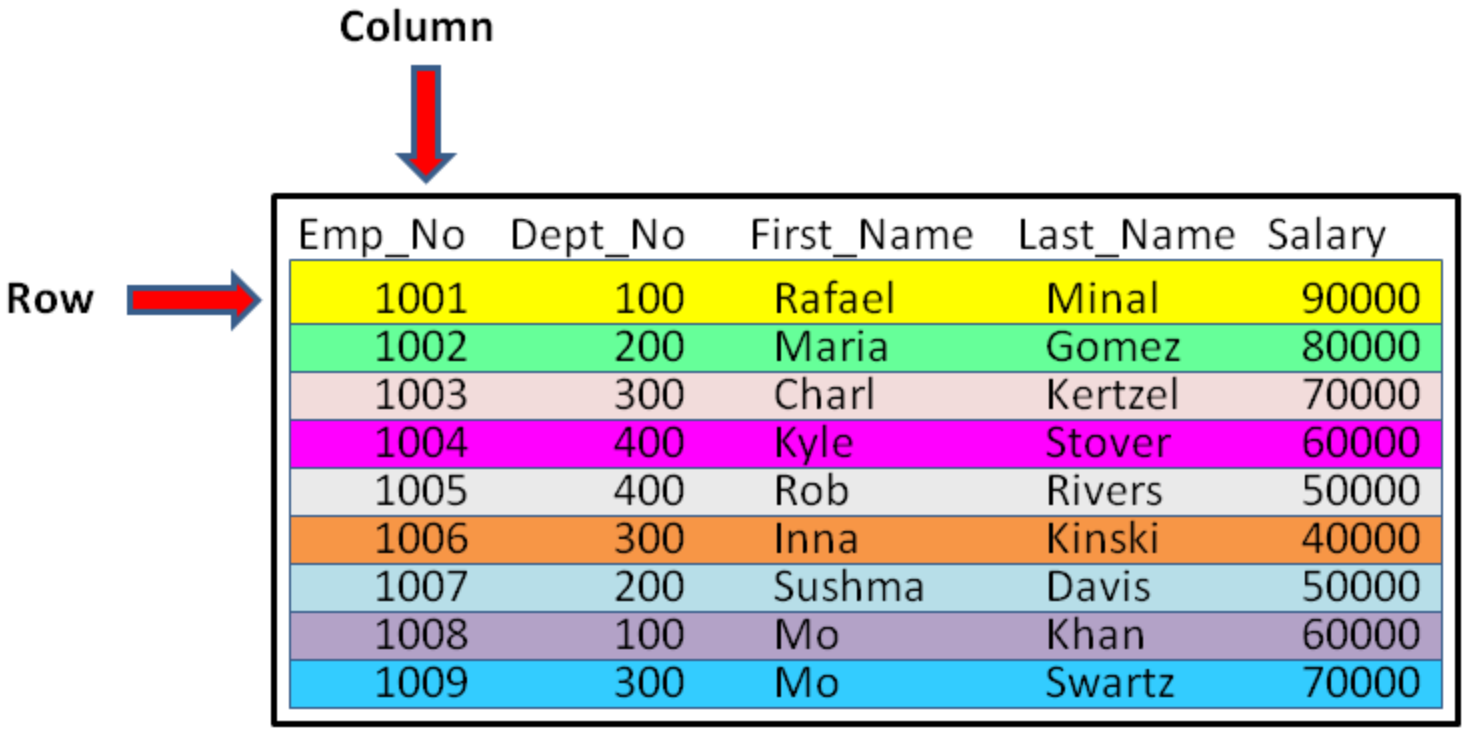
Teradata and Temporal Tables for DAMA

Tom Coffing (Tera-Tom)
CEO, Coffing Data Warehousing
Tom.Coffing@CoffingDW.Com
Direct: 513 300-0341
Website: www.CoffingDW.com

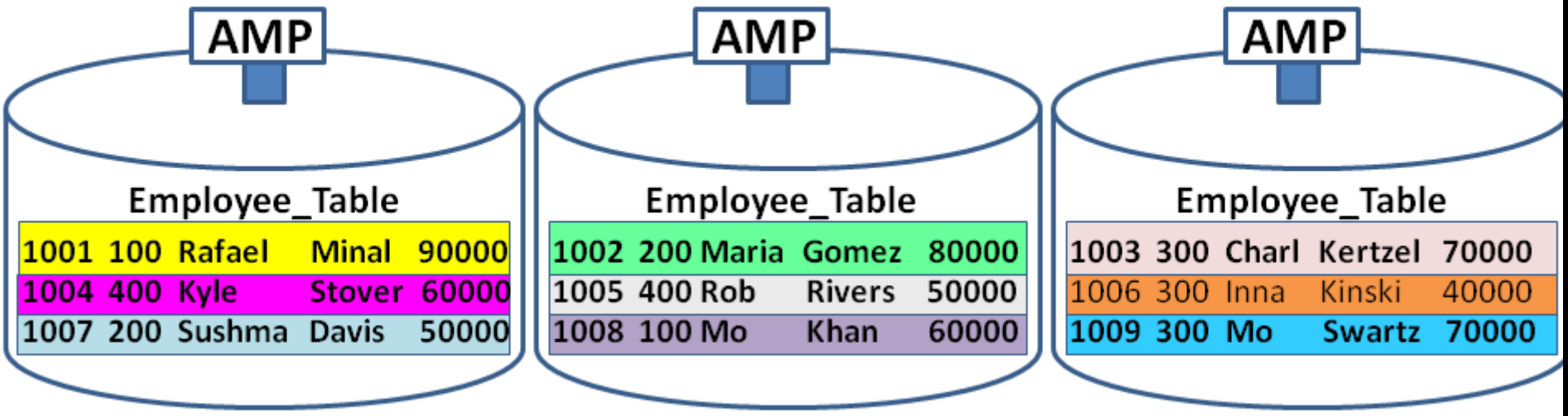
Relational Databases store their data in Tables

