Introduction to PostgreSQL

The Open Source Object-Relational Database Management System

Varlena, LLC

A. Elein Mustain

www.varlena.com
elein@varlena.com

(c) A. Elein Mustain, Varlena. LLC
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution in source or binary must maintain copyright and following disclaimer
- Neither the name of the organization nor the names of its contributors may be used to endorse or promote products.
Agenda

- PostgreSQL Features
- Installation and Configuration
- Maintenance and Monitoring
- Command Line Interface
- Database Basics in PostgreSQL
Not the Agenda

- Client Interfaces
- Inheritance
- Comparisons to other Databases
- Replication, Point in Time Recovery
- Full Text Search
History of Postgres

1986: UCB

1992: Miró/Illustra

1995: postgres95

1996: PostgreSQL Global Dev

Informix

Informix IUS 9

IBM

2000

2001

2006
What is PostgreSQL?

- Relational Database Management System
- Object-Relational Database

- Ability to add First Class simple and complex objects, with methods, that can be used *in a Relational Context (SQL)*
PostgreSQL Relational Features

- Foreign keys
- Triggers
- Views
- Transactional Integrity
  - ACID compliance
- Complex Queries
Data Centricity

- Data stands on its own
  - Data is money
  - Many applications one database
- Database centric logic
  - Integrity cannot be circumvented by applications
ACID Compliance

- Atomic
  - transactions seen in full or not at all

- Consistent
  - system enforced constraints

- Isolated
  - transactions do not interfere with each other transactions

- Durable
  - On Commit, result will not be lost
Multi-Version Concurrency Control

- Snapshot of data for command or transaction
- Virtually eliminates need for locking
- Reading does not block writing and vice versa

SET TRANSACTION ISOLATION LEVEL

READ COMMITTED

SERIALIZABLE
SQL and PostgreSQL

- Excellent Standards Compliance
  - SQL89, SQL92, SQL98, SQL2003
- Documentation includes Compliance
- Design Issues decided by Standards
Object Relational Features

- Data types
- Functions
- Operators
- Rules
- Aggregates
- Index Methods
select hotel_name, hotel_address
from hotels h, airports a
where a.name = 'OAK' and
    h.loc @ Circle(a.loc, '5 miles');

select name, num_kids from people;

select pdf( doc, '/home/me')
from doc d
where dnameget(doc) = 'myresume';

PostgreSQL Queries with Objects
Client GUI Interfaces

- PgAdmin III
  - www.pgadmin.org
- phppgadmin
  - phppgadmin.sourceforge.net
- DbVisualizer
  - www.minq.se/products/dbvis/
- Others, e.g. pgaccess
  - See sourceforge.net
Client Programming Interfaces

- psql - Command Line
- libpq – C library
- ECPG – Embedded SQL
- pgtcl – Tcl binding library

Drivers
- JDBC
- ODBC
- DBI: Perl, Python, PHP, etc.
- .NET
Server Side Languages

- PL/pgsql
- SQL
- C
- Other server side languages
  - PL/perl, PL/python
  - PL/R, PL/Tcl, PL/Ruby
  - PL/bash, PL/Java
  - etc.
Downloading PostgreSQL

http://www.postgresql.org

- By Source: ftp, bittorrent
- By CVS tree
- In Packages: RPM, Debian
- Company Distributions
Operating System Distributions

- Most Linux like OS distributions
- MacOSX:
  - www.entropy.ch/software/macosx/postgresql
- 8.1 Native Win32 Version
  - pginstaller at pgfoundry.org
- Cygwin:
  - www.cygwin.com
Configuration Points

- **Build Time**
  - Build directives
  - Installation directory
  - PL Language options

- **Server Environment**
  - postgresql.conf, pg_hba.conf

- **Runtime/Client Environment**
  - PG environment variables
Configuration Points
Build Time

As user postgres ...

