Get to Know

PostgreSQL!

XYZ, ABC

Agenda (must be updated for the final set)

1. Background
2. Practical use of PostgreSQL
3. Features
4. PostgreSQL behind the scenes
5. Replication
6. Use of PostgreSQL from various languages
7. Third party tools
8. How to get started

Background

What is PostgreSQL?

- PostgreSQL is an:
  - advancaed
  - freely available open source
  - relational database management server (RDBMS)

- Supports much of SQL including advanced features:
  - Complex queries, including subselects
  - Foreign keys
  - Triggers
  - Views
  - Transactional integrity (ACID)
  - Multiversion concurrency control (MVCC)

- BSD-style license (“do what you want, but don’t bother us”)

Where does it come from?

Michael Stonebraker

- From INGRES to POSTGRES: 1977-1994
  - Michael Stonebraker, professor at UC @ Berkeley from 1971
  - Developed INGRES from 1977
    - Proof-of-concept for relational databases
    - Established the company Ingres in 1980
    - Bought by Computer Associates in 1994
  - Continued research on POSTGRES from 1986
    - Further development of the concepts in INGRES with a focus on object orientation and the query language Quel
    - The code base of INGRES was not used as a basis for POSTGRES
    - Commercialized as Illustra (bought by Informix, bought by IBM)
- From POSTGRES to PostgreSQL: 1994-1996
  - Support for SQL was added in 1994
  - Released as Postgres95 in 1995
  - Re-released as PostgreSQL 6.0 in 1996
  - Establishment of the PostgreSQL Global Development Team

1977-1985	INGRES
1986-1994	POSTGRES
1994-1995	Postgres95
1996-	PostgreSQL

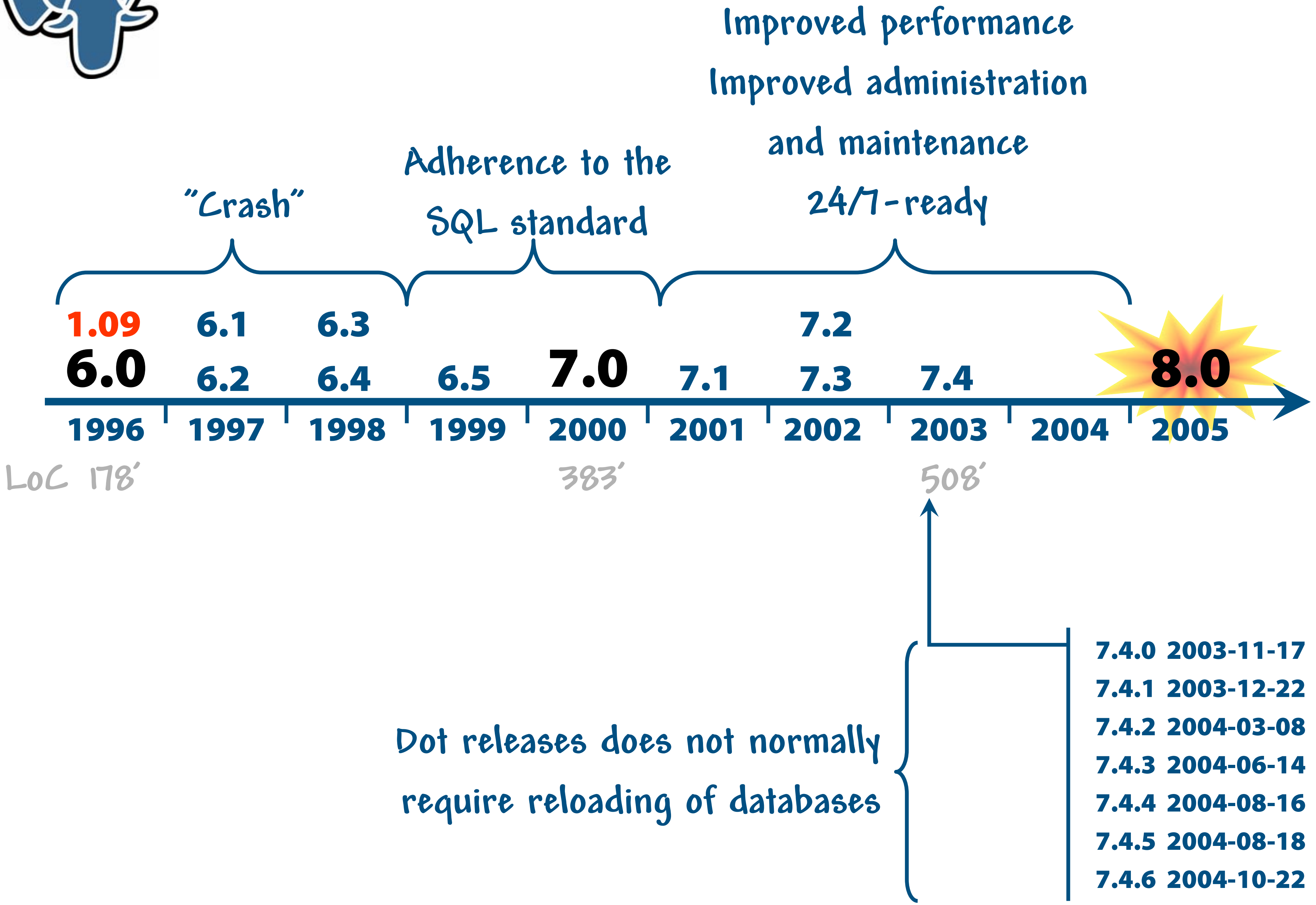
PostgreSQL Global Development Team

- Thomas Lockhart
- Jolly Chen
- Vadim Mikheev
- Jan Wieck
- Andrew Yu
- Tom Lane
- Bruce Momjian
- Marc Fournier