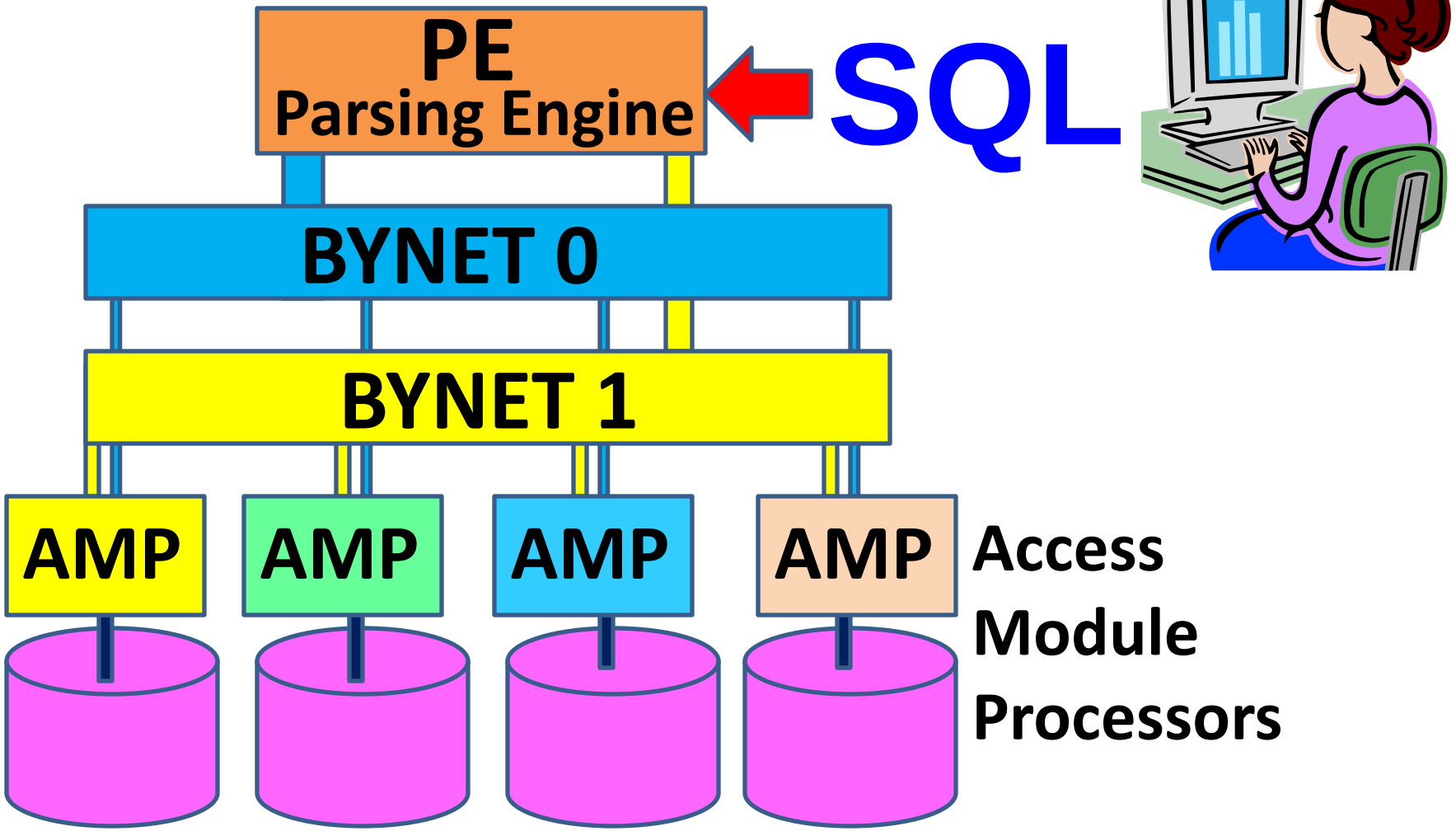


A Table has rows and columns.

Emp_No	Dept_No	First_Name	Last_Name	Salary
1001	100	Rafael	Minal	90000
1002	200	Maria	Gomez	80000
1003	300	Charl	Kertzel	70000
1004	400	Kyle	Stover	60000
1005	400	Rob	Rivers	50000
1006	300	Inna	Kinski	40000
1007	200	Sushma	Davis	50000
1008	100	Mo	Khan	60000
1009	300	Mo	Swartz	70000

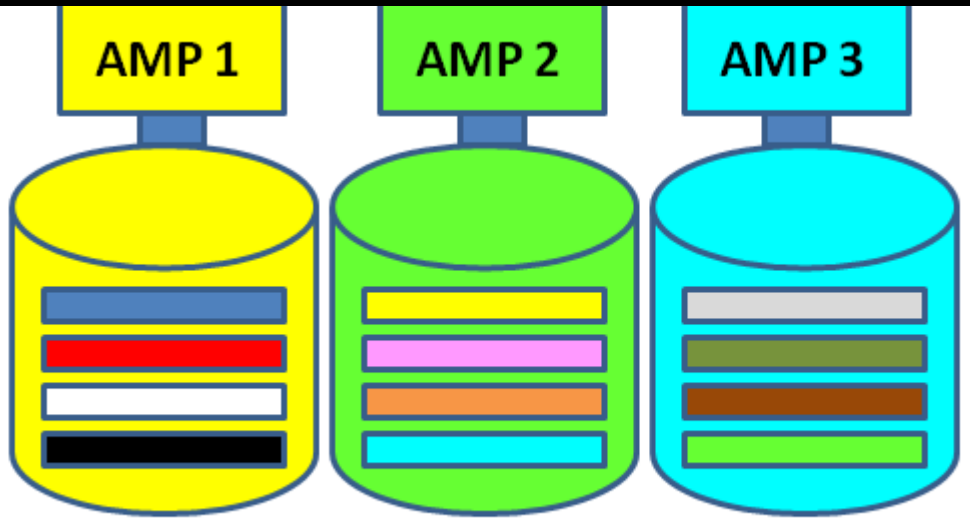
Teradata spreads rows across AMPs that have Disks



