# Introduction to PostgreSQL

## The Open Source Object-Relational Database Management System

*Varlena, LLC*

*A. Elein Mustain*
*www.varlena.com*
*elein@varlena.com*

## PostgreSQL BSD License

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.

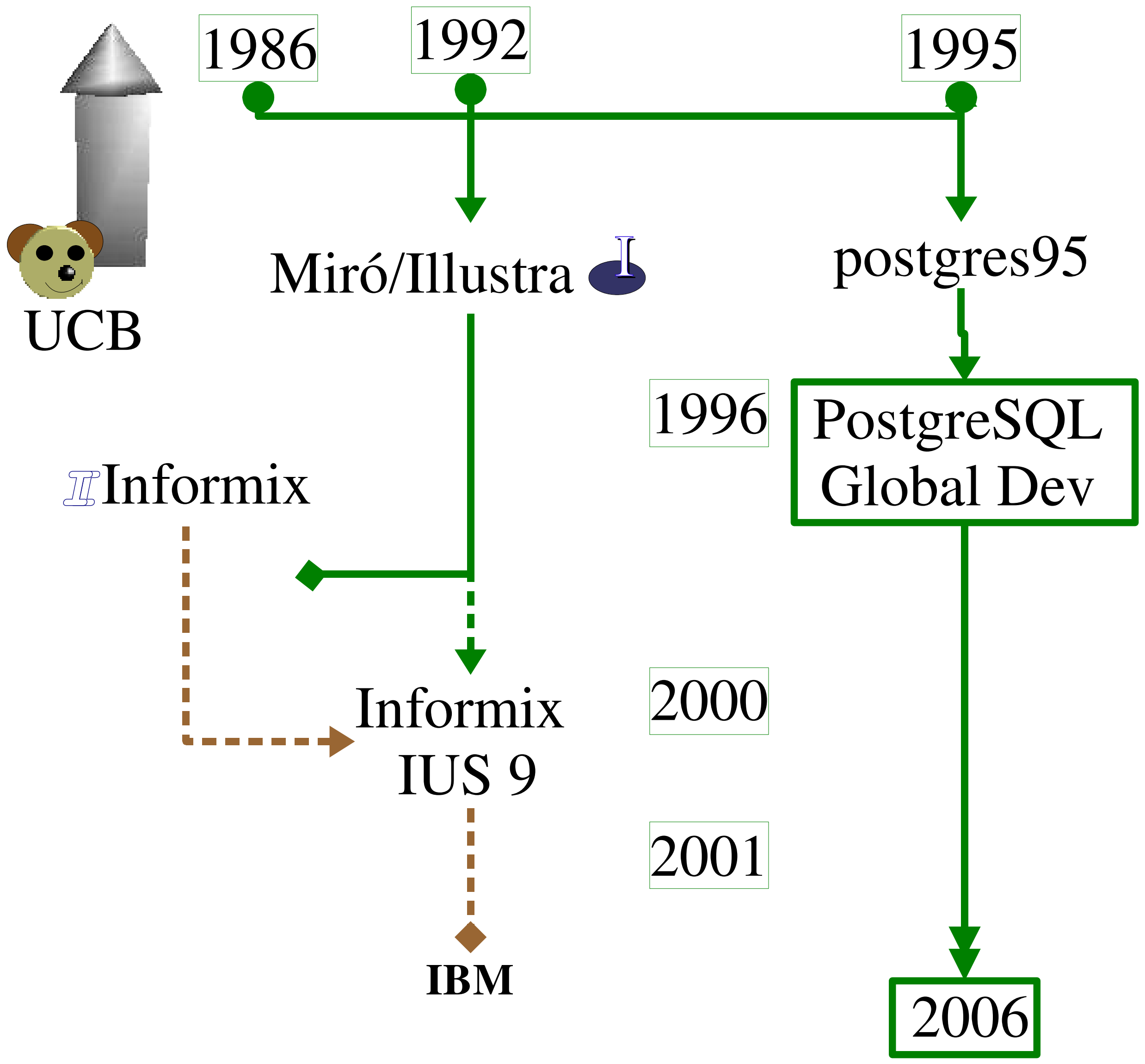
## Agenda

- PostgreSQL  Features
- Installation and Configuration
- Maintenance and Monitoring
- Command Line Interface
- Database Basics in PostgreSQL