$ ./configure \n   --prefix=/local/pgsql81 \n   --with-perl \n   --with-python \n   --with-tcl \n   --enable-depend
$ make
$ sudo make install
Initdb -D $PGDATA

Creates Data Directory with:

- configuration files
  - postgresql.conf
  - pg_hba.conf
- template databases
  - template0
  - template1
- super user database
Configuration Points
Server Environment

- Global User Configuration
  - $PGDATA/postgresql.conf
  - Environment variables for server startup
- Access Security
  - $PGDATA/pg_hba.conf
  - Host, user and database access.
Configuration Points
Global User Configuration

- Environment Variables for Server Startup
- postgresql.conf
- See also:
  - www.varlena.com/GeneralBits/Tidbits/#Performance
## Configuration Points

### Global User Configuration

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
<th>@ 2G RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>1000</td>
<td>25000</td>
</tr>
<tr>
<td>work_mem</td>
<td>1024</td>
<td>16384</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>16384</td>
<td>16384</td>
</tr>
<tr>
<td>max_fsm_pages</td>
<td>20000</td>
<td>*</td>
</tr>
<tr>
<td>max_fsm_relations</td>
<td>1000</td>
<td>*</td>
</tr>
<tr>
<td>effective_cache_size</td>
<td>1000</td>
<td>82500</td>
</tr>
<tr>
<td>log_destination</td>
<td>stderr</td>
<td>stderr</td>
</tr>
<tr>
<td>redirect_stderr</td>
<td>off</td>
<td>on</td>
</tr>
</tbody>
</table>
## Configuration Points

### Global User Configuration

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
<th>@ 2G RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_directory</td>
<td>pg_log</td>
<td>/var/log/pgsql</td>
</tr>
<tr>
<td>log_min_duration_statement</td>
<td>-1</td>
<td>500</td>
</tr>
<tr>
<td>log_line_prefix</td>
<td>[%p - %t ]</td>
<td></td>
</tr>
<tr>
<td>log_statement</td>
<td>none</td>
<td>ddl</td>
</tr>
<tr>
<td>stats_start_collector</td>
<td>on</td>
<td>on</td>
</tr>
<tr>
<td>stats_command_string</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>stats_block_level</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>stats_row_level</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>autovacuum</td>
<td>off</td>
<td>on</td>
</tr>
<tr>
<td>Host</td>
<td>DB</td>
<td>USER</td>
</tr>
<tr>
<td>--------</td>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td>local</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td># IPv4 local:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td># IPv6 local:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>all</td>
<td>all</td>
</tr>
<tr>
<td># bad bernie</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>all</td>
<td>bernie</td>
</tr>
<tr>
<td># demo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>demo</td>
<td>varlena</td>
</tr>
<tr>
<td># users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>all</td>
<td>all</td>
</tr>
</tbody>
</table>
Configuration Points
Runtime/Client Environment

- Environment Variables
  - PGHOST – default localhost
  - PGPORT – default 5432
  - PGUSER – default $USER
  - PGDATABASE – default $PGUSER

- Different for multiple installations
Configuration Points
Session Setting

- View: pg_settings
- Show values and descriptions
  
  ```sql
  SELECT name, setting, short_desc
  FROM pg_settings
  ORDER BY name;
  ```
- What can be set in a session?
  
  ```sql
  SELECT name
  FROM pg_settings
  where context='user';
  ```
Housekeeping PostgreSQL Start and Stop

- Starting & Stopping PostgreSQL
  - Installation Specific Script (/etc/init.d)
    $ pg_ctl start -D $PGDATA
    $ pg_ctl stop
  - Windows PostgreSQL--> Programs
Housekeeping PostgreSQL Logging

- Log Maintenance
  - Rotate Log Settings in `postgresql.conf`
  - Alternative:
    
    ```
    $ pg_ctl start -D $PGDATA \rotatelogs $PGDATA/pglog 86400 2>&1;
    ```

- Always know where your log file is!
Housekeeping PostgreSQL Vacuuming

- Autovacuum
  - Configure in postgresql.conf
- Vacuum
  - $ vacuumdb --analyze --full
- Updates Statistics
  - Improves Performance
- Recovers Disk Space
- Frequency tuning required
Housekeeping PostgreSQL Backing Up

- **Backup**
  
  `pg_dumpall > \`
  `.../`date +%Y%m%d`dump.sql`

- **Restore**

  `psql -f 20061231dump.sql`

**Backup! Now!**

**No excuses! Really!**
Monitoring PostgreSQL

Client Server Architecture

- `pg_stat_activity`
  - Set `pg_stats_command` in `postgresql.conf`
- `ps -alx`
- Log files
  - check `pgfoundry` for log parsers
Documentation and Help

- Online & Downloadable Docs
- Mailing Lists: www.postgresql.org
- IRC #postgresql freenode.net
- PostgreSQL General Bits :-)
  - http://www.varlena.com/GeneralBits
Creating Databases

$ createdb accounts
Adding Users

$ createuser bob

Shall the new role be a superuser? (y/n) n

Shall the new role be allowed to create databases? (y/n) y

Shall the new role be allowed to create more new roles? (y/n) y

CREATE ROLE
psql Basics

Always learn help first.