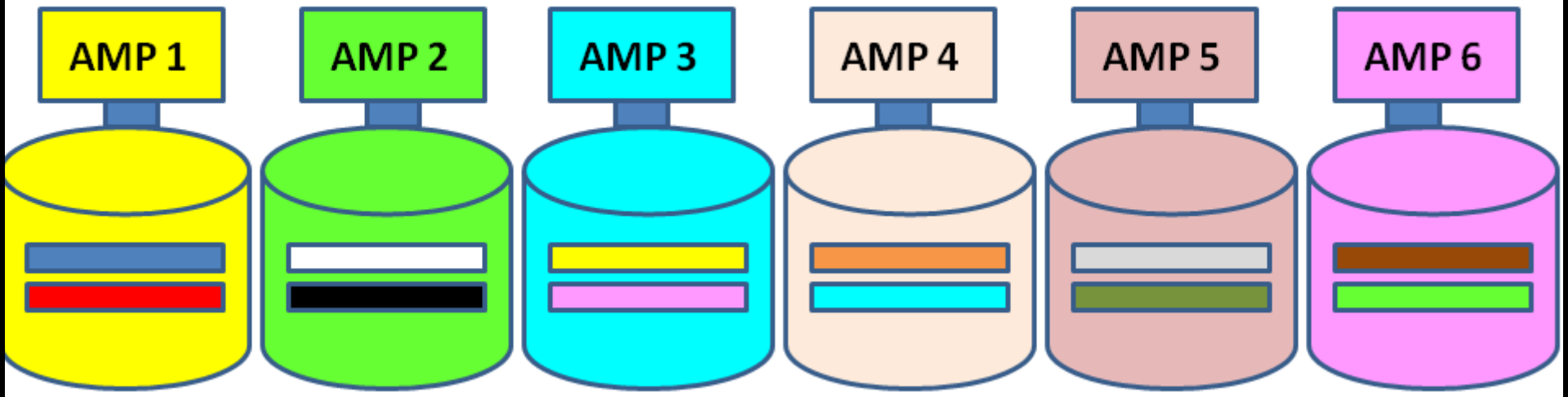


When users login to Teradata a Parsing Engine will take each SQL request and come up with a Plan for the AMPs to retrieve the data from their disks.

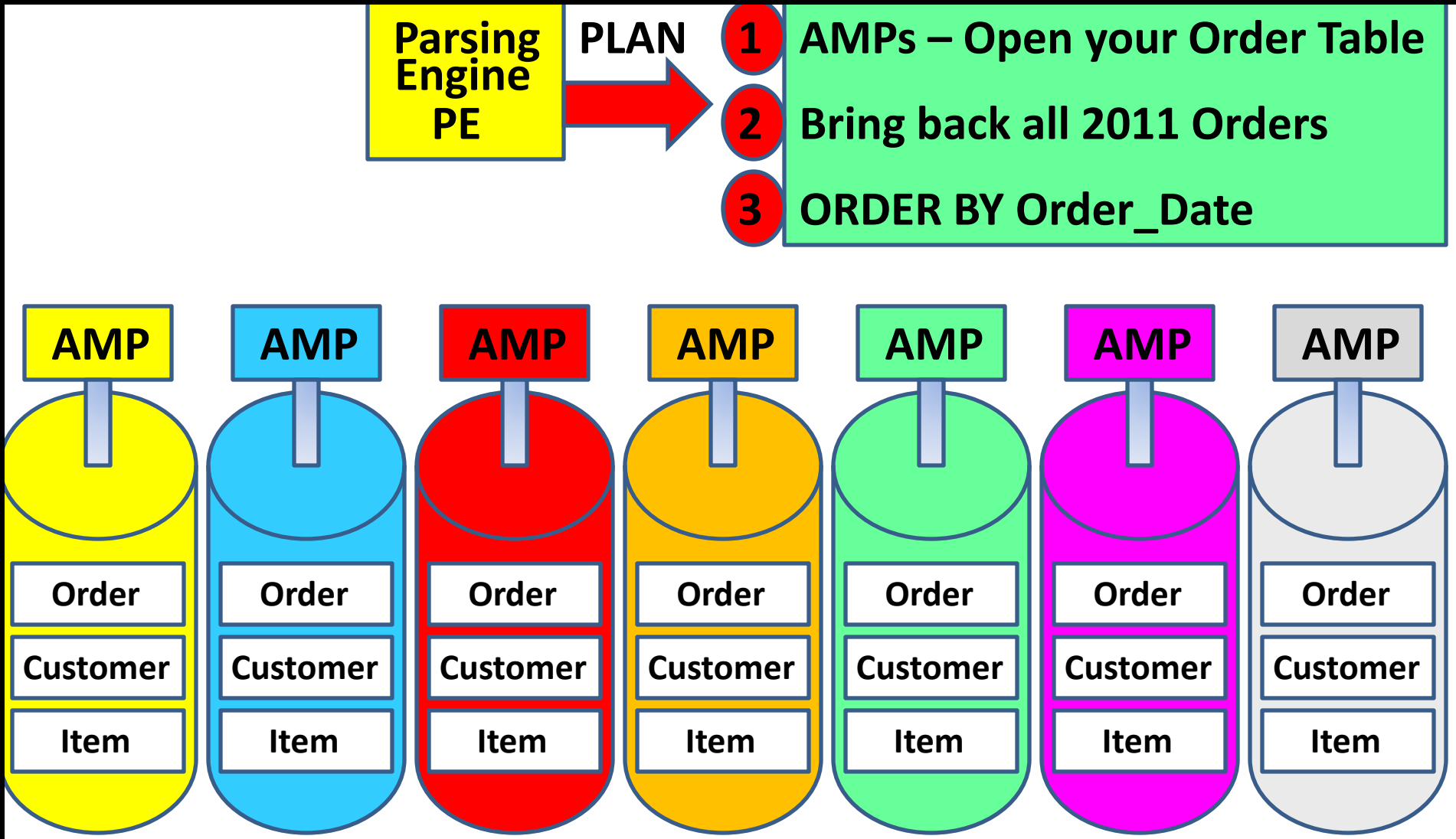
12 Rows of the Order_Table Spread across a 3-AMP System



As AMPs are added the data re-spreads the rows for each Table



Each time AMPs are added to the Teradata system the rows of each table are redistributed.



