Pivotal

A NEW PLATFORM FOR A NEW ERA

GPDB Table Partitioning

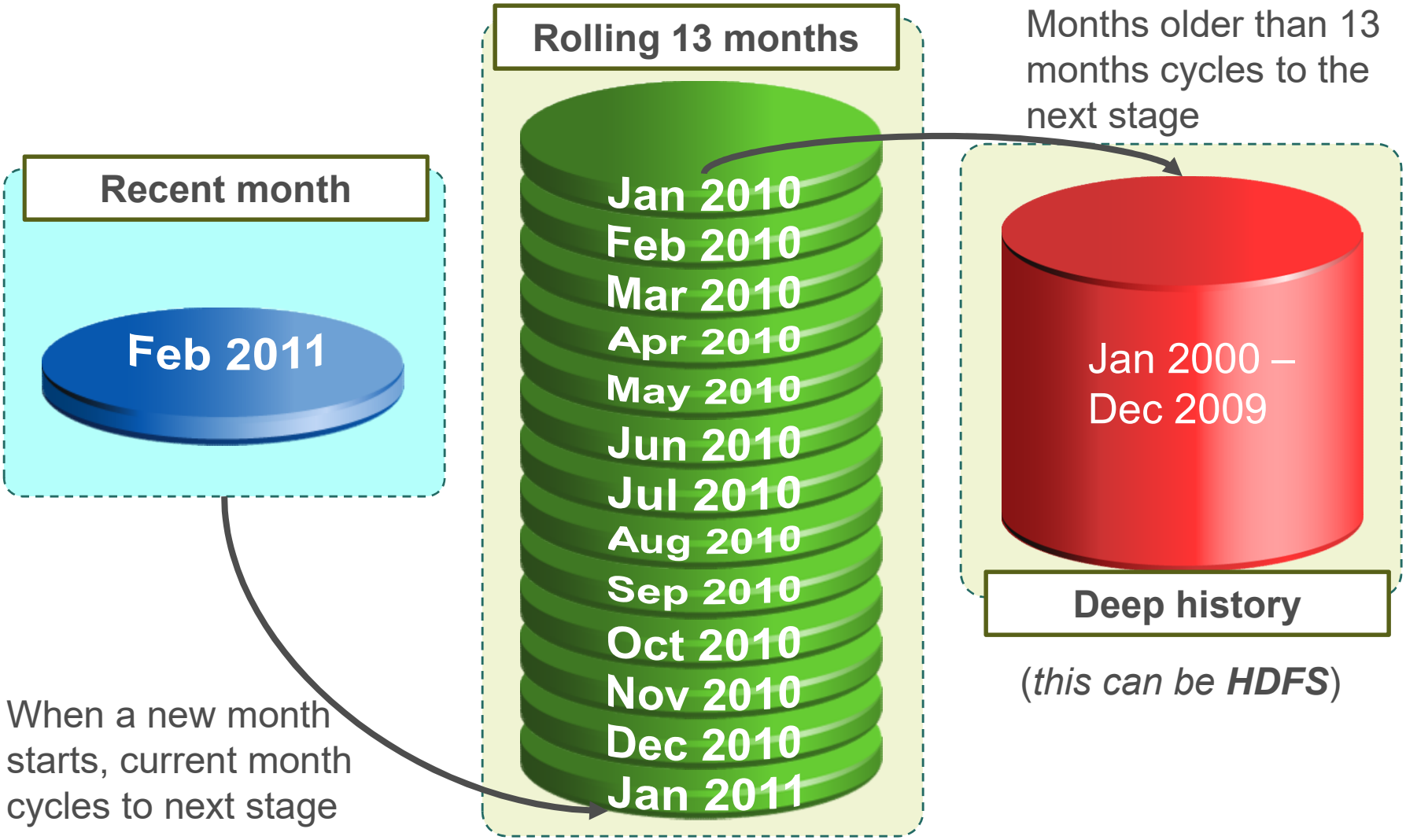
Pivotal® Greenplum Database

Agenda

- Introduction
- What is table partitioning?
- Two methods of partitioning
- Why partition?
- Steps to partition a table
- Test it out in the lab