- Command Line options
  
  $ psql --help

- Backslash Command Help
  
  $ psql db
db=# \?

- SQL Help
  
  $ psql db
db=# \help [SQL command]
Database Design Elements

- Data Types & Sequences
- Nulls
- Keys
- Constraints & Defaults
- Triggers, Functions & Operators
- Tablespaces
- Simple domains
- Rules
Create Table

- AS, LIKE
- WITH OIDS
  - Current default WITH may change
  - See default_with_oids
- Temporary Tables
  - PRESERVE ROWS, DELETE ROWS, DROP
- INHERITS
- CONSTRAINTS
- TABLESPACE
CREATE TABLE people ( 
id SERIAL PRIMARY KEY, 
name text, 
dept_no int REFERENCES dept(dept_no) 
); 

CREATE temp TABLE ships_temp as 
SELECT ship_id, cargo_no, voyage 
FROM ships;
Data Types

- Integers, big and small
- Serials
- Arbitrary precision–numeric
- Floating points
- Serial Types–Identity
- Character Types
- Binary Data, big and small
- Date/Time/Timestamp
- Boolean
- Geometric
- Network Addresses
- Bit Types
- Arrays
- Oids
- Pseudo Types
Data Type Mapping

- Integers.................. 2, 4, 8 bytes
- Serials.................... Identity, Autoincrement
- Numeric.................. Money
- Floats.................... Arithmetic
- Text....................... Character Types
- Date/Time/Interval... Dates & Times
- Timestamp............... Timestamps
- Boolean.................... Boolean
- bytea....................... Byte stream, images
Keys

Primary Keys

- Implemented as B-Tree Unique indexes

Foreign Keys

- Implement Referential Integrity.
- A FK in table A says that this value references a unique value in table B.
- Cascading updates, deletes
- Nulls OK
Defaults & Constraints

- Initialize column with constants
- Check value for validity
- UNIQUE, [NOT] NULL, KEYS

```
CREATE TABLE players ( 
  nick_name text PRIMARY KEY, 
  team_name text REFERENCES teams(team_name), 
  age integer CHECK (age > 15) NOT NULL, 
  games_played integer DEFAULT 0 
);
```
Nulls

- A NULL is a NULL is a NULL
- NULLS are not equal to each other
- NULLS are not equivalent to an empty string
- NULLS are not equivalent to 0
- NULLS are not indexed
TableSpaces

- Creating a tablespace

  CREATE TABLESPACE bd LOCATION '/bigdisk';

- Using a tablespace

  CREATE TABLE FOO (...) TABLESPACE bd;

- Altering a tablespace

  - alter owner, alter name

- Alter a table's tablespace

  ALTER TABLE SET TABLE SPACE TO bd;
SELECT

- Target List – list of columns to be returned
  - any expression,
  - aggregate,
  - subquery,
  - function,
  - columns from FROM clause data sources
SELECT

- FROM – data sources
  - Tables,
  - Views,
  - Set Returning Functions,
  - SubQueries,
  - JOINS,
  - UNIONS
SELECT

- WHERE – boolean expression qualifying data
  - Expressions,
  - Columns,
  - Functions,
  - SubQueries
SELECT

- GROUP BY – scope of Aggregate
  - Elements of Target List not involved in aggregation.
  - Determines Break columns

```sql
select tname, count(match_id)
from tmatches
group by tname;
```
HAVING – boolean expression qualifying aggregates
• Expressions usually involving aggregates

```
select team1, count(matid)
from tmatches
group by team1
having count(matid) > 5;
```
Conditional Statements

- **COALESCE**

  ```python
coalesce(description, short_description, 'N/A')
  ```

- **CASE**

  ```sql
  (select case when $1 is null then '#ffffff'
  else '#000000'
  end)
  ```

- **NULLIF (value1, value2)**

  - NULL if values are equal else value1
SubQuery Expressions

• Expressions and Lists

• EXISTS

  \[\text{WHERE EXISTS (select id from bigtable)}\]

• IN

  \[\text{WHERE thisid IN (select id from bigtable)}\]

• ANY (SOME)

  \[\text{name = ANY (select user from users)}\]

• ALL

  \[\text{due\_date > ALL (select milestones from projects)}\]
UNIONS & JOINS

- Inner Join
- Left Outer Join
- Right Outer Join
- Full Outer Join
SELECT ...
FROM matches m JOIN events e
  USING (matchid)