Logical View of Teradata

All tables are spread across all AMPs in Teradata to take advantage of the parallel processing.

Every Teradata Table has one and only one Primary index which does 3 things.

1. Distributes the rows to the proper AMP
2. Fastest way to retrieve a row(s)
3. Incredibly important for Joins

There are two types of Primary Indexes:

UPI – Unique Primary Index

NUPI – NON-Unique Primary Index

When is the Primary Index Created?

```
CREATE Table Employee_Table *
(
 Emp_No          Integer    Not Null
,Dept_No         Integer
,First_Name      Varchar(12)
,Last_Name       Char(20)
,Salary          Decimal (10,2)
)
Unique Primary Index (Emp_No) ;
```

The Primary Index is created at table create time.

UPI

Emp_No	Dept_no	First_Name	Last_Name	Salary
2	100	Rakish	Ratel	50000.00
4	200	Vu	Vatish	45000.00
6	300	Mary	Mason	98000.00
8	400	Leona	Lacy	15500.00
10	400	Sandy	Stewart	84000.00
12	400	Matt	Mason	65000.00
14	100	Javier	Jones	47000.00
16	200	Shelby	Stewart	52000.00

```
SELECT * FROM Employee_Table
WHERE Emp_No = 16 ;
```

If users use the Primary Index in the WHERE Clause of their SQL it is a 1-AMP operation every time.

NUPI

Emp_No	Dept_no	First_Name	Last_Name	Salary
2	100	Rakish	Ratel	50000.00
4	200	Vu	Vatish	45000.00
6	300	Mary	Mason *	98000.00
8	400	Leona	Lacy	15500.00
10	400	Sandy	Stewart *	84000.00
12	400	Matt	Mason *	65000.00
14	100	Javier	Jones	47000.00
16	200	Shelby	Stewart *	52000.00

```
SELECT * FROM Employee_Table
WHERE Last_Name = 'Mason';
```

If users use the Primary Index in the WHERE Clause of their SQL it is a 1-AMP operation every time.