Pivotal™

Historical Data Management



Partitions in Greenplum

- Are a mechanism in Greenplum for use in physical database design
- Increase the available options to improve the performance of a certain class of queries
- Propagate data to the child tables

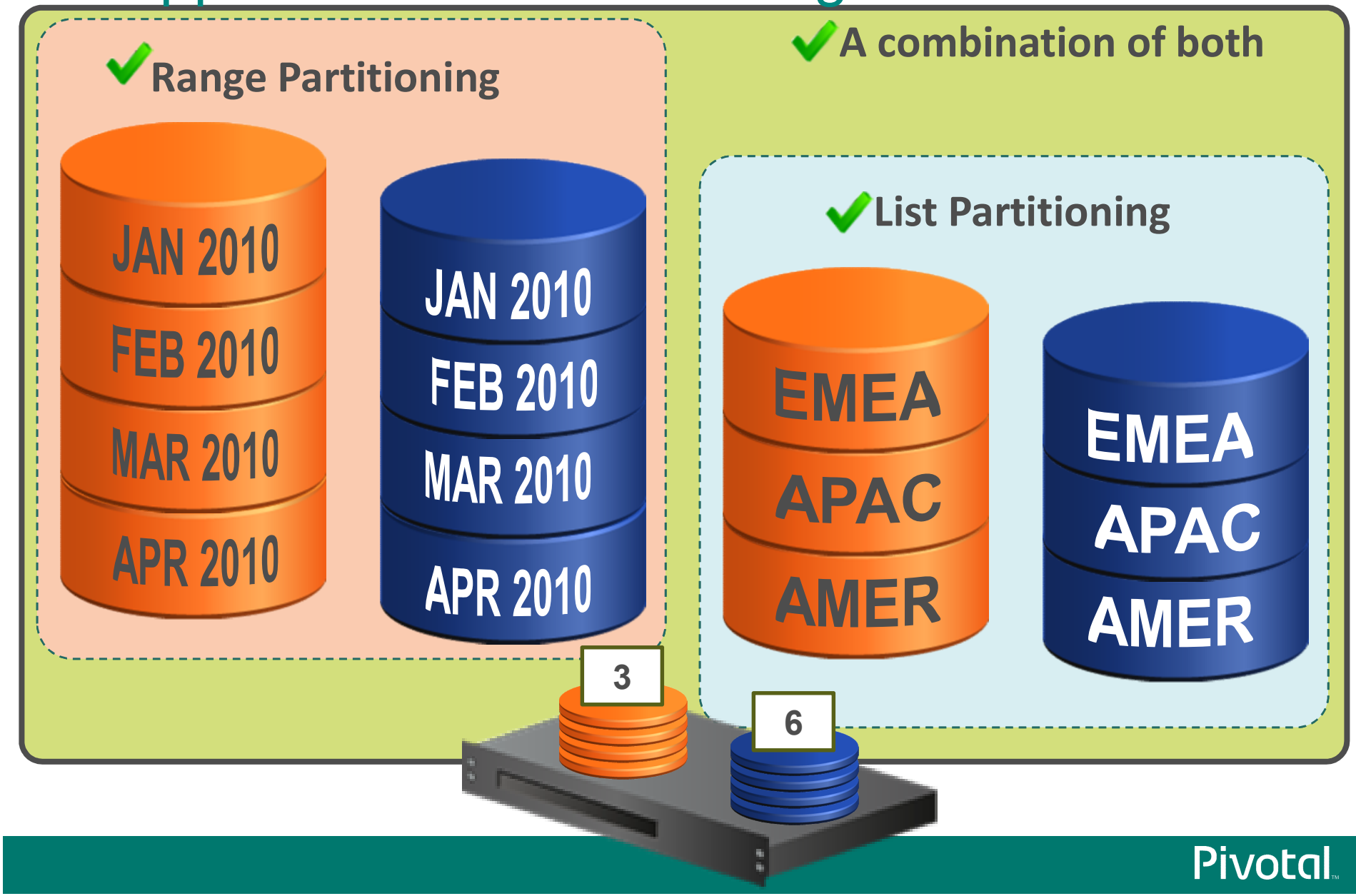
Pivotal™

Table Partitioning Overview

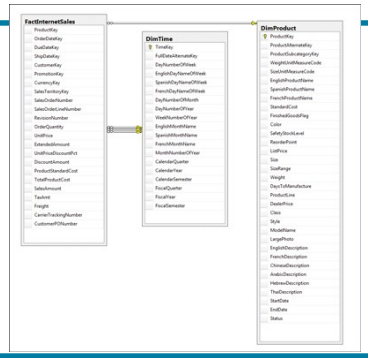
- Address challenges in supporting very large tables
- Divide data into smaller, more manageable pieces
- Can improve query performance
- Can facilitate database maintenance tasks
- Utilizes inheritance and constraints
- Distributes data across segments (as usual)

Pivotal™

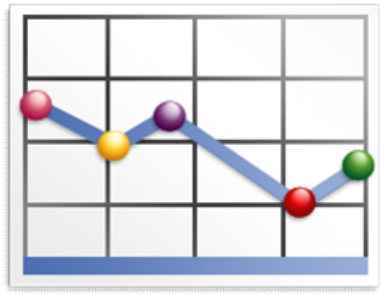
Supported Table Partitioning Methods



When Do You Partition?