PostgreSQL development

- Core team (a la FreeBSD)
- Source code in CVS (a la FreeBSD)
- http://developer.postgresql.org/
  - Developer-specific mailing lists
  - Centralized TODO list
  - Developer's FAQ
  - Beta-versions of PostgreSQL + documentation
  - Presentations
  - Webinterface to CVS
  - Patches awaiting testing
  - Listing of reported bugs

Release history



Improved performance

Improved administration and maintenance

24/7-ready

"Crash"

Adherence to the SQL standard

1.09 6.0 | 6.1 6.2 | 6.3 6.4 | 6.5 | 7.0 | 7.1 | 7.2 7.3 | 7.4 | | 8.0

1996 1997 1998 1999 2000 2001 2002 2003 2004 2005

LoC 178' 383' 508'

Dot releases does not normally require reloading of databases

7.4.0	2003-11-17
7.4.1	2003-12-22
7.4.2	2004-03-08
7.4.3	2004-06-14
7.4.4	2004-08-16
7.4.5	2004-08-18
7.4.6	2004-10-22

Practical use of PostgreSQL

Installation of PostgreSQL

- FreeBSD:

```
# cd /usr/ports/databases/postgresql80-server
# sudo make install distclean
# cd /usr/ports/databases/postgresql80-client
# sudo make install distclean
# cd /usr/ports/databases/postgresql-docs
# sudo make install distclean
```

```
======================================================================

To initialize the database, you should run initdb as the "pgsql" user.
Example:

  su -l pgsql -c initdb

You can then start PostgreSQL by running:

  /usr/local/etc/rc.d/010.pgsql.sh start

For postmaster settings, see ~pgsql/data/postgresql.conf
For more tips, read ~pgsql/post-install-notes
======================================================================
```

Initializing PostgreSQL

```
pgsql@home> initdb
The files belonging to this database system will be owned by user "pgsql"
This user must also own the server process.
The database cluster will be initialized with locale C.
creating directory /usr/local/pgsql/data... ok
creating directory /usr/local/pgsql/data/base... ok
creating directory /usr/local/pgsql/data/global... ok
creating directory /usr/local/pgsql/data/pg_xlog... ok
creating directory /usr/local/pgsql/data/pg_clog... ok
creating template1 database in /usr/local/pgsql/data/base/1... ok
creating configuration files... ok
initializing pg_shadow... ok
enabling unlimited row size for system tables... ok
initializing pg_depend... ok
creating system views... ok
loading pg_description... ok
creating conversions... ok
setting privileges on built-in objects... ok
creating information schema... ok
vacuuming database template1... ok
copying template1 to template0... ok
Success. You can now start the database server using:
/usr/local/pgsql//bin/postmaster -D /usr/local/pgsql/data
or
/usr/local/pgsql//bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
```

Establishing a database

```
oddbjorn@home ~> createdb demo
createdb: database creation failed: ERROR: permission denied to create database
oddbjorn@home ~> su - pgsql

pgsql@home ~> createdb demo
CREATE DATABASE

pgsql@home ~> psql demo
Welcome to psql 7.4.2, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

demo=# grant all on database demo to oddbjorn;
GRANT

oddbjorn@home ~> psql demo
Welcome to psql 7.4.2, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

demo=>
```

psql: The primary CLI client

```
Usage:
  psql [OPTIONS]... [DBNAME [USERNAME]]

General options:
  -d DBNAME       specify database name to connect to (default: "oddbjorn")
  -c COMMAND      run only single command (SQL or internal) and exit
  -f FILENAME     execute commands from file, then exit
  -l              list available databases, then exit
  -v NAME=VALUE   set psql variable NAME to VALUE
  -X              do not read startup file (~/.psqlrc)
  --help          show this help, then exit
  --version       output version information, then exit

Input and output options:
  -a              echo all input from script
  -e              echo commands sent to server
  -E              display queries that internal commands generate
  -q              run quietly (no messages, only query output)
  -o FILENAME     send query results to file (or |pipe)
  -n              disable enhanced command line editing (readline)
  -s              single-step mode (confirm each query)
  -S              single-line mode (end of line terminates SQL command)

Output format options:
  -A              unaligned table output mode (-P format=unaligned)
  -H              HTML table output mode (-P format=html)
  -t              print rows only (-P tuples_only)
  -T TEXT         set HTML table tag attributes (width, border) (-P tableattr=)
  -x              turn on expanded table output (-P expanded)
  -P VAR[=ARG]    set printing option VAR to ARG (see \pset command)
  -F STRING       set field separator (default: "|") (-P fieldsep=)
  -R STRING       set record separator (default: newline) (-P recordsep=)

Connection options:
  -h HOSTNAME     database server host or socket directory (default: "local socket")
  -p PORT         database server port (default: "5432")
  -U NAME         database user name (default: "oddbjorn")
  -W              prompt for password (should happen automatically)
```