UPI

Department

Dept_No	Department_Name	Mgr_No	Budget
100	Sales	2000000	500000.00

Row

Take UPI value 100

Hash with secret Math Formula

Value 100 has a 32-bit Row Hash of 000000110

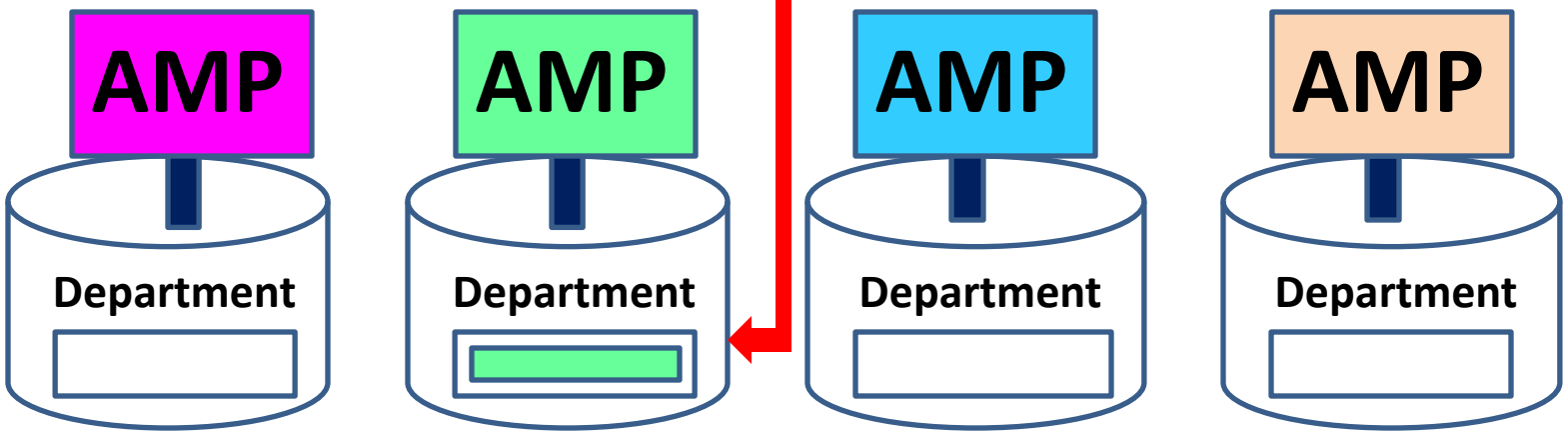
=6

Count 6 steps In Hash Map

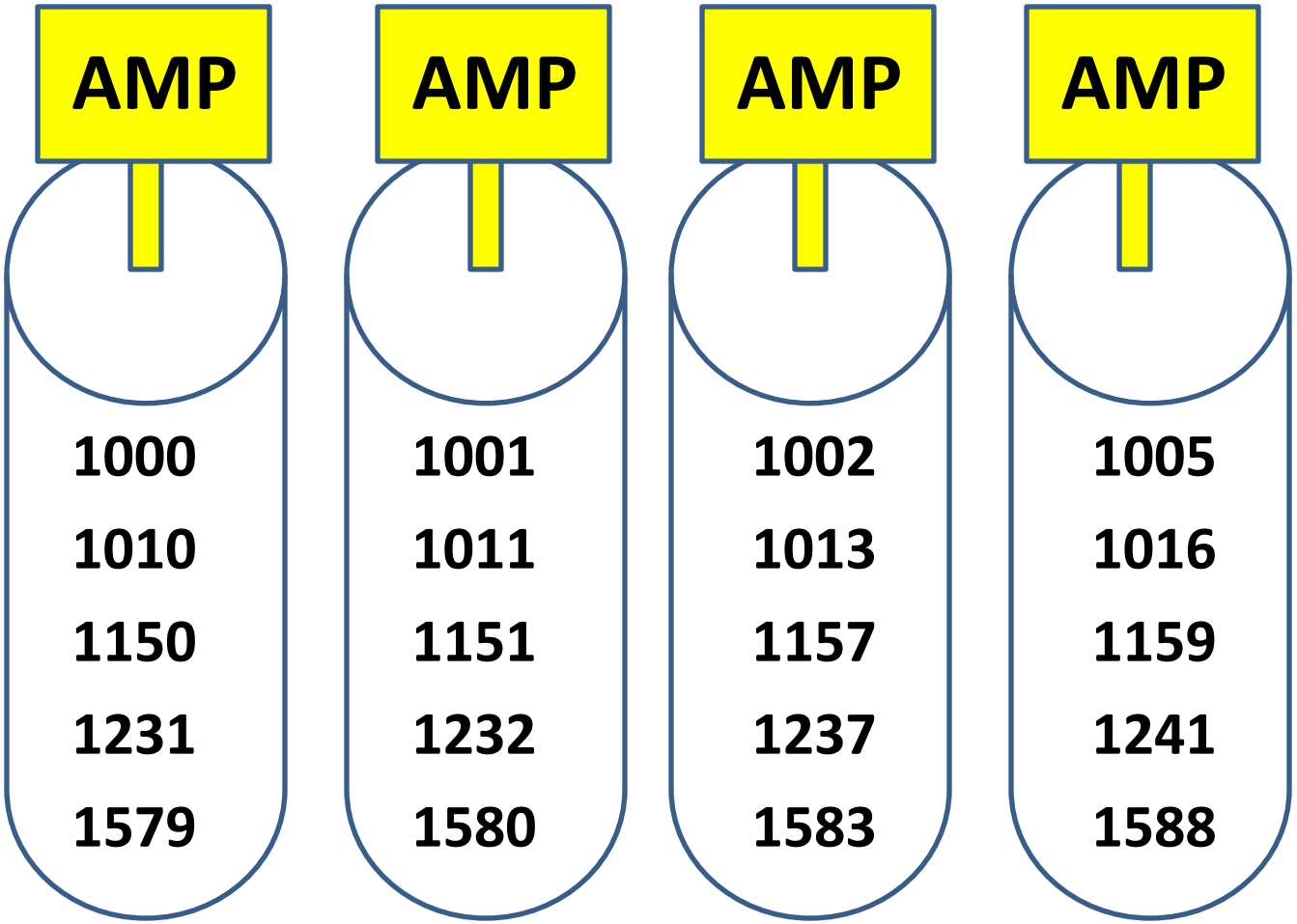
AMP Numbers

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

AMP 2 will hold this row

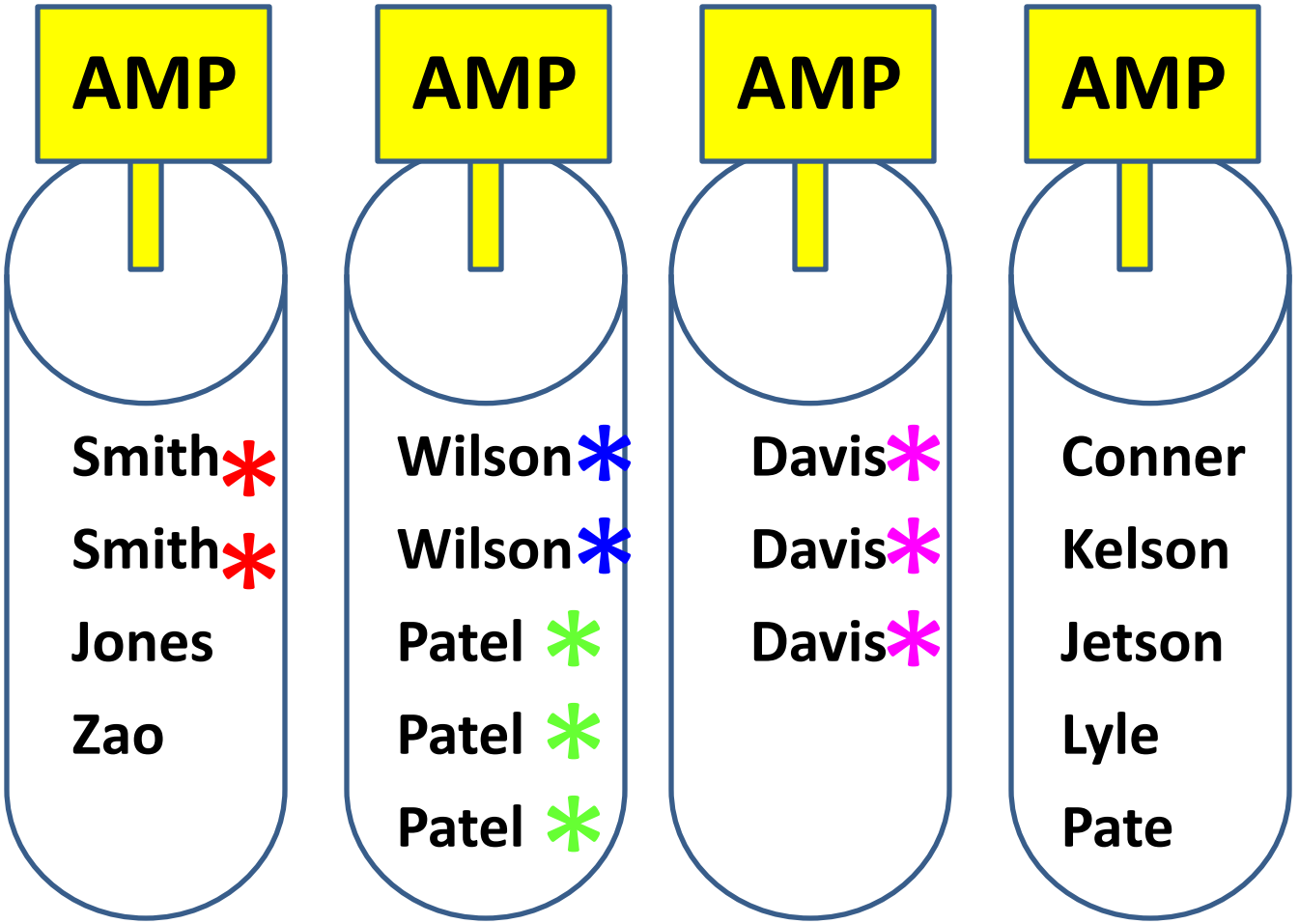


The PE takes the Primary Index of each row and runs it through a math formula and then looks at the hash map to determine which AMP will hold the row.



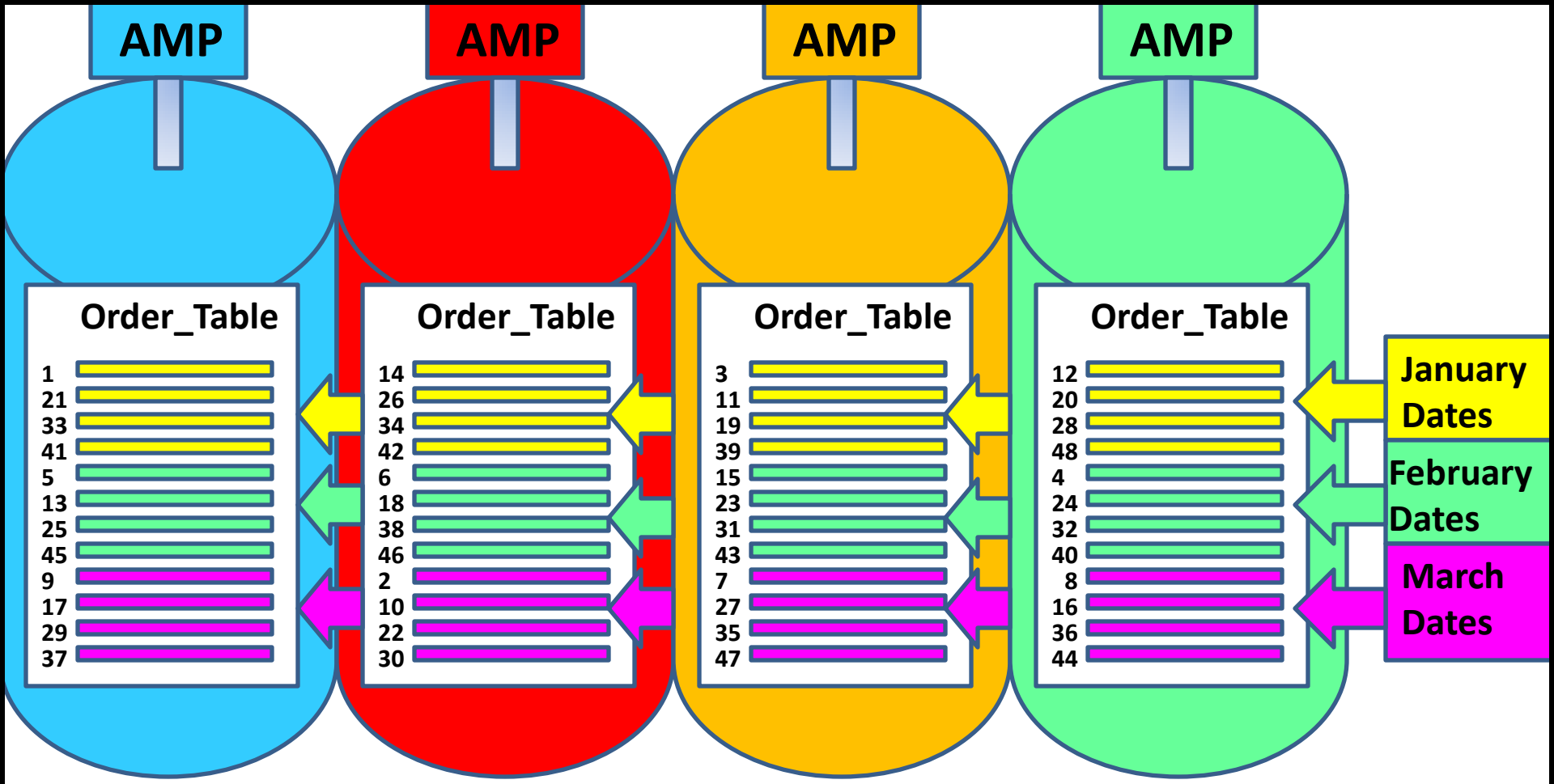
UPI on Emp_No will cause Perfect distribution

A Unique Primary Index (UPI) will spread the rows of a table evenly across the AMPs almost every time.



NUPI on Last_Name will cause uneven distribution

A Non-Unique Primary Index (NUPI) will spread the rows of a table unevenly across the AMPs because duplicate values go to the same AMP every time.



SELECT * FROM ORDER_Table WHERE ORDER_DATE BETWEEN
'2010-01-01' and '2010-01-31';

Teradata even has the ability to Partition Tables for range queries so a Full Table Scan is prevented.

Temporal Tables are tables based on time. There are three types of Temporal Tables.

1) Valid Time Temporal Tables

2) Transaction Time Temporal Tables

3) Bi-Temporal Tables which combine both:

- **Valid Time**
- **Transaction Time**

Below is a normal Employee_Table

Employee_Table

Emp_No	Dept_No	First_Name	Last_Name	Salary
1	100	Mary	Mason	50000.00
2	200	Hitesh	Patel	60000.00
4	400	Chen	Wang	60000.00

Imagine if someone called and said, "Does Hans Svenson work here?"

```
SELECT * FROM Employee_Table
WHERE Last_Name = 'Svenson' ;
```

No Rows Returned

What if they asked, "Did Hans Svenson ever work here?" How would I know I'm not a fortune teller.