SELECT ...
FROM matches m JOIN events e
  ON (m.matchid = e.m_id)

SELECT ...
FROM matches m, events e
  WHERE m.matchid = e.matchid
INSERT INTO tmatches
(matid, team1, team2, score1, score2)
VALUES
(DEFAULT, 'Berkeley', 'KC', 40,2);
INSERT

- Target Table
- SubQuery

```
INSERT INTO events (ename, year, descr)
SELECT lower(ename), 2006, description
FROM events2006
WHERE lower(ename) not in
  (select ename from events);
```
UPDATE

- Target Table
- SET Column_Name = Value,
  Column_Name = Value
  expression, value from Target Table, FROM list
- FROM
  Other Tables
- WHERE
  DON'T FORGET THE WHERE CLAUSE!
UPDATE
doctrine
databases
(updated)

UPDATE teams
SET descr = nt.longdescr
FROM newteam_names nt
WHERE teams.sname = nt.sname;
DELETE

- Table Name
- USING
  - Data Sources (i.e. table list)
- WHERE
  - *DON'T FORGET THE WHERE CLAUSE!*

DELETE FROM daily_log
where log_ts < (current_date -1) + '12:00pm':::time
Views

- Named Queries
- Implemented Using Rules
- Can do Updates, Inserts, Deletes via Rules
- Usability

CREATE OR REPLACE VIEW phonelist AS
SELECT t.team, p.player, p.name, p.phone
FROM teams t, p.players
WHERE t.team = p.team;
Blobs, Slobs and TOAST

- Large Objects
  - special interface lo_
  - seek, read, write
- TOAST
  - automatic and invisible promotion
  - INSERT, UPDATE, DELETE
  - no seek
Simple Domains

- Subtype Inherits Parent Type
  - Attributes and
  - Operators, Functions
- May Over Ride
  - DEFAULT, CHECK
  - CONSTRAINT, [NOT] NULL
  - Operators, Functions
Simple Domains

- May Not Over Ride
  - Casts
  - LIKE
  - AS PRIMARY KEY use UNIQUE INDEX

CREATE DOMAIN degrees float CHECK (degrees > -180 and degrees <= 180);
Built-in Functions & Operators

- Logical & Comparison Operators
- Math Functions, Aggregates & Operators
- Type Conversions
- Date, Time & Interval Arithmetic
- String and pattern matching
- Conditional Statements
SELECT ('1/1/' || 2006) + 7*( week - 1 ),
    SUM(cookies), scout_name
FROM cookie_sales c JOIN scouts s
    USING (s.name),
    generate_series(1,53) g(week)
WHERE
    date_part('week',c.sales_date) = week
GROUP BY week, scout_name;
Functions & Operators: Casts

- INTERVAL '2 days 3 hours'
- TIMESTAMP '12/31/59'
- 'gotta wanna'::text
- 16::bigint
- '(1.5,2.7)'::point
- 123.456::numeric(6,3)
Input/Output Functions

- Output Format
  - `to_char( ----, text)`
  - `to_char( idate, 'dd-Mon-YYYY');`
  - `to_char( price, '999D99');`
Input/Output Functions

- Input Format
  - to_date(text, text)
  - to_timestamp(text, text)
  - to_number(text, text)

```
to_date( '31 Dec 2006', 'DD Mon YYYY')
to_timestamp( '5/24/06', 'DD/MM/YY');
to_number( '543', '999D99')
```
Functions & Operators

Interval Arithmetic

- Regular Arithmetic Expressions
  
  current_date + INTERVAL '5 days'
  start_date + duration

- Regular Comparison Operators
  
  item_date > due_date
  start_date + INTERVAL '5 days' <= due_date
  logtime <> last_log
Functions & Operators
Date, Time Arithmetic

- extract( field FROM src)

  extract(epoch FROM
    TIMESTAMP '2004-12-31 01:43:03');
  extract(hours FROM
    INTERVAL '2 days 5 hours');

- age( timestamp )

  age('12/31/1959');
Functions & Operators
Interval Arithmetic

- (start, end) OVERLAPS (start2, end2)

(proposed_start, proposed_end)
OVERLAPS
('12/23/06':::date, '1/4/06':::date)

(session_time, INTERVAL '1 hour')
OVERLAPS
(breaktime, INTERVAL '15 minutes')
Functions & Operators
String and Pattern matching

- **LIKE, ILIKE or ~~, ~~***
  
  ```
  city LIKE 'San_%'
  city ~~ 'San_%'
  city ILIKE 'oak%%'
  city ~~* 'oak%%'
  ```