## Not the Agenda

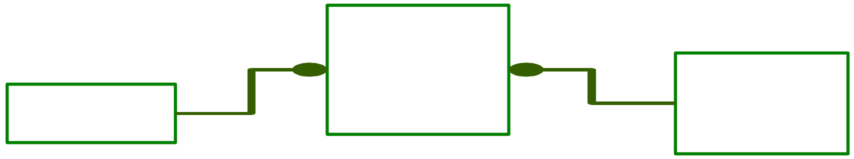
- Client Interfaces
- Inheritance
- Comparisons to other Databases
- Replication,  Point in Time Recovery
- Full Text Search

# History of Postgres



## 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)**

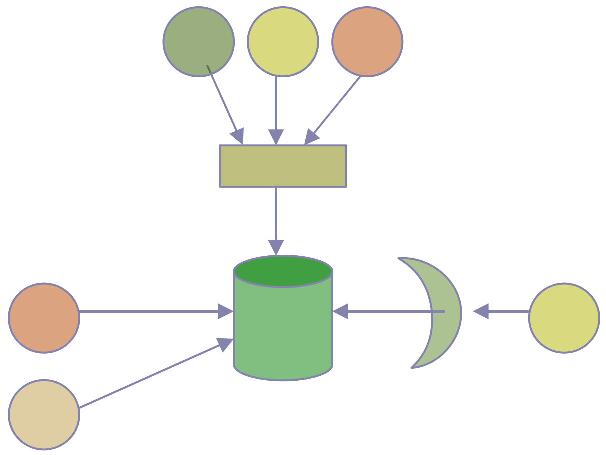


## PostgreSQL Relational Features

- 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 contraints
- 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

## PostgreSQL Queries with Objects

```
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';
```

## 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/pythonu,
  - 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

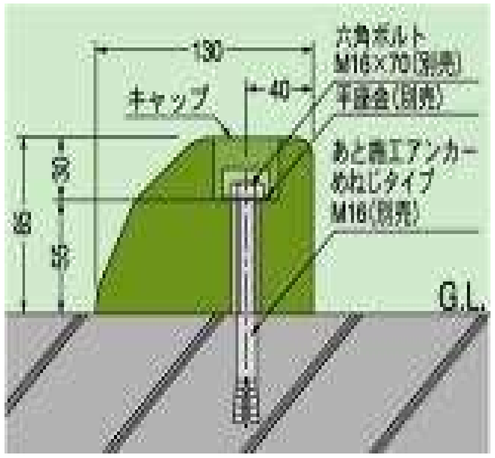
As user postgres ...

```
 $ ./configure \
   --prefix=/local/pgsql81 \
   --with-perl \
   --with-python \
   --with-tcl \
   --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

| Variable | Default | @ 2G RAM |
|---|---:|---:|
| max_connections | 100 | 100 |
| shared_buffers | 1000 | 25000 |
| work_mem | 1024 | 16384 |
| maintenance_work_mem | 16384 | 16384 |
| max_fsm_pages | 20000 | * |
| max_fsm_relations | 1000 | * |
| effective_cache_size | 1000 | 82500 |
| log_destination | stderr | stderr |
| redirect_stderr | off | on |

# Configuration Points
# Global User Configuration

| Variable | Default | @ 2G RAM |
|---|---:|---:|
| log_directory | pg_log | /varl/log/pgsql |
| log_min_duration_statem | -1 | 500 |
| log_line_prefix | | [%p - %t ] |
| log_statement | none | ddl |
| stats_start_collector | on | on |
| stats_command_string | off | on |
| stats_block_level | off | on |
| stats_row_level | off | on |
| autovacuum | off | on |

## Configuration Points Basic Security

```
# Host          DB      USER        ADDRESS         METHOD
# "local" is for Unix domain socket connections only
local           all     all                         trust
# IPv4 local:
host            all     all     127.0.0.1/32        trust
# IPv6 local:
host            all     all     ::1/128             trust
# bad bernie
host            all     bernie  163.555.9.9         reject
# demo
host            demo    varlena 163.555.9.9     trust
# users
host            all    all          163.555.9.9     md5
```

## 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

```
SELECT name, setting, short_desc
FROM pg_settings
ORDER BY name;
```

- What can be set in a session?