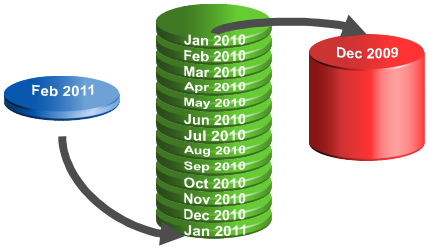


When you have a large fact table

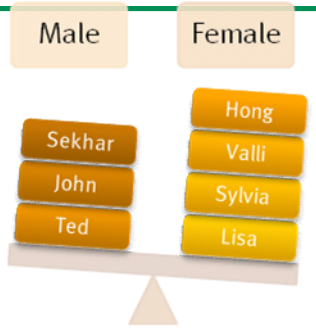


Experiencing unsatisfactory performance

You have identifiable access patterns

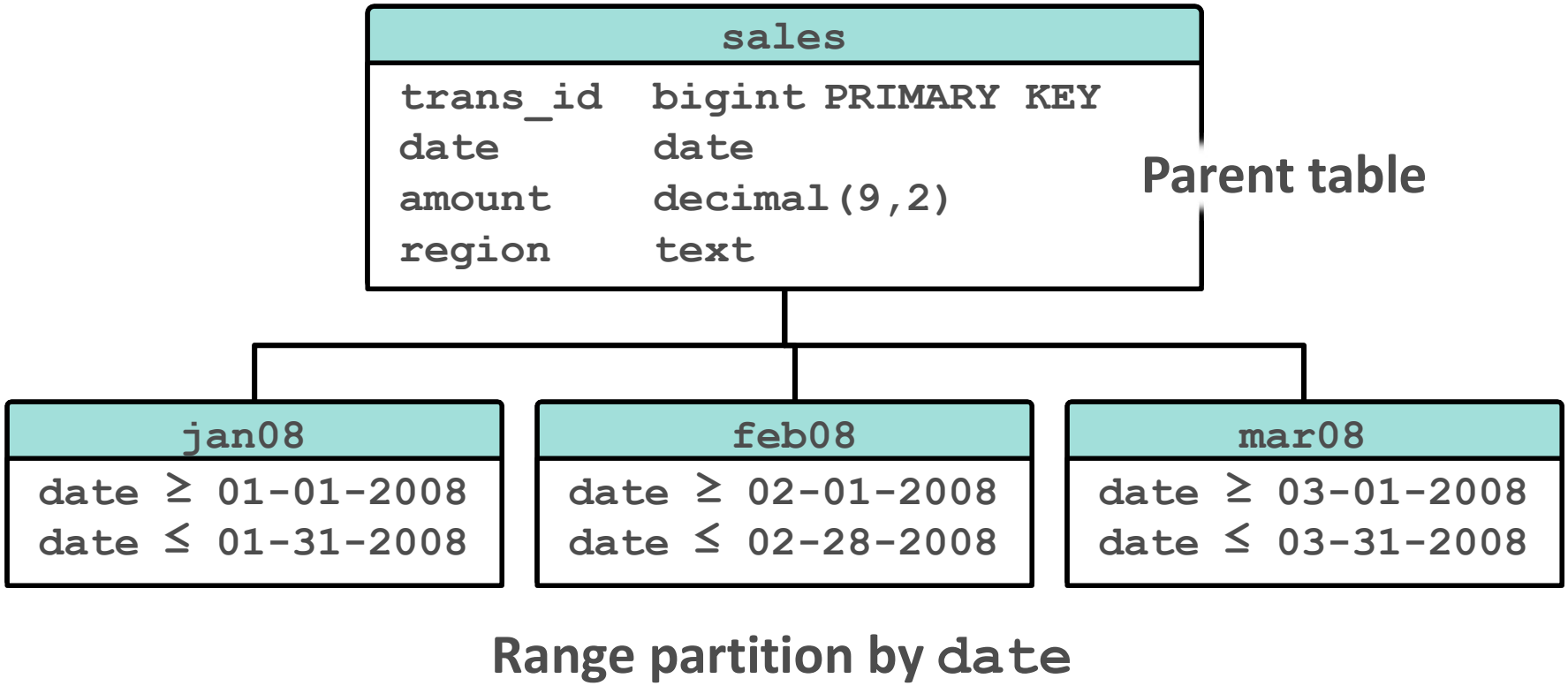


You need to maintain rolling data



Data can be divided into equal parts based on a column

Partitioned Table Design



Creating Partitioned Tables

A table …

- Can be partitioned only at creation time
- Can be partitioned with the command, `CREATE TABLE`
- Can be subpartitioned into additional levels
- Can be partitioned using the following partition designs:
  - Date range
  - Numeric range
  - List

Pivotal™

Creating a Range Partitioned Table

Example: Creating a partitioned table with date range table partitions

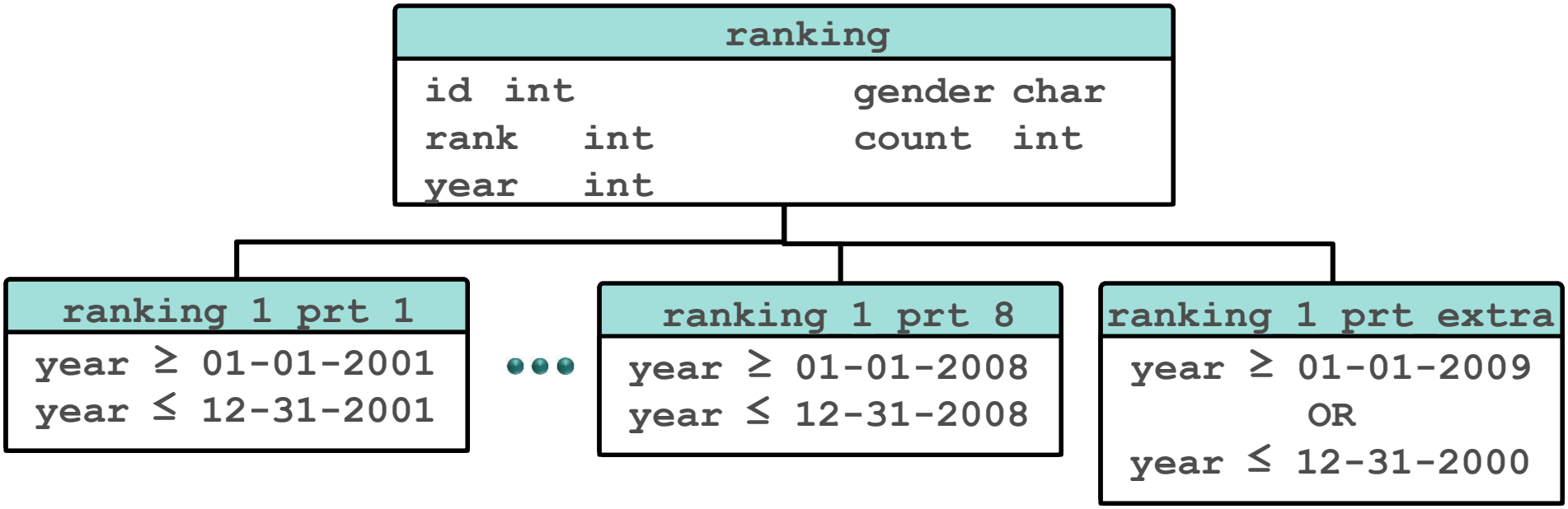
```
CREATE TABLE sales (id int, date date, amt decimal(10,2))
DISTRIBUTED BY (id)
PARTITION BY RANGE (date)
( PARTITION Jan08 START (date '2008-01-01') INCLUSIVE ,
  PARTITION Feb08 START (date '2008-02-01') INCLUSIVE ,
  PARTITION Mar08 START (date '2008-03-01') INCLUSIVE ,
  …
  PARTITION Dec08 START (date '2008-12-01') INCLUSIVE
  END (date '2009-01-01') EXCLUSIVE );
```