Normal Table Vs ValidTime Temporal Table

Employee_Table_Normal

Emp_No	Dept_No	First_Name	Last_Name	Salary
1	100	Mary	Mason	50000.00
2	200	Hitesh	Patel	60000.00
4	400	Chen	Wang	60000.00

Employee_Table_Temporal

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
1	100	Mary	Mason	50000.00	2010-01-01	Forever
2	200	Hitesh	Patel	60000.00	2010-01-02	Forever
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31
4	400	Chen	Wang	50000.00	2010-01-04	2011-01-20
4	400	Chen	Wang	60000.00	2011-01-20	Forever

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
1	100	Mary	Mason	50000.00	2010-01-01	Forever
2	200	Hitesh	Patel	60000.00	2010-01-02	Forever
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31
4	400	Chen	Wang	50000.00	2010-01-04	2011-01-20
4	400	Chen	Wang	60000.00	2011-01-20	Forever

"Does Hans Svenson work here?"

```
SELECT * FROM Employee_Table_Temporal
WHERE Last_Name = 'Svenson' ;
```

No Rows Returned

Keywords placed in front of your standard SQL can Return closed rows.

```
NONSEQUENCED VALIDTIME
SELECT * FROM Employee_Table_Temporal
WHERE Last_Name = 'Svenson' ;
```

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
1	100	Mary	Mason	50000.00	2010-01-01	Forever
2	200	Hitesh	Patel	60000.00	2010-01-02	Forever
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31
4	400	Chen	Wang	50000.00	2010-01-04	2011-01-20
4	400	Chen	Wang	60000.00	2011-01-20	Forever

The two major concepts I want you to understand are:

1. No rows are physically deleted, so the table is APPEND ONLY. Row will be logically deleted.
2. Rows are deleted and updated by logically closing out the row. So rows will be open when data is inserted and then logically closed on a delete. Rows that are open are current rows and can be queried exactly like a normal table, but rows that are deleted logically will be invisible unless the user uses special keywords in their SQL.

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
1	100	Mary	Mason	50000.00	2010-01-01	Forever
2	200	Hitesh	Patel	60000.00	2010-01-02	Forever
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31
4	400	Chen	Wang	50000.00	2010-01-04	2011-01-20
4	400	Chen	Wang	60000.00	2011-01-20	Forever

1) How many employees are currently employed? _____
2) When did Hans Svenson leave the company? _____
3) What is the only column ever updated for a row physically? _____
4) What keyword determines that a row is still open? _____
5) What was the date that Chen Wang received a raise? _____

When Chen Wang received a raise the SQL was written like this:

```
UPDATE Employee_Table_Temporal
SET Salary = 60000
WHERE Emp_No = 4;
```

6) What two things did Teradata automatically do behind the scenes?____ ____

Emp_No	Dept_No	First_Name	Last_Name	Salary	ValidStart	ValidEnd
1	100	Mary	Mason	50000.00	2010-01-01	Forever
2	200	Hitesh	Patel	60000.00	2010-01-02	Forever
3	300	Hans	Svenson	80000.00	2010-01-03	2010-12-31
4	400	Chen	Wang	50000.00	2010-01-04	2011-01-20
4	400	Chen	Wang	60000.00	2011-01-20	Forever

1) How many employees are currently employed? **3**
2) When did Hans Svenson leave the company? **December 31, 2010**
3) What is the only column ever updated for a row physically? **ValidEnd**
4) What keyword determines that a row is still open? **Forever**
5) What was the date that Chen Wang received a raise? **January 20, 2011**

When Chen Wang received a raise the SQL was written like this:

```
UPDATE Employee_Table_Temporal
SET Salary = 60000 WHERE Emp_No = 4;
```

6) What two things did Teradata automatically do behind the scenes? **Updated the ValidEnd column and inserted a new row to reflect the salary change.**

Normal Table Vs TransactionTime Temporal Table

Policy_Table_Normal

Cust_No	Policy_No	Amount
1	10	500,000.00
2	20	600,000.00
3	30	1,000,000.00

Policy_Table_Temporal

Cust_No	Policy_No	Amount	TransactionTime
1	10	500,000.00	2011-01-01 , 9999-12-31
2	20	600,000.00	2011-01-02 , 9999-12-31
3	30	1,000,000.00	2011-01-03 , 9999-12-31

Policy_Table_Normal

Cust_No	Policy_No	Amount
1	10	500,000.00
2	20	600,000.00
3	30	1,000,000.00

```
UPDATE Policy_Table_Normal
SET Insured_Amount = 2000000.00
WHERE Customer_No = 1 ;
```

```
UPDATE Policy_Table_Temporal
SET Insured_Amount = 2000000.00
WHERE Customer_No = 1 ;
```

Policy_Table_Temporal

Cust_No	Policy_No	Amount	TransactionTime
1	10	500,000.00	2011-01-01 , 9999-12-31
2	20	600,000.00	2011-01-02 , 9999-12-31
3	30	1,000,000.00	2011-01-03 , 9999-12-31

How will both tables look like after the UPDATE statement done on December 1, 2011?

Here are both tables after customer 1 was updated to an amount of 2,000,000 on December 1, 2011.