```
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!**

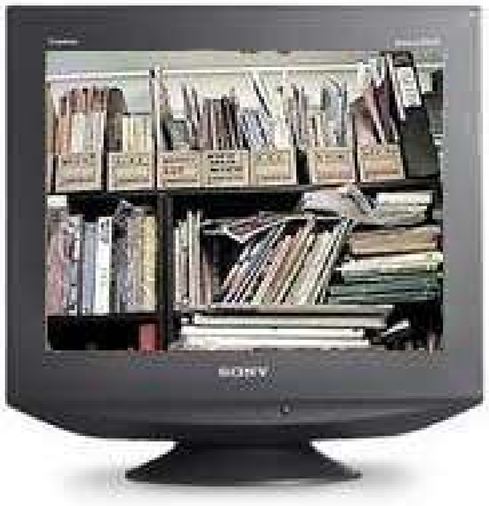
# Monitoring PostgreSQL

Client Server Architecture

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

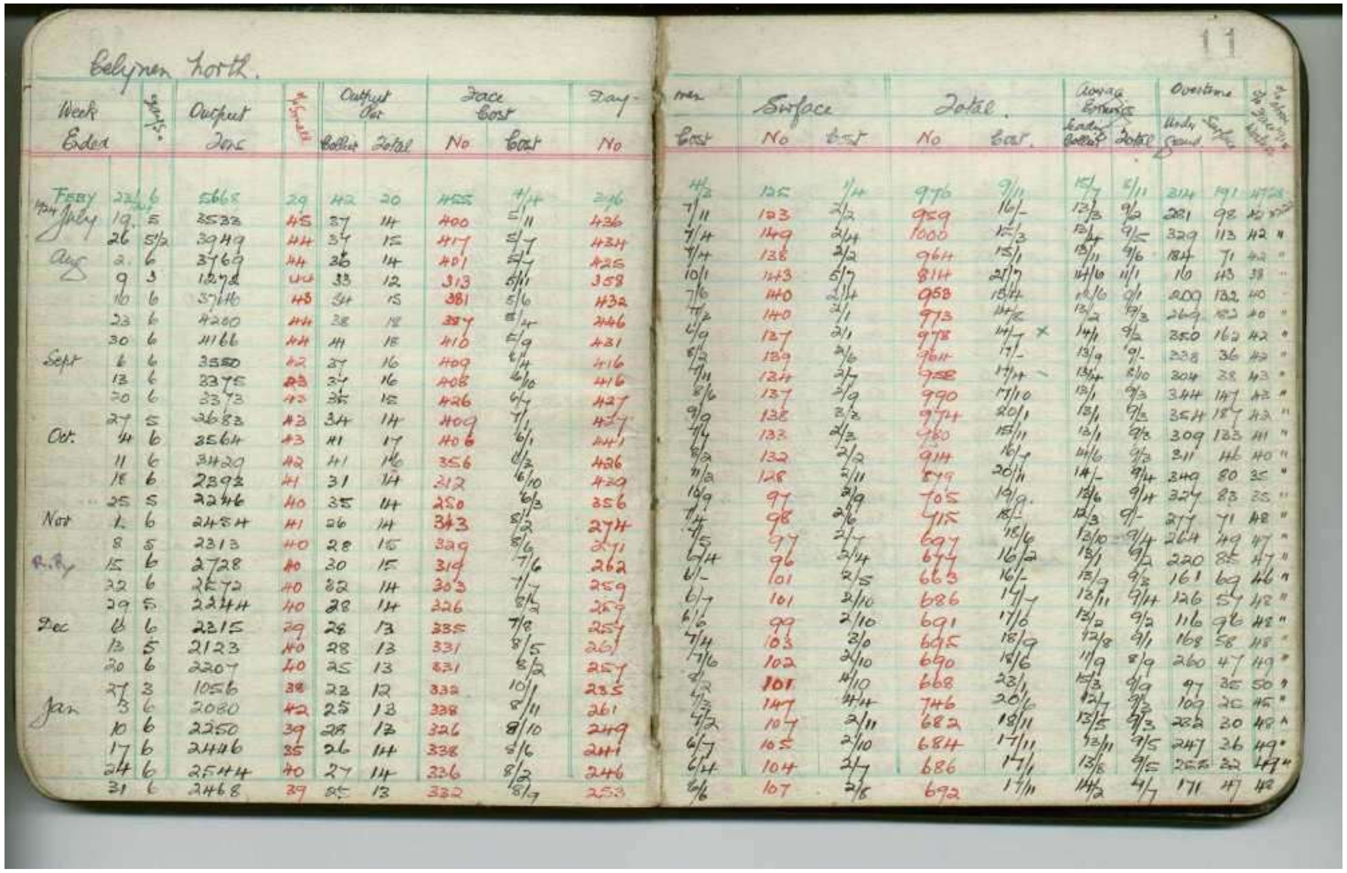
## Documentation and Help

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



## Creating Databases

```
$ createdb accounts
```



Celynen North.

