Pivotal

A NEW PLATFORM FOR A NEW ERA

Data Manipulation Language (DML) in GPDB

Pivotal® Greenplum Database

Agenda

- Introduction
- SQL support in GPDB
- Built-in operators and functions
- Transactions

Pivotal™

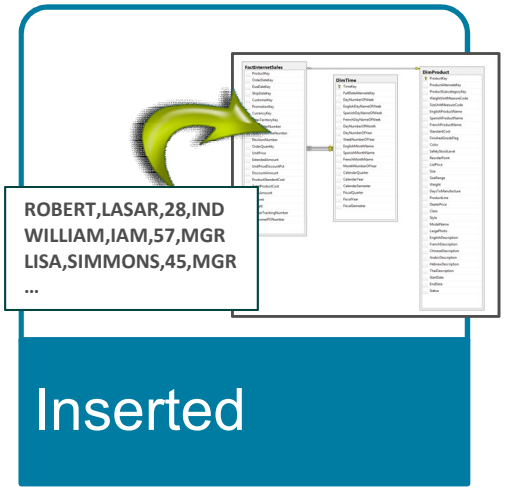
SQL Support in Greenplum Database

- Greenplum Database is almost fully compliant with the SQL 1992 standard
- GPDB supports most of the features from SQL 1999
- Several features from SQL 2003 have also been implemented (e.g. SQL OLAP features)
- A detailed treatment of GPDB's SQL conformance can be found here (and within links found there): http://gpdb.docs.pivotal.io/4330/ref_guide/feature_summary.html

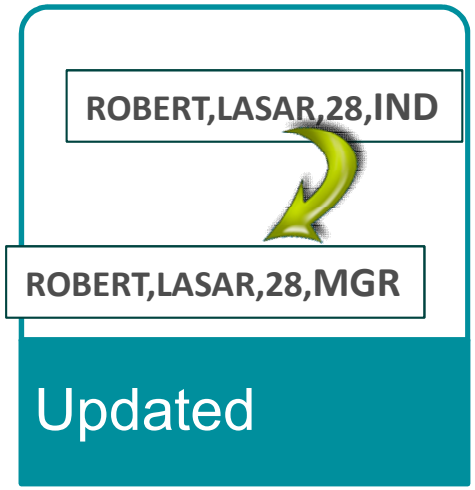
Pivotal™

Managing Data

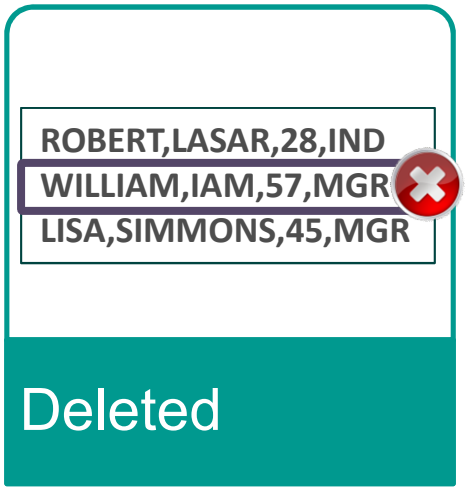
When working with data, data can be:



Inserted



Updated



Deleted

Inserting Data

`INSERT` command:

- Is fully supported
- Single row INSERTs are fine for small numbers of rows
- Can be substituted with:

```
INSERT INTO <table>
 SELECT FROM <external table>
```

or

```
COPY time_dim FROM stdin DELIMITER ','
NULL '' HEADER;
```

The following is an example of its use:

```
INSERT INTO names VALUES
   (nextval('names_seq'), 'test', 'U');
```

Updating Rows

The `UPDATE` command:

- Is used to update individual, multiple, or all rows in a table
- There are two ways to modify a table using information contained in other tables in the database: using sub-selects, or specifying additional tables in the FROM clause. Which technique is more appropriate depends on the specific circumstances.
- Distribution key columns to be updated only if using Pivotal Query Optimizer (PQO)

Simple UPDATE example:

```
UPDATE names SET name='Emily' WHERE name='Emmmily';
```

Pivotal™

Removing Data

The following commands are used to remove data:

- `DELETE FROM t WHERE … [some predicate]`
- `TRUNCATE TABLE t`
- `DROP TABLE t`
- There are two ways to delete rows in a table using information contained in other tables in the database: using sub-selects, or specifying additional tables in the `USING` clause. Which technique is more appropriate depends on the situation.

```
DELETE FROM ranking; (deletes all rows)
DELETE FROM ranking WHERE year='2001';
```

Pivotal™

Correlated Subqueries

- A query that is nested inside an outer query block and references values in the outer query

Example:

```
SELECT * FROM part p1
WHERE price > (
   SELECT avg(price)
   FROM part p2
   WHERE  p2.brand = p1.brand
)
```

Common Table Expressions

- Think of it as a temp table used within a single query
- Purpose is to avoid re-execution of expressions referenced more than once within a query
- Also known as “CTE” or the `WITH` clause

```
WITH v as (SELECT i_brand, i_current_price, max(i_units) m
           FROM item
           WHERE i_color = 'red'
           GROUP BY i_brand, i_current_price)
SELECT * FROM v WHERE m < 100
AND v.i_current_price IN (SELECT min(i_current_price)
                          FROM v WHERE m > 5);
```