psql: \?: Listing the internal commands

```
General
  \c[onnect] [DBNAME|- [USER]]
                 connect to new database
  \cd [DIR]      change the current working directory
  \copyright     show PostgreSQL usage and distribution terms
  \encoding [ENCODING]
                 show or set client encoding
  \h [NAME]      help on syntax of SQL commands, * for all commands
  \q             quit psql
  \set [NAME [VALUE]]
                 set internal variable, or list all if no parameters
  \timing        toggle timing of commands (currently off)
  \unset NAME    unset (delete) internal variable
  \! [COMMAND]   execute command in shell or start interactive shell

Query Buffer
  \e [FILE]      edit the query buffer (or file) with external editor
  \g [FILE]      send query buffer to server (and results to file or
      |pipe)
  \p             show the contents of the query buffer
  \r             reset (clear) the query buffer
  \s [FILE]      display history or save it to file
  \w [FILE]      write query buffer to file

Input/Output
  \echo [STRING] write string to standard output
  \i FILE        execute commands from file
  \o [FILE]      send all query results to file or |pipe
  \qecho [STRING]
                 write string to query output stream (see \o)
```

```
Informational
  \d [NAME]      describe table, index, sequence, or view
  \d{t|i|s|v|S} [PATTERN] (add "+" for more detail)
                 list tables/indexes/sequences/views/system tables
  \da [PATTERN]  list aggregate functions
  \dc [PATTERN]  list conversions
  \dC            list casts
  \dd [PATTERN]  show comment for object
  \dD [PATTERN]  list domains
  \df [PATTERN]  list functions (add "+" for more detail)
  \dn [PATTERN]  list schemas
  \do [NAME]     list operators
  \dl            list large objects, same as \lo_list
  \dp [PATTERN]  list table access privileges
  \dT [PATTERN]  list data types (add "+" for more detail)
  \du [PATTERN]  list users
  \l             list all databases (add "+" for more detail)
  \z [PATTERN]   list table access privileges (same as \dp)

Formatting
  \a             toggle between unaligned and aligned output mode
  \C [STRING]    set table title, or unset if none
  \f [STRING]    show or set field separator for unaligned query output
  \H             toggle HTML output mode (currently off)
  \pset NAME [VALUE]
                 set table output option
                 (NAME := {format|border|expanded|fieldsep|footer|null|
                 recordsep|tuples_only|title|tableattr|pager})
  \t             show only rows (currently off)
  \T [STRING]    set HTML <table> tag attributes, or unset if none
  \x             toggle expanded output (currently off)

Copy, Large Object
  \copy ...      perform SQL COPY with data stream to the client host
  \lo_export
  \lo_import
  \lo_list
  \lo_unlink     large object operations
```

psql: \d: Describe

```
\d [NAME]      describe table, index, sequence, or view

\d{t|i|s|v|S} [PATTERN] (add "+" for more detail)
          list tables/indexes/sequences/views/system tables

\da [PATTERN]  list aggregate functions
\dc [PATTERN]  list conversions
\dC            list casts
\dd [PATTERN]  show comment for object
\dD [PATTERN]  list domains
\df [PATTERN]  list functions (add "+" for more detail)
\dn [PATTERN]  list schemas
\do [NAME]     list operators
\dl            list large objects, same as \lo_list
\dp [PATTERN]  list table access privileges
\dT [PATTERN]  list data types (add "+" for more detail)
\du [PATTERN]  list users
\l             list all databases (add "+" for more detail)
\z [PATTERN]   list table access privileges (same as \dp)
```

psql: Example of \d in use

```
testdb=> CREATE TABLE my_table (
testdb(>  first integer not null default 0,
testdb(>  second text
testdb-> );
CREATE TABLE

testdb=> \d my_table
             Table "my_table"
 Attribute |  Type   |      Modifier
-----------+---------+--------------------
 first     | integer | not null default 0
 second    | text    |
```

psql: \h: SQL-help

```
ABORT                     CREATE LANGUAGE           DROP TYPE
ALTER AGGREGATE           CREATE OPERATOR CLASS     DROP USER
ALTER CONVERSION          CREATE OPERATOR           DROP VIEW
ALTER DATABASE            CREATE RULE               END
ALTER DOMAIN              CREATE SCHEMA             EXECUTE
ALTER FUNCTION            CREATE SEQUENCE           EXPLAIN
ALTER GROUP               CREATE TABLE              FETCH
ALTER LANGUAGE            CREATE TABLE AS           GRANT
ALTER OPERATOR CLASS      CREATE TRIGGER            INSERT
ALTER SCHEMA              CREATE TYPE               LISTEN
ALTER SEQUENCE            CREATE USER               LOAD
ALTER TABLE               CREATE VIEW               LOCK
ALTER TRIGGER             DEALLOCATE                MOVE
ALTER USER                DECLARE                   NOTIFY
ANALYZE                   DELETE                    PREPARE
BEGIN                     DROP AGGREGATE            REINDEX
CHECKPOINT                DROP CAST                 RESET
CLOSE                     DROP CONVERSION           REVOKE
CLUSTER                   DROP DATABASE             ROLLBACK
COMMENT                   DROP DOMAIN               SELECT
COMMIT                    DROP FUNCTION             SELECT INTO
COPY                      DROP GROUP                SET
CREATE AGGREGATE          DROP INDEX                SET CONSTRAINTS
CREATE CAST               DROP LANGUAGE             SET SESSION AUTHORIZATION
CREATE CONSTRAINT TRIGGER DROP OPERATOR CLASS       SET TRANSACTION
CREATE CONVERSION         DROP OPERATOR             SHOW
CREATE DATABASE           DROP RULE                 START TRANSACTION
CREATE DOMAIN             DROP SCHEMA               TRUNCATE
CREATE FUNCTION           DROP SEQUENCE             UNLISTEN
CREATE GROUP              DROP TABLE                UPDATE
CREATE INDEX              DROP TRIGGER              VACUUM
```

CREATE / ALTER / DROP of objects

- AGGREGATE
- CAST
- CONSTRAINT
- CONVERSION
- DATABASE
- DOMAIN
- FUNCTION
- GROUP
- LANGUAGE
- OPERATOR
- RULE
- SCHEMA
- SEQUENCE
- TABLE
- TYPE
- TRIGGER
- USER
- VIEW

SQL-transactions and maintenance

- Inserting, updating and deleting data
  - INSERT / UPDATE / DELETE
  - COPY
  - TRUNCATE
- Queries
  - SELECT
  - SELECT INTO
- Permissions
  - GRANT / REVOKE
- Maintenance and optimization
  - EXPLAIN
  - ANALYZE
  - VACUUM