11

| Week Ended | | Shifts | Output Tons | % Small | Output per Man Collier | Output per Man Total | Face Cost No | Face Cost Cost | Day-men No | Day-men Cost | Surface No | Surface Cost | Total No | Total Cost | Average Earnings Loading Collier | Average Earnings Total | Overtime Under Ground | Overtime Surface | [illegible] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Feby | 22 | 6 | 5668 | 29 | 42 | 20 | 455 | 4/4 | 296 | 4/2 | 125 | 1/4 | 976 | 9/11 | 15/7 | 5/11 | 214 | 191 | 47 |
| 1924 July | 19 | 5 | 3533 | 45 | 37 | 14 | 400 | 5/11 | 436 | 7/11 | 123 | 2/2 | 959 | 16/- | 12/3 | 9/2 | 281 | 98 | 40 |
| | 26 | 5½ | 3949 | 44 | 34 | 15 | 417 | 5/7 | 434 | 7/4 | 149 | 2/4 | 1000 | 15/3 | 13/4 | 9/5 | 329 | 113 | 42 |
| Aug | 2 | 6 | 3769 | 44 | 36 | 14 | 401 | 5/7 | 425 | 4/4 | 138 | 2/2 | 964 | 15/1 | 13/11 | 9/6 | 184 | 71 | 42 |
| | 9 | 3 | 1272 | 44 | 33 | 12 | 313 | 6/11 | 358 | 10/1 | 143 | 5/7 | 814 | 21/7 | 14/6 | 11/1 | 16 | 43 | 38 |
| | 16 | 6 | 3746 | 48 | 34 | 15 | 381 | 5/6 | 432 | 7/6 | 140 | 2/4 | 958 | 15/4 | 14/6 | 9/1 | 209 | 132 | 40 |
| | 23 | 6 | 4200 | 44 | 38 | 18 | 384 | 5/4 | 446 | 7/2 | 140 | 2/1 | 973 | 14/2 | 13/2 | 9/3 | 269 | 183 | 40 |
| | 30 | 6 | 4166 | 44 | 41 | 18 | 410 | 5/9 | 431 | 6/9 | 127 | 2/1 | 978 | 14/7 × | 14/1 | 9/2 | 350 | 162 | 42 |
| Sept | 6 | 6 | 3550 | 42 | 37 | 16 | 409 | 6/4 | 416 | 6/2 | 139 | 2/6 | 964 | 17/- | 13/9 | 9/- | 238 | 36 | 42 |
| | 13 | 6 | 3375 | 43 | 34 | 16 | 408 | 6/10 | 416 | 7/11 | 124 | 2/7 | 958 | 17/4 | 13/4 | 8/10 | 304 | 38 | 43 |
| | 20 | 6 | 3373 | 43 | 35 | 15 | 426 | 6/4 | 427 | 8/6 | 137 | 2/9 | 990 | 17/10 | 13/1 | 9/3 | 344 | 147 | 43 |
| | 27 | 5 | 2683 | 43 | 34 | 14 | 409 | 7/1 | 427 | 9/9 | 138 | 3/3 | 974 | 20/1 | 13/1 | 9/3 | 354 | 187 | 42 |
| Oct. | 4 | 6 | 2564 | 43 | 41 | 17 | 406 | 6/1 | 441 | 4/4 | 133 | 2/3 | 980 | 15/11 | 13/1 | 9/3 | 309 | 133 | 41 |