Policy_Table_Normal

Cust_No	Policy_No	Amount
1	10	2,000,000.00
2	20	600,000.00
3	30	1,000,000.00

Policy_Table_Temporal

Cust_No	Policy_No	Amount	TransactionTime
1	10	500,000.00	2011-01-01 , 2011-12-01
2	20	600,000.00	2011-01-02 , 9999-12-31
3	30	1,000,000.00	2011-01-03 , 9999-12-31
1	10	2,000,000.00	2011-12-31 , 9999-12-31

Policy_Table_Temporal

Cust_No	Policy_No	Amount	TransactionTime
1	10	500,000.00	2011-01-01 – 2011-12-01
2	20	600,000.00	2011-01-02 – 9999-12-31
3	30	1,000,000.00	2011-01-03 – 9999-12-31
1	10	2,000,000.00	2011-12-31 – 9999-12-31

```
SELECT * FROM Policy_Table_Temporal ;
```

Cust_No	Policy_No	Amount
1	10	2,000,000.00
2	20	600,000.00
3	30	1,000,000.00

Policy_Table_Temporal

Cust_No	Policy_No	Amount	TransactionTime
1	10	500,000.00	2011-01-01 – 2011-12-01
2	20	600,000.00	2011-01-02 – 9999-12-31
3	30	1,000,000.00	2011-01-03 – 9999-12-31
1	10	2,000,000.00	2011-12-31 – 9999-12-31

TRANSACTIONTIME as of '2011-06-30'

SELECT * FROM Policy_Table_Temporal ;

Cust_No	Policy_No	Amount
1	10	500,000.00
2	20	600,000.00
3	30	1,000,000.00

Temporal Tables are tables based on time. There are Four types of time concepts.

1) Current Time uses normal SQL and the answer set reflects the current rows that are open.

2) AS of Time uses an exact point in time and the answer set reflects the current rows open at that particular point in time.

3) Sequenced Time uses Period of Applicability and the answer set reflects the current rows open during that period of time.

4) Non-Sequenced Time ignores the ValidTime or TransactionTime and the answer set reflects the Rows by ignoring the temporal time.

Let's CREATE a Bi-Temporal Table

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

What is it about this Bi-Temporal table CREATE statement that makes Teradata know the table is Bi-Temporal?

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

What is it about this Bi-Temporal table CREATE statement that makes Teradata know the table is Bi-Temporal?

The Keywords VALIDTIME and TRANSACTTIME

Let's CREATE a Bi-Temporal Table

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

What does a PERIOD Data Type mean?

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

A Period Data Type means a beginning and ending date or Timestamp such as:

2011-01-01 , 9999-12-31 ← Forever

Or

2011-01-01 , 2012-06-30 ← Closed

Or

Timestamp

2011-01-01 08:09:290000-05:00, 9999-12-31 23:59:59.999999+00:00

Timestamp Forever

Let's CREATE a Bi-Temporal Table

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

What type of PERIOD Data Types are required for ValidTime and TransactionTime?

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

What type of PERIOD Data Types are required for ValidTime and TransactionTime?

ValidTime can be either a date or a Timestamp

TransactionTime must be a Timestamp written exactly as above!

On January 1, 2011 Tera-Tom buys property 100 which is beach front property! Tera-Tom is Customer_No 1 in your table and in your heart!

```
CREATE MULTISET TABLE Property_Owners
(
 Cust_No INTEGER
,Prop_No INTEGER
,Prop_Val_Time PERIOD (DATE) NOT NULL as VALIDTIME
,Prop_Tran_Time PERIOD (TIMESTAMP(6) with TIME ZONE)
                                NOT NULL as TRANSACTIONTIME
)
PRIMARY INDEX(Prop_No) ;
```

```
INSERT INTO PROPERTY_OWNERS
(Cust_No, Prop_No)
  VALUES (1, 100) ;
```

```
INSERT INTO PROPERTY_OWNERS
(Cust_No, Prop_No)
VALUES (1, 100) ;
```

Below is what the table looks like internally after Tera-Tom bought the property on January 1, 2011 (2011-01-01)

Property_Owners

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 9999-12-31

Note: The TransactionTime should have a Timestamp, but not shown here for space reasons.

On February 14, 2011 Tera-Tom sells property 100 to Socrates who is Customer_No 2!

```
UPDATE Property_Owners
SET Cust_No = 2
WHERE Prop_No = 100 ;
```

Property_Owners

Table before the UPDATE

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 9999-12-31

What does the table look like after the UPDATE?
How many rows are in the new table?

```
UPDATE Property_Owners
SET Cust_No = 2
WHERE Prop_No = 100 ;
```

Socrates bought on February 14, 2011 (2011-02-14)

Property_Owners

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 9999-12-31

Property_Owners

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 2011-02-14
1	100	2011-01-01 , 2011-02-14	2011-02-14 , 9999-12-31
2	100	2011-02-14 , 9999-12-31	2011-02-14 , 9999-12-31

On April 1, 2011 Socrates sells property 100 but through a different mortgage company so our mortgage company no longer owns property 100!

```
DELETE FROM Property_Owners
WHERE Prop_No = 100 ;
```

Table before the DELETE

Property_Owners

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 2011-02-14
1	100	2011-01-01 , 2011-02-14	2011-02-14 , 9999-12-31
2	100	2011-02-14 , 9999-12-31	2011-02-14 , 9999-12-31

What does the table look like after the DELETE?
How many rows are in the table now?

DELETE FROM Property_Owners WHERE Prop_No = 100 ;

Property_Owners

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 2011-02-14
1	100	2011-01-01 , 2011-02-14	2011-02-14 , 9999-12-31
2	100	2011-02-14 , 9999-12-31	2011-02-14 , 9999-12-31

What does the table look like after the DELETE that happened on April 1, 2011 (2011-04-01)?

Cust_No	Prop_No	ValidTime	TransactionTime
1	100	2011-01-01 , 9999-12-31	2011-01-01 , 2011-02-14
1	100	2011-01-01 , 2011-02-14	2011-02-14 , 9999-12-31
2	100	2011-02-14 , 9999-12-31	2011-02-14 , 2011-04-01
2	100	2011-02-14 , 2011-04-01	2011-04-01 , 9999-12-31

The End

Thank You

Teradata and Temporal Tables for DAMA

Tom Coffing (Tera-Tom)
CEO, Coffing Data Warehousing
Tom.Coffing@CoffingDW.Com
Direct: 513 300-0341
Website: www.CoffingDW.com