SQL: Miscellaneous

- Transactional support
  - BEGIN / ABORT / ROLLBACK / CHECKPOINT / COMMIT
  - SET TRANSACTION / START TRANSACTION / SET CONSTRAINTS
- Cursors
  - DECLARE / FETCH / MOVE / CLOSE
- Triggers
  - LISTEN / NOTIFY / UNLISTEN
- Parameters
  - SHOW / SET / RESET
- Miscellaneous
  - PREPARE / EXECUTE / DEALLOCATE
  - LOAD
  - LOCK
  - COMMENT
  - REINDEX
  - CLUSTER
  - SET SESSION AUTHORIZATION

psql: Example of \h select

```
testdb=> \h select
Command:     SELECT
Description: retrieve rows from a table or view
Syntax:
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR UPDATE [ OF table_name [, ...] ] ]

where from_item can be one of:

    [ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias [, ...]
   ) ] ]
    ( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
    function_name ( [ argument [, ...] ] ) [ AS ] alias [ (
   column_alias [, ...] | column_definition [, ...] ) ]
    function_name ( [ argument [, ...] ] ) AS ( column_definition [,
   ...] )
    from_item [ NATURAL ] join_type from_item [ ON join_condition |
   USING ( join_column [, ...] ) ]
```

psql: Miscellaneous features

- Batch use of psql:
  - psql -f file.sql *database*
  - program | psql -f - *database*

- Support for readline

- Built-in support for timing queries:

```
db=> \timing
Timing is on.
net=> select count(*) from table;
 count
-------
 25523
(1 row)
Time: 52.729 ms
```

- Choose output format
  - `HTML|format|border|expanded|fieldsep|footer|null`
  - `recordsep|tuples_only|title|tableattr|pager`

psql: Bulk copy of data: \COPY

- Loads TSV data from files in one transaction
  - Advantage: fast
  - Disadvantage: if one row isn't accepted, all the rows from the file are thrown away

- \copy tablename from 'filename'

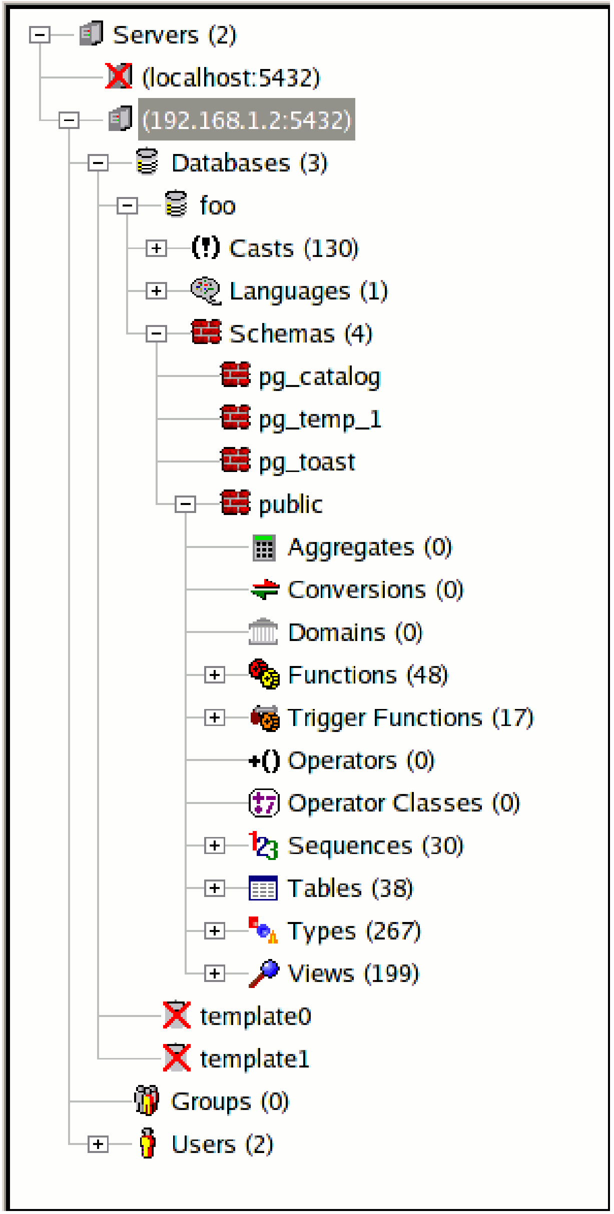
- psql also supports loading of large objects (lo__*)