| | 11 | 6 | 3429 | 42 | 41 | 16 | 356 | 6/3 | 426 | 8/2 | 132 | 2/2 | 914 | 16/7 | 14/6 | 9/3 | 311 | 46 | 40 |
| | 18 | 6 | 2392 | 41 | 31 | 14 | 312 | 6/10 | 429 | 7/2 | 128 | 2/11 | 879 | 20/1 | 14/- | 9/4 | 349 | 80 | 35 |
| | 25 | 5 | 2246 | 40 | 35 | 14 | 250 | 6/3 | 356 | 16/9 | 97 | 2/9 | 705 | 19/9 | 13/6 | 9/4 | 324 | 83 | 35 |
| Nov | 1 | 6 | 2454 | 41 | 26 | 14 | 343 | 8/2 | 274 | 7/4 | 98 | 2/6 | 715 | 18/- | 12/3 | 9/- | 277 | 71 | 48 |
| | 8 | 5 | 2313 | 40 | 28 | 15 | 329 | 8/6 | 271 | 4/5 | 97 | 2/7 | 697 | 18/6 | 13/10 | 9/4 | 264 | 49 | 47 |
| R.Ry | 15 | 6 | 2728 | 40 | 30 | 15 | 319 | 7/6 | 262 | 6/4 | 96 | 2/4 | 677 | 16/2 | 13/1 | 9/2 | 220 | 85 | 47 |
| | 22 | 6 | 2672 | 40 | 32 | 14 | 303 | 7/7 | 259 | 6/- | 101 | 2/5 | 663 | 16/- | 13/9 | 9/2 | 161 | 69 | 46 |
| | 29 | 5 | 2244 | 40 | 28 | 14 | 326 | 8/2 | 259 | 6/7 | 101 | 2/10 | 686 | 14/7 | 13/11 | 9/4 | 126 | 57 | 42 |
| Dec | 6 | 6 | 2315 | 29 | 28 | 13 | 335 | 7/8 | 257 | 6/6 | 99 | 2/10 | 691 | 17/6 | 13/2 | 9/2 | 116 | 96 | 48 |
| | 13 | 5 | 2123 | 40 | 28 | 13 | 331 | 8/5 | 261 | 7/4 | 103 | 3/0 | 695 | 18/9 | 12/8 | 9/1 | 168 | 58 | 48 |
| | 20 | 6 | 2207 | 40 | 25 | 13 | 331 | 8/2 | 257 | 7/6 | 102 | 2/10 | 690 | 18/6 | 11/9 | 8/9 | 260 | 47 | 49 |
| | 27 | 3 | 1056 | 38 | 23 | 12 | 332 | 10/1 | 235 | 4/2 | 101 | 4/10 | 668 | 23/1 | 15/3 | 9/9 | 97 | 35 | 50 |
| Jan | 3 | 6 | 2080 | 42 | 25 | 13 | 338 | 8/11 | 261 | 4/3 | 147 | 4/4 | 746 | 20/6 | 12/7 | 9/3 | 109 | 26 | 45 |
| | 10 | 6 | 2250 | 39 | 28 | 13 | 326 | 8/10 | 249 | 4/2 | 107 | 2/11 | 682 | 18/11 | 13/5 | 9/3 | 232 | 30 | 48 |
| | 17 | 6 | 2446 | 35 | 26 | 14 | 338 | 8/6 | 241 | 6/7 | 105 | 2/10 | 684 | 17/11 | 13/11 | 9/5 | 247 | 36 | 49 |
| | 24 | 6 | 2544 | 40 | 27 | 14 | 336 | 8/2 | 246 | 6/4 | 104 | 2/7 | 686 | 17/1 | 13/6 | 9/5 | 255 | 32 | 47 |
| | 31 | 6 | 2468 | 39 | 25 | 13 | 332 | 8/7 | 253 | 6/6 | 107 | 2/8 | 692 | 17/11 | 14/2 | 4/7 | 171 | 47 | 42 |