Creating a Range Partitioned Table (Cont'd.)

Example: Creating a partition table with numeric range table partitions

```
CREATE TABLE ranking (id int, rank int, year int, gender
char(1), count int)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
( START (2001) END (2008) EVERY (1),
  DEFAULT PARTITION extra );
```

Creating a Range Partitioned Table (Cont'd.)

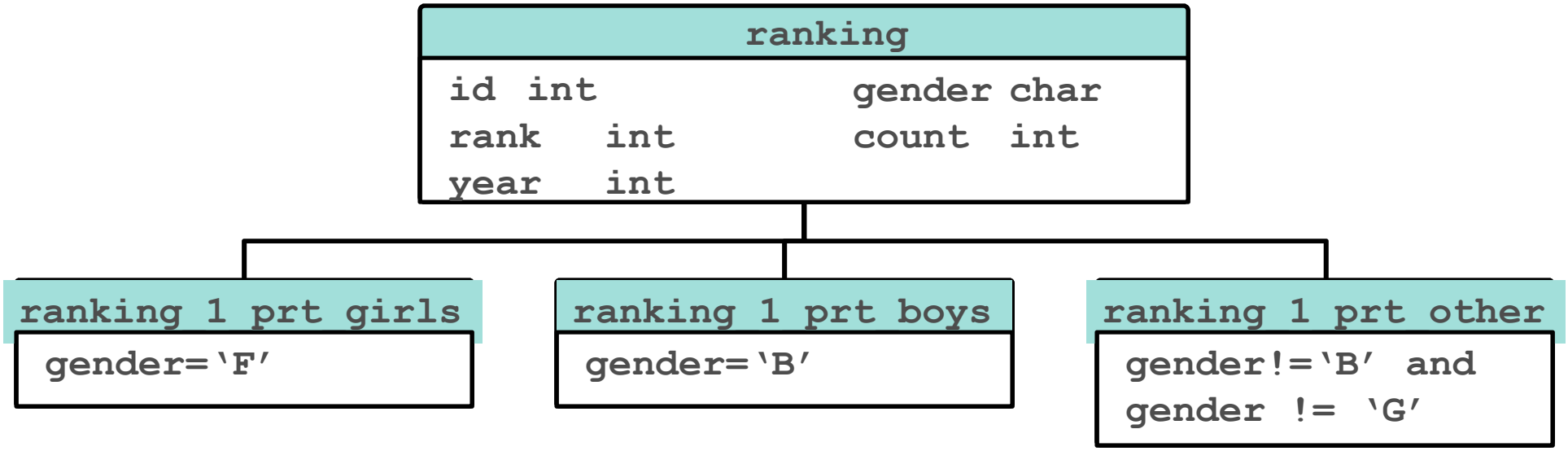


Creating a List Partitioned Table

Example: Creating a partition table with list

```
CREATE TABLE ranking (id int, rank int, year int, gender
char(1), count int )
DISTRIBUTED BY (id)
PARTITION BY LIST (gender)
( PARTITION girls VALUES ('F'),
  PARTITION boys VALUES ('M'),
  DEFAULT PARTITION other );
```

Creating a List Partitioned Table (Cont'd.)



Partitioning an Existing Table

When working with an existing table:

- You cannot partition the table once it's been created
- You can create a new partitioned table and migrate your data

Pivotal™

Partitioning an Existing Table (Cont'd.)