pgAdmin III

- Freely available graphical administration application for PostgreSQL

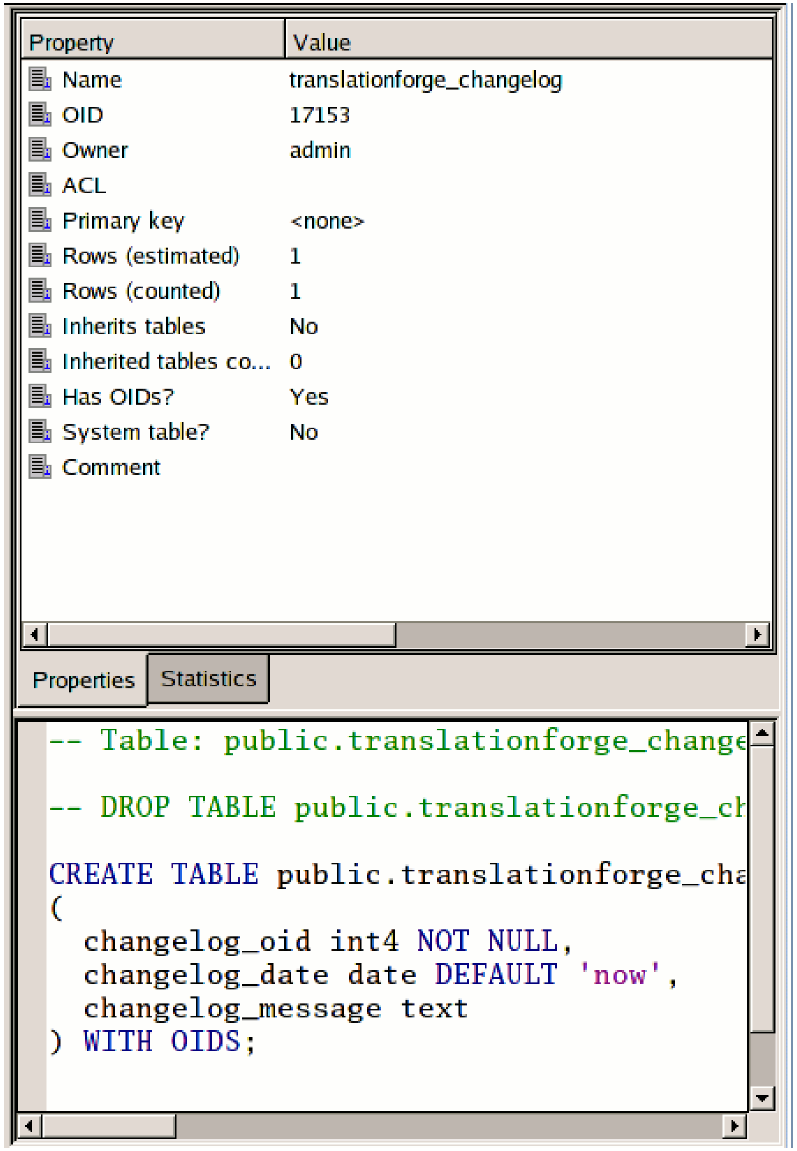
- Runs on:
  - Linux,
  - FreeBSD &
  - Windows

- Version 1.2 supports 8.0



```
Servers (2)
    (localhost:5432)
    (192.168.1.2:5432)
        Databases (3)
            foo
                Casts (130)
                Languages (1)
                Schemas (4)
                    pg_catalog
                    pg_temp_1
                    pg_toast
                    public
                        Aggregates (0)
                        Conversions (0)
                        Domains (0)
                        Functions (48)
                        Trigger Functions (17)
                        Operators (0)
                        Operator Classes (0)
                        Sequences (30)
                        Tables (38)
                        Types (267)
                        Views (199)
            template0
            template1
        Groups (0)
        Users (2)
```

pgAdmin III: Screenshots

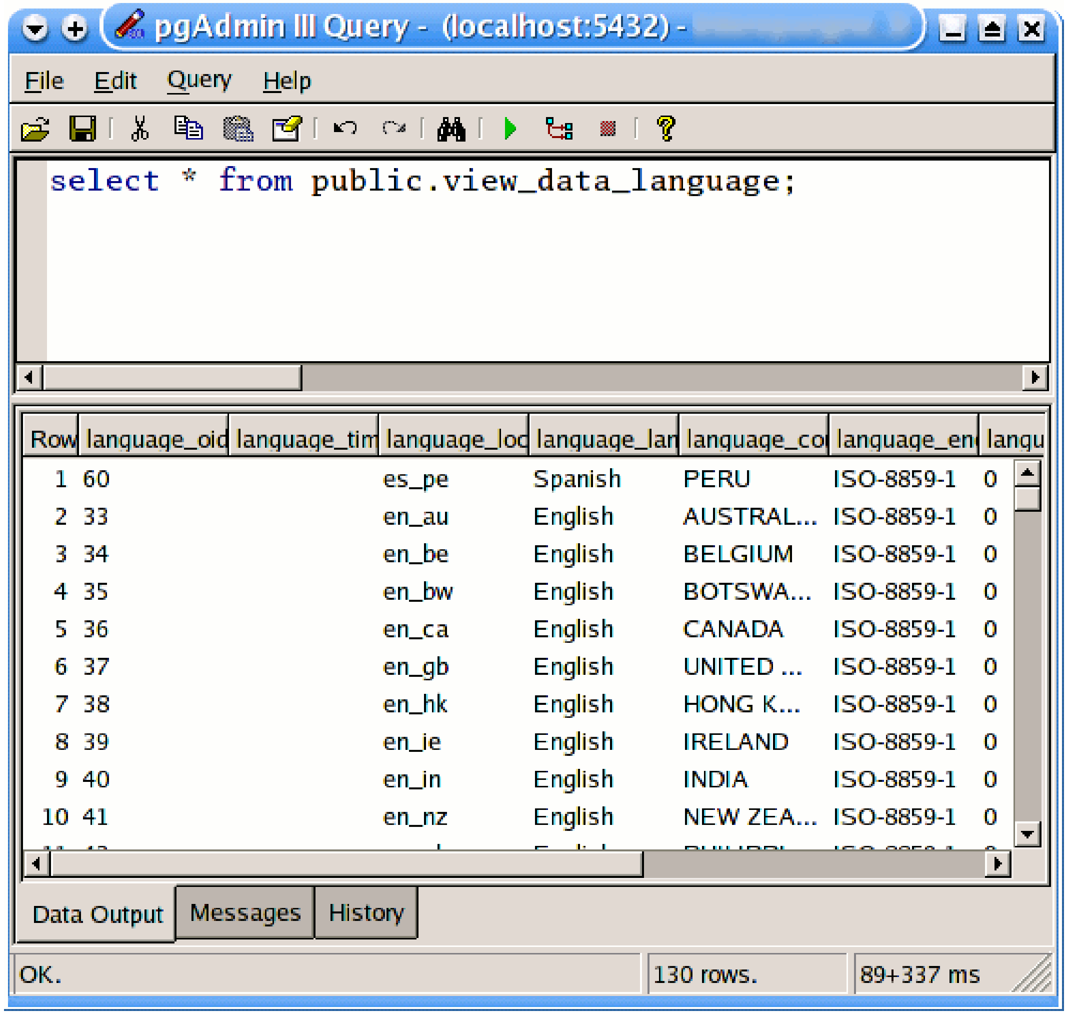


Property	Value
Name	translationforge_changelog
OID	17153
Owner	admin
ACL	
Primary key	<none>
Rows (estimated)	1
Rows (counted)	1
Inherits tables	No
Inherited tables co...	0
Has OIDs?	Yes
System table?	No
Comment	

```
-- Table: public.translationforge_change

-- DROP TABLE public.translationforge_ch

CREATE TABLE public.translationforge_cha
(
  changelog_oid int4 NOT NULL,
  changelog_date date DEFAULT 'now',
  changelog_message text
) WITH OIDS;
```