## 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=# \?
```

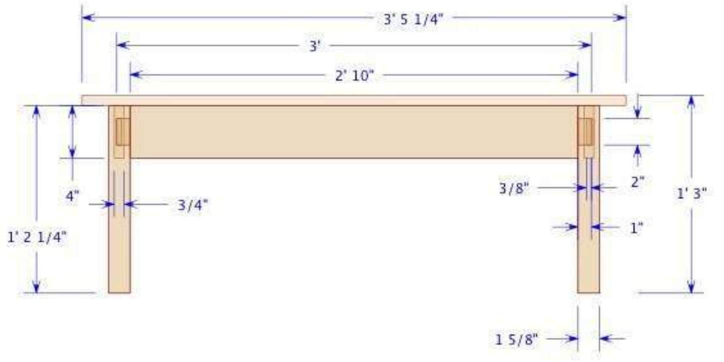
- SQL Help

```
$ psql db
db=# \help [SQL command]
```

## Database Design Elements

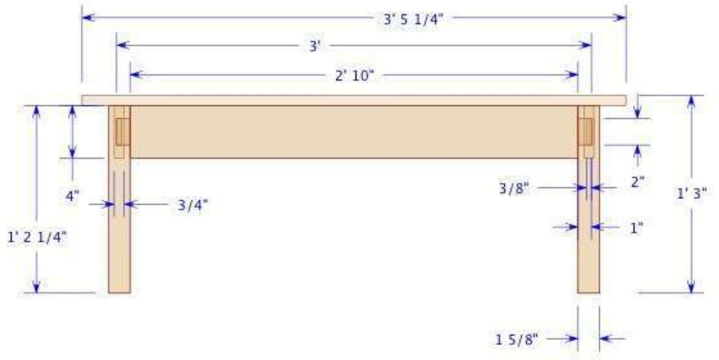
- Data Types & Sequences
- Nulls
- Keys
- Constraints & Defaults
- Triggers, Functions & Operators
- Tablespaces
- Simple domains
- Rules

# 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



```
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

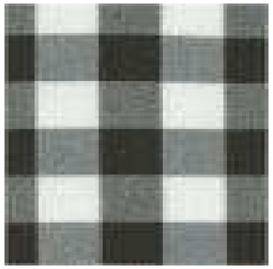
Primary Keys

- 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



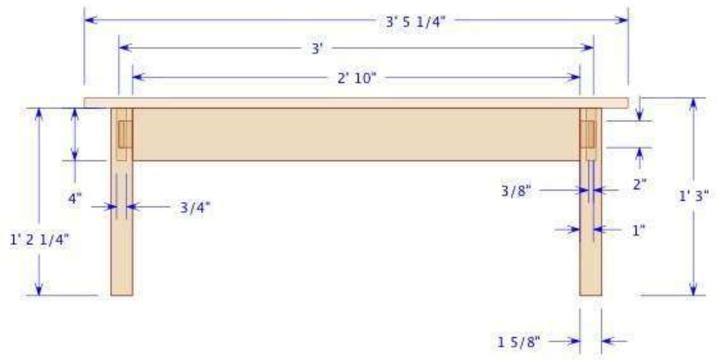
- Initialize column with constants
- Check value for validity
- UNIQUE, [NOT] NULL, KEYS

```
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
```

- 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

```
select tname, count(match_id)
from tmatches
group by tname;
```

## SELECT

- 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

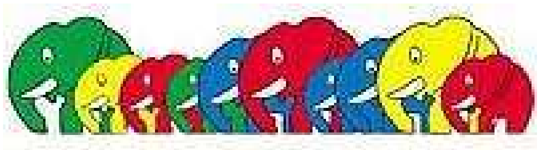
```
coalesce( description,
 short_description, 'N/A')
```