Built-in Functions and Operators

- GPDB Supports a rich set of functions and operators for the built-in data types
- Date/time, mathematical, string, aggregate functions and operators
- The following attributes inform the query optimizer about the behavior of the function:
  - `IMMUTABLE`
  - `STABLE`
  - `VOLATILE`

Pivotal™

IMMUTABLE

- Does not modify the database
- When called with a given set of arguments, will always return the same value
- Does not do database lookups or use information not provided in its argument list
- Any call to such a function with constant arguments can be replaced by the function's return value

Pivotal™

STABLE

- Cannot modify the database
- Within single table scan, consistently returns same result for a given set of argument values
- However, the return value could change across SQL statements

Pivotal™

VOLATILE

- Function value can change even within a single table scan
- No optimizations can be made
- May have side effects
- Examples: random(), timeofday(), currval()

Pivotal™

Built-in Functions

Function	Description	Example
`RANDOM()`	Returns pseudo-random value on [0.0 1.0)	0.108874355442822
`SUBSTRING(str FROM x FOR n)`	n chars of str starting at offset ‘x’ (indexes start at 1)	(see demo)
`CURRENT_TIMESTAMP`	Returns the current system date and time	2008-01-06 16:51:44.430000+00:00
`DATE_TRUNC(lim, ts)`	Truncate the timestamp, ts, to lim (‘month’, ‘day’, ‘week’, etc.)	(see demo)
`LENGTH(‘1234’)`	Returns length of string	4
`CURRENT_ROLE ROLE`	Returns the current database user	jdoe

Pivotal™

Comparison Operators

Operator	Description
=	Equal to
!= OR <>	NOT Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
x BETWEEN y AND z	Short hand for x >= y and x <= z
x IS NULL	True if x has NO VALUE
‘abc’ LIKE ‘%abcde%’	Pattern Matching

Pivotal™

Mathematical Operators

Function	Returns	Description	Example	Results
+ − * /	same	Add, Subtract, Multiply & Divide	1 + 1	2
%	Integer	Modulo	10%2	0
^	Same	Exponentiation	2^2	4
|/	Numeric	Square Root	|/9	3
||/	Numeric	Cube Root	||/8	2
!	Numeric	Factorial	!3	6
& | # ~	Numeric	Bitwise And, Or, XOR, Not	91 & 15	11
<< >>	Numeric	Bitwise Shift left, right	1 << 4 8 >> 2	16 2

Pivotal™

Aggregate Functions

Function	Returns	Description
sum	bigint for smallint or int arguments, numeric for bigint arguments, double precision for floating-point arguments, otherwise the same as the argument data type	Sum of *expression* across all input values
count	bigint	Number of input rows for which the value of *expression* is not null
avg	numeric for any integer type argument, double precision for a floating-point argument, otherwise the same as the argument data type	the average (arithmetic mean) of all input values
min	same as argument type	Minimum value of *expression* across all input values
max	same as argument type	Maximum value of *expression* across all input values

Pivotal™

What Is NULL?

- Represents the absence of value
- A place holder indicating that no value is present
- Literal: NULL
- Query predicates can incorporate NULL:
  - `WHERE ... IS [NOT] NULL`

Pivotal™

Concurrency Control and Multi-version Concurrency Control Features

Data consistency:

- Is maintained by using the MVCC model (Multi-version Concurrency Control).
- Lets each transaction see a snapshot of data
- Protects the user from viewing inconsistent data that could be caused by other transactions executing concurrent updates on the same data rows

MVCC:

- Provides transaction isolation for each database session.
- Uses locking methodologies to minimize lock contention
- Ensures reading never blocks writing and writing never blocks reading

Pivotal™

Transactions

- Bundle multiple statements into one *all-or-nothing* operation
- Are managed with the following commands:

Action	SQL Syntax
Start a transaction block	`BEGIN` or `START TRANSACTION`
Commit the results of a transaction	`END` or `COMMIT`
Abandon the transaction	`ROLLBACK`
Create a savepoint	`SAVEPOINT`

Autocommit mode:

- Is enabled by default in `psql`
- Can be turned off with `\set autocommit on|off`

Pivotal™

Transaction Concurrency Control

Greenplum supports all transaction isolation levels, including:

- `READ COMMITTED` / `READ UNCOMMITTED`
- `SERIALIZABLE` / `REPEATABLE READ`

In Greenplum:

- `INSERT` / `COPY` acquire locks at the row-level
- `UPDATE` / `DELETE` acquire locks at the table-level
- You can use the `LOCK` command to acquire specific locks
- The `LOCK` command must be used within a transaction block

Pivotal™

Checking for Lock Conflicts

Lock conflicts can be:

- Verified by querying `pg_locks`
- Resolved by an administrator
- Caused by:
  - Concurrent transactions accessing the same object
  - Resource queue locks
  - Transaction deadlocks between segments (rare)

Pivotal™

Review

- SQL support in GPDB
- Built-in operators and functions
- Transactions

- Apply these concepts in the lab

Pivotal™

Pivotal

A NEW PLATFORM FOR A NEW ERA