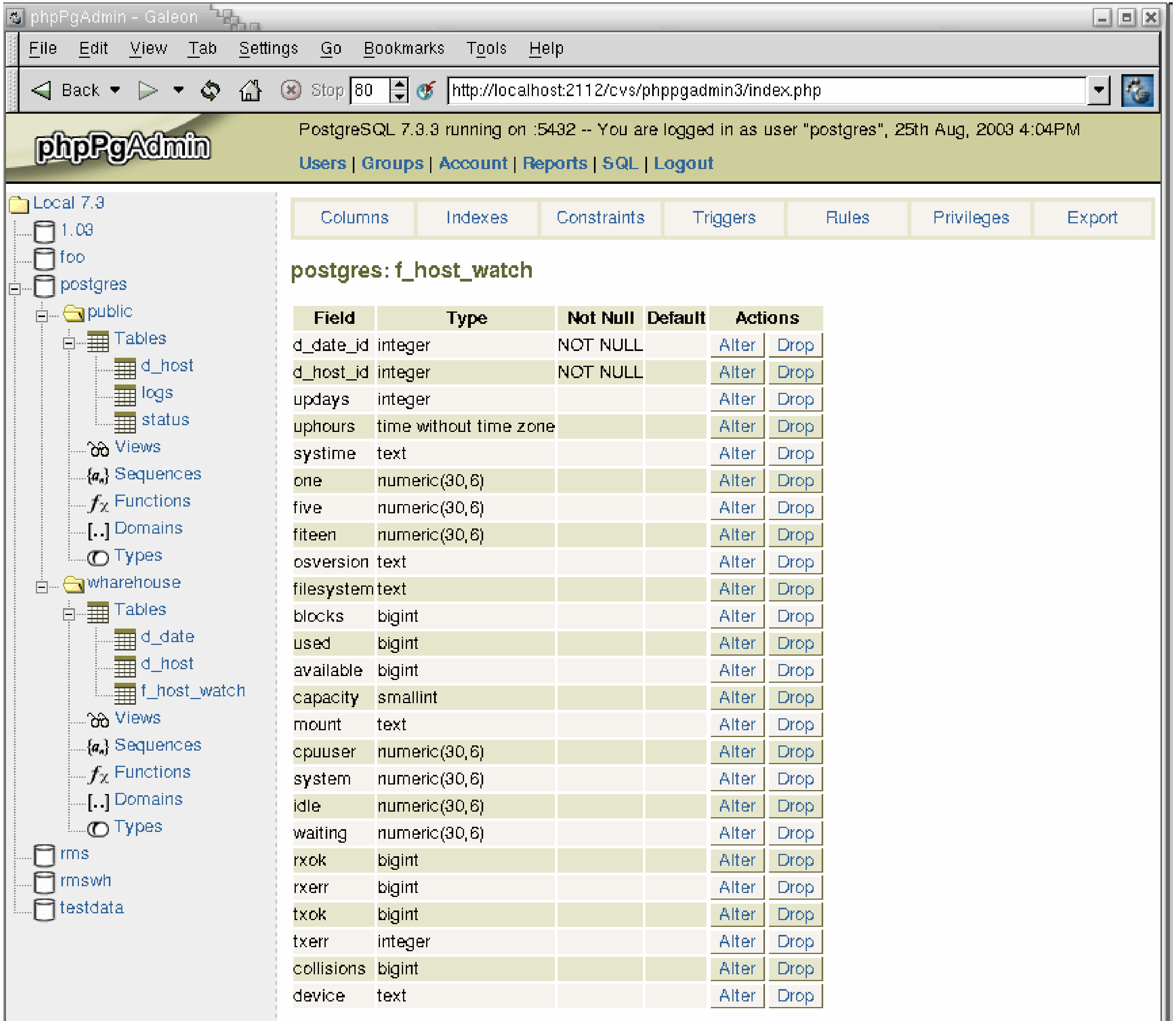
pgAdmin III Query - (localhost:5432) - [redacted]

```
select * from public.view_data_language;
```

Row	language_oid	language_tim	language_loc	language_lan	language_co	language_en	langu
1	60		es_pe	Spanish	PERU	ISO-8859-1	0
2	33		en_au	English	AUSTRAL...	ISO-8859-1	0
3	34		en_be	English	BELGIUM	ISO-8859-1	0
4	35		en_bw	English	BOTSWA...	ISO-8859-1	0
5	36		en_ca	English	CANADA	ISO-8859-1	0
6	37		en_gb	English	UNITED ...	ISO-8859-1	0
7	38		en_hk	English	HONG K...	ISO-8859-1	0
8	39		en_ie	English	IRELAND	ISO-8859-1	0
9	40		en_in	English	INDIA	ISO-8859-1	0
10	41		en_nz	English	NEW ZEA...	ISO-8859-1	0

OK. 130 rows. 89+337 ms

phpPgAdmin



PostgreSQL 7.3.3 running on :5432 -- You are logged in as user "postgres", 25th Aug, 2003 4:04PM