- CASE

```
(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

```
WHERE EXISTS (select id from bigtable)
```

- IN

```
WHERE thisid IN (select id from bigtable)
```

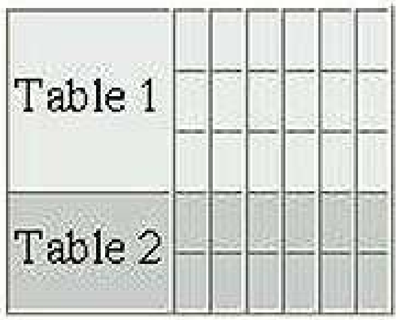
- ANY (SOME)

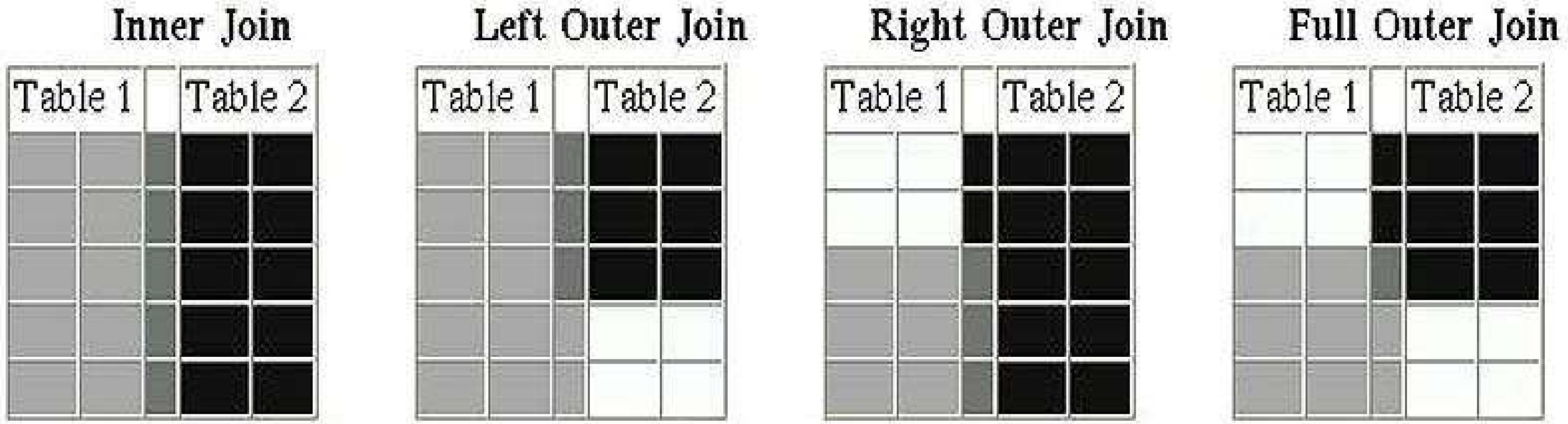
```
name = ANY (select user from users)
```

- ALL

```
due_date > ALL (select milestones from
  projects)
```

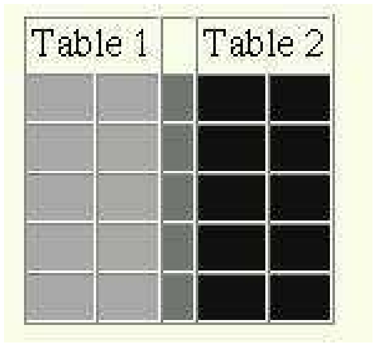
## UNIONS & JOINS





## JOINS: ON, USING, WHERE

```
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

- Target Table
- (Column Names)
- VALUES
- (Column Values)
    - Expressions

```
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

```
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!*

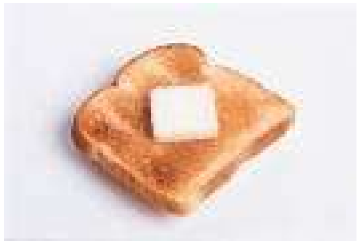
```
DELETE FROM daily_log
where log_ts < (current_date -1)
 + '12:00pm'::time
```

## 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

## Functions & Operators

```
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)
  - timestamp, integer, double precision, numeric

```
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)

(sessiontime, 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])*'
```

## 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

```
create index tsdate_idx on
   log_table date(createtimestamp);
create name_idx on
   users lower(user_name);
```

- Expressional indexes
  - Result of any immutable expression

```
create overdue_idx
   on books duedate + '30 days'
```

## Partial Indexing

- Indexes over parts of tables

```
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.

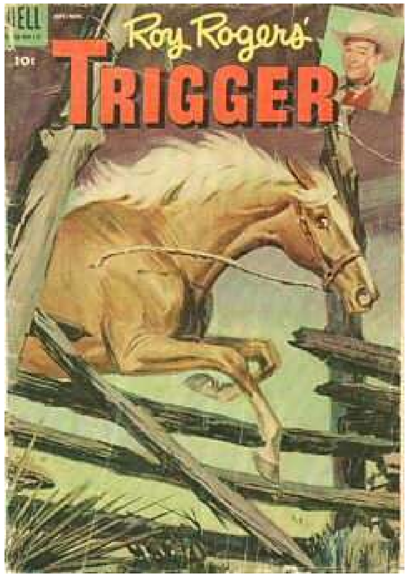
## Server Side Functions

```
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

## Rules View

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.eventid
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...;
```

## Rules Implement a View

```
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;
```

## Rules

```
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);
```

## Rules

```
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

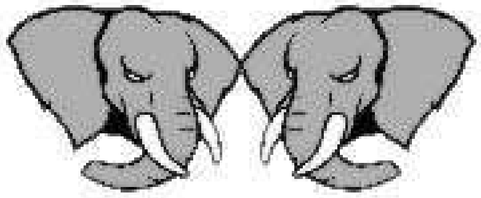
## Explain

- 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