Example: How to partition an existing table

```
CREATE TABLE sales2 (LIKE sales)
PARTITION BY RANGE (date)
( START (date '2008-01-01') INCLUSIVE
END (date '2009-01-01') EXCLUSIVE
EVERY (INTERVAL '1 month') );

INSERT INTO sales2 SELECT * FROM sales;
DROP TABLE sales;
ALTER TABLE sales2 RENAME TO sales;
GRANT ALL PRIVILEGES ON sales TO admin;
GRANT SELECT ON sales TO guest;
```

Pivotal™

Maintaining Partitioned Tables

Use `ALTER TABLE` to:

Example: Add a partition to an existing partitioned table

```
ALTER TABLE sales ADD PARTITION
START (date '2009-02-01') INCLUSIVE
END (date '2009-03-01') EXCLUSIVE;
```

Example: Rename a partition

```
ALTER TABLE sales RENAME TO globalsales;
ALTER TABLE sales RENAME PARTITION FOR ('2008-01-01') TO
jan08;
```

Example: Add a default partition

```
ALTER TABLE sales ADD DEFAULT PARTITION other;
```

Maintaining Partitioned Tables (Cont'd.)

Example: Remove a partition

```
ALTER TABLE sales DROP PARTITION FOR (RANK(1));
```

Example: Truncate a partition

```
ALTER TABLE sales TRUNCATE PARTITION FOR (RANK(1));
```

Example: Exchange a partition

```
CREATE TABLE jan08 (LIKE sales) WITH (appendonly=true);
INSERT INTO jan08 SELECT * FROM sales_1_prt_1 ;
ALTER TABLE sales EXCHANGE PARTITION FOR ('2008-01-01') WITH
TABLE jan08;
```

Example: Split an existing partition in a partitioned table

```
ALTER TABLE sales SPLIT PARTITION FOR ('2008-01-01')
AT ('2008-01-16')
INTO (PARTITION jan081to15, PARTITION jan0816to31);
```

Pivotal™

Viewing Details on Partitioned Tables

gpadmin@mdw:~

```
names=# \d+ ranking
                    Table "baby.ranking"
 Column |      Type      | Modifiers | Storage  | Description
--------+----------------+-----------+----------+-------------
 id     | integer        |           | plain    |
 rank   | integer        |           | plain    |
 year   | integer        |           | plain    |
 gender | character(1)   |           | extended |
 count  | integer        |           | plain    |
Child tables: ranking_1_prt_boys,
              ranking_1_prt_girls,
              ranking_1_prt_other
Has OIDs: no
Distributed by: (id)

names=#
```

Details on the parent and child tables

`pg_partitions` view lets you obtain more details on partitioned tables

gpadmin@mdw:~

```
names=# select partitionboundary, partitiontablename, partitionname,partitiontype,partitionlistvalues
FROM pg_partitions
WHERE tablename=
'ranking';
      partitionboundary       | partitiontablename  | partitionname | partitiontype | partitionlistvalues
------------------------------+---------------------+---------------+---------------+---------------------
 PARTITION girls VALUES('F')  | ranking_1_prt_girls | girls         | list          | 'F'::bpchar
 PARTITION boys VALUES('M')   | ranking_1_prt_boys  | boys          | list          | 'M'::bpchar
 DEFAULT PARTITION other      | ranking_1_prt_other | other         | list          |
(3 rows)

names=#
```

`pg_partition_columns` displays the column used for partitioning

gpadmin@mdw:~

```
names=# select * from pg_partition_columns;
 schemaname | tablename | columnname | partitionlevel | position_in_partition_key
------------+-----------+------------+----------------+--------------------------
 baby       | ranking   | gender     |              0 |                         1
(1 row)

names=#
```

Pivotal

Using PQO With Partitioned Tables

- Pivotal Query Optimizer requires statistics on the root partition of partitioned tables
- `gpconfig -c optimizer_analyze_root_partition -v on –masteronly`
- *OR*
- `ANALYZE ROOTPARTITION;`

Pivotal™

Review

- What is table partitioning?
- Two methods of partitioning
- Why partition?
- Steps to partition a table
- Test it out in the lab

Pivotal™

Pivotal

A NEW PLATFORM FOR A NEW ERA