Users | Groups | Account | Reports | SQL | Logout

- Local 7.3
  - 1.03
  - foo
  - postgres
    - public
      - Tables
        - d_host
        - logs
        - status
      - Views
      - Sequences
      - Functions
      - Domains
      - Types
    - wharehouse
      - Tables
        - d_date
        - d_host
        - f_host_watch
      - Views
      - Sequences
      - Functions
      - Domains
      - Types
  - rms
  - rmswh
  - testdata

postgres: f_host_watch

Field	Type	Not Null	Default	Actions	
d_date_id	integer	NOT NULL		Alter	Drop
d_host_id	integer	NOT NULL		Alter	Drop
updays	integer			Alter	Drop
uphours	time without time zone			Alter	Drop
systime	text			Alter	Drop
one	numeric(30,6)			Alter	Drop
five	numeric(30,6)			Alter	Drop
fiteen	numeric(30,6)			Alter	Drop
osversion	text			Alter	Drop
filesystem	text			Alter	Drop
blocks	bigint			Alter	Drop
used	bigint			Alter	Drop
available	bigint			Alter	Drop
capacity	smallint			Alter	Drop
mount	text			Alter	Drop
cpuuser	numeric(30,6)			Alter	Drop
system	numeric(30,6)			Alter	Drop
idle	numeric(30,6)			Alter	Drop
waiting	numeric(30,6)			Alter	Drop
rxok	bigint			Alter	Drop
rxerr	bigint			Alter	Drop
txok	bigint			Alter	Drop
txerr	integer			Alter	Drop
collisions	bigint			Alter	Drop
device	text			Alter	Drop

Pgbash: PostgreSQL access from the shell

```
home ~> pgbash
Welcome to Pgbash version 7.3 ( bash-2.05a.0(1)-release )

  Type '?' for HELP.
  Type 'connect to DB;' before executing SQL.
  Type 'SQL;' to execute SQL.
  Type 'exit' or 'Ctrl+D' to terminate Pgbash.

home ~> CONNECT TO testdb;

home ~> SELECT * FROM iso3166 LIMIT 10;
 cc | country
----+----------------------
 AF | Afghanistan
 AL | Albania
 DZ | Algeria
 AS | American Samoa
 AD | Andorra
 AO | Angola
 AI | Anguilla
 AQ | Antarctica
 AG | Antigua and Barbuda
 AR | Argentina
(10 rows)
```

Miscellaneous commands

- Administrative tools
  - pg_ctl - start, stop eller restart av server
  - pg_config - dumper config-informasjon

- Dump & restore
  - pg_dump og pg_dumpall
    - Dumping one or all databases, respectively
    - Choose everything / schema only / data only
    - Output:
      - plain-text SQL,
      - tar,
      - custom archive format with compression

  - pg_restore
    - Loads input from the non-plaintext outputs of pg_dump
    - (psql loads the plaintext variants)

Contrib

- Dedicated contrib distribution with extensions and utilities:

- dblink - Allows remote query execution
- dbmirror - Replication server
- dbsize - Reports database and table disk space
- fuzzystrmatch - Levenshtein, metaphone, and soundex fuzzy string matching
- isbn_issn - PostgreSQL type extensions for ISBN and ISSN
- mysql - Utility to convert MySQL schema dumps to PostgreSQL
- oracle - Converts Oracle database schema to PostgreSQL
- pg_autovacuum - Automatically performs vacuum
- pg_upgrade - Upgrade from previous PostgreSQL version
- pgbench - TPC-B like benchmarking tool
- pgcrypto - Cryptographic functions
- reindexdb - Reindexes a database
- apache_logging - Getting Apache to log to PostgreSQL
- tsearch2 - Full-text-index support using GiST
- xml2 - Storing XML in PostgreSQL

PostgreSQL features

Overall features

- Freely available; no license costs to worry about
- Proven robustness over many years
- Designed to require minimal administration
- Simple, but good administrative tools (both CLI & GUI-based)
- Portable, runs on"all" relevant plattforms
- Extensible, with a well documented API for additional features
- A number of alternatives for high availability and replication
- Very good "de facto" support
  - With the option of commercial support from many companies

Features

- Database
  - Fully ACID compliance
  - Foreign keys (referential integrity)
  - Better than row-level locking (MVCC)
  - Functional and partial indices
- Development
  - Stored procedures 5.0
  - Procedural languages
  - Native interfaces for ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python, and Ruby
  - Open and documented API
- Security
  - Native SSL support
  - Native Kerberos authentication
- SQL-støtte
  - Good ANSI SQL-support
  - Rules
  - Views 5.0
  - Triggers 5.1 (rudimentary)
  - Cursors 5.0
  - Unicode
  - Sequences 5.1?
  - Inheritance ?
  - Outer Joins
  - Sub-selects
  - Support for UNION (ALL/EXCEPT)
- Extensible
  - Data types
  - Functions
  - Operators

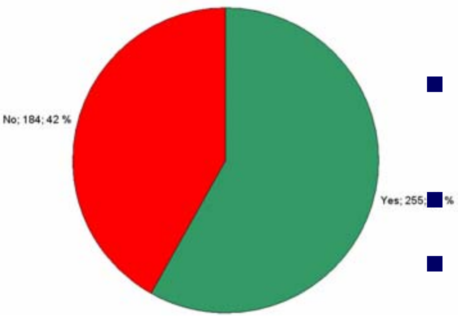
The numbers in red show when mySQL is supposed to get similar features.