- **SIMILAR TO or ~, ~***

  ```
  name SIMILAR TO
  '(Mr.|Ms.) [A-Z][a-z]([ a-z])*'
  ```
Indexing Operators

create index uname_idx
    on users (user_name);

create index ttnotes_idx
    on trouble_tickets(ticket_id, note_id);

create index range_idx
    on cows USING RTREE (range);
Functional Indexing

- Functional indexes
  - Result of any immutable procedure
    
    ```sql
    create index tsdate_idx on log_table date(createtimestamp);
    create name_idx on users lower(user_name);
    ```

- Expressional indexes
  - Result of any immutable expression
    
    ```sql
    create overdue_idx on books duedate + '30 days'
    ```
Partial Indexing

- Indexes over parts of tables

```sql
create index active_clients on clients
    where status = 'A';

create index currentyear on accounts
    where reg_date = '2005';
```
Server Side Languages

- PL/pgsql and SQL Primary languages
- Query & Trigger enabled
- Trusted vs. untrusted languages
- Available server side languages
  - PL/perl, PL/pythonu,
  - PL/R, PL/Tcl, PL/Ruby,
  - PL/bash
  - C, etc.
CREATE FUNCTION foo(text, integer)
RETURNS integer AS
$$
...
$$

LANGUAGE 'plpgsql' [OPTIONS...]
PlPgSQL Trigger Functions

- Executes once per row
- Often Used for
  - complex or dynamic defaults
  - logging
Triggers

- Function executed per Row
- Before or After Event
- Insert, Update or Delete

CREATE OR REPLACE FUNCTION lastmod
RETURNS TRIGGER AS $$
BEGIN
    NEW.last_modified = now();
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
CREATE TRIGGER team_upd
BEFORE INSERT OR UPDATE on teams
FOR EACH ROW EXECUTE PROCEDURE lastmod();
Rules

- Re-Write a Query
- Action On a Table or View
- Select Rules Implement Views
- Updateable Views Implemented via Rules
Example View:

CREATE VIEW matches_v
SELECT m.matchname, m.matchid, 
       t1.team AS team1, t2.team AS team2, 
       t1.teamid as t1id, t2.teamid as t2id, 
       e.eventname, m.eventidid
FROM matches m JOIN teams t1 USING 
       (t1.id=teamid)
JOIN teams t2 USING (t2.id=teamid)
JOIN event e ON (eventid);
Rules Implement a View

(Implicit)
CREATE RULE "_RETURN" AS ON SELECT TO matches_v DO INSTEAD SELECT...;
CREATE RULE upd_matches
AS ON UPDATE TO matches_v
DO INSTEAD
UPDATE matches
SET matchname=NEW.matchname,
    eventid=NEW.eventid,
    t1id=NEW.t1id, t2id=NEW.t2id
WHERE matchid=OLD.matchid;
CREATE RULE ins_matches
AS ON INSERT TO matches_v
DO INSTEAD
INSERT INTO matches
(matchid, eventid, t1id, t2id, matchname)
VALUES
(default, NEW.eventid, NEW.t1id, NEW.t2id, NEW.matchname);
CREATE RULE del_matches
AS ON DELETE TO matches_v
DO INSTEAD
DELETE FROM tmatches
WHERE matchid=OLD.matchid
Operators

- Create first class operators
- Implemented by functions
- Use the same way as ordinary built-in operators.
- Natural cost overhead.
Tuning Queries

The usual suspects

- DID YOU VACUUM?
- Type mismatch
- Indexing Expressions
- GUC configurations
- Explaining Explain
- plpgsql-performance@postgresql.org
Explain

- Explain [analyze] [verbose]

OP (cost=n...n rows=n width=n)
    (actual time=t..t rows=n loops=n)
    OP cond: (...)
    -> OP (cost=...)(actual time=...)
    OP cond: (...)

- Look for
  - Seq Scan, Hash Join,
  - Subquery, Hash,
  - Index Scan
  - Index usage
Replication Products

- SLONY-1
- Mammoth Replicator Command Prompt, Inc.
- pgpool (client side)
- postgres-r, dbmirror async, Rserv async, clustgres, pglcluster, osogres (client side replication)
References

- www.postgresql.org
- www.varlena.com/GeneralBits
- Mailing Lists
  - general, sql, novice, interfaces
  - hackers
  - advocacy
  - performance, bugs
  - docs
- IRC #postgresql freenode.net