Compliance with the SQL standard

- The SQL standard
  - ISO/IEC 9075 "Database Language SQL"
  - Last revision in 2003, aka ISO/IEC 9075:2003 or just SQL:2003
  - Earlier versions was SQL:1999 and SQL-92, but SQL:2003 supercedes both

  - The requirements are defined as individual features:
    - "Core", which all SQL implementations must implement
    - the rest is optional, grouped in"packages"
  - No known RDBMS system today fully supports Core SQL:2003

- PostgreSQL versus SQL:2003
  - PostgreSQL is trying to adhere to there standard where possible, without destroying backward compatibility and common sense
  - Much of SQL:2003 is supported, but sometimes with a slightly different syntax
  - Further compliance is continually implemented
  - Of 255 requirements are currently 58%



Features to ensure data integrity: ACID

- Atomic
  - A transaction is inseperable– "all or nothing"
- Consistent
  - A transaction shall bring the database from one consistent state to another consistent state, even if its not necessarily consistent during the transaction.
- Isolated
  - Transactions are not affected by changes done by concurrent transactions
- Durable
  - When a transaction is COMMITed, the changes are permanent, even after a crash

MultiVersion Concurrency Control (MVCC)

- Traditional row-locking locks the row for the duration of an update. MVCC, on the other hand, maintains versions of each row. This enable:

  1. Every transaction see a snapshot of the database as it was when the transaction started, regardless of what concurrent transactions might be doing
  2. Reading does not block writing
  3. Writing does not block reading
  4. Writing only blocks writing when updating the same row

- Another advantage of MVCC is the possibility of consistent hot backups

- See *“Transaction Processing in PostgreSQL”* by Tom Lane

Transactions

- Tightly coupled to ACID/MVCC is the notion of transactions:
  - A transaction groups several operations to one atomic operation
  - The result of the transaction is ´all or nothing´

One transaction

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
   WHERE name = ’Alice’;
UPDATE branches SET balance = balance - 100.00
   WHERE name = (SELECT branch_name FROM accounts
   WHERE name = ’Alice’);
UPDATE accounts SET balance = balance + 100.00
   WHERE name = ’Bob’;
UPDATE branches SET balance = balance + 100.00
   WHERE name = (SELECT branch_name FROM accounts
   WHERE name = ’Bob’);
COMMIT;
```

Views

- A view masks a query behind a virtual table. Advantages:
  - A consistent interface to the data, even if the tables behind it changes
  - Can masks the details of the tables
  - Queries against views can reduce complexity
  - Can improve security by giving selective access to data

- Merging selected columns from two tables:

```
CREATE VIEW myview AS
   SELECT city, temp_lo, temp_hi, prcp, date, location
   FROM weather, cities
   WHERE city = name;

SELECT * FROM myview;
```

- PostgreSQL does not currently support materialized views

Schemas

- Schemas provide a means to separate the namespace within a database, almost like directories in a file hierarchy (but just one level). Provides the following possibilities:
  - Logical grouping of database objects
  - Separate various users from each other
  - Avoiding name collisions in large databases
- Does not affect the permissions

```
CREATE SCHEMA blug;
CREATE TABLE blug.tabell (..);
SHOW   search_path;
DROP   SCHEMA blug [CASCADE];
```

Constraints

- The data type of a column define which kind of data that's acceptable; constraints give further flexibility in quality checking the data

- PostgreSQL supports five types of constraints
  - Check - `price numeric CHECK (price > 0)`
  - Not NULL - `product_no integer NOT NULL`
  - Uniqueness - `product_no integer UNIQUE`
  - Primary keys - Unique+!NULL: `PRIMARY KEY (a, c)`
  - Foreign keys - `product_no integer REFERENCES products (product_no),`

Triggers

- A trigger can be defined to either execute before or after an INSERT, UPDATE or DELETE, either per statement or per modified row

- Example:

```
CREATE TRIGGER if_film_exists
    BEFORE DELETE OR UPDATE ON distributors
    FOR EACH ROW
    EXECUTE PROCEDURE check_foreign_key
          (1, 'CASCADE', 'did', 'films', 'did');
```

- The trigger function must be defined in one of the available procedural languages

Inheritance

- Inheritance in PostgreSQL is roughly the same concept as inheritance in object-oriented languages like C++ and Java

- A table inheriting another table get all the columns from the parent table

- Possibility of limiting queries to only the parent table:
  - SELECT a, b from ONLY tableA

- Supported by UPDATE, DELETE and other statements

- Not currently fully integrated with unique and foreign key constraints

Example of inheritance

```
CREATE TABLE capitals (
    name text,
    population real,
    altitude int, -- (in ft)
    state char(2)
);
CREATE TABLE non_capitals (
    name text,
    population real,
    altitude int -- (in ft)
);
CREATE VIEW cities AS
    SELECT name, population, altitude FROM capitals
UNION
    SELECT name, population, altitude FROM non_capitals;
```

```
CREATE TABLE cities (
    name text,
    population real,
    altitude int -- (in ft)
);

CREATE TABLE capitals (
    state char(2)
) INHERITS (cities);
```

Cursors

- Cursorer give the ability of ´chunking´ the result set, thus making it easier to process.

- This can be used to avoid resource problems in the client, and supports returning a reference to a cursor instead of the complete result set

Sequences

```
testdb=> CREATE TABLE tabell (
   id    integer default nextval('news_id') UNIQUE not
   NULL,
   news       text not NULL,
   post_time  time default now()
);

testdb=> INSERT INTO tabell (news) VALUES (‘abc');
INSERT 7259941 1
testdb=> INSERT INTO tabell (news) VALUES (‘def');
INSERT 7259943 1
testdb=> INSERT INTO tabell (news) VALUES (‘ghi');
INSERT 7259944 1

testdb=> SELECT * FROM tabell;
  id  | news   | post_time
------+-------+----------
 1000 | abc    | 15:18:40
 1001 | def    | 15:18:56
 1002 | ghi    | 15:19:36
```

Subqueries

- Subqueries as a constant:

```
SELECT f1.firstname, f1.lastname, f1.state
  FROM friend f1
  WHERE f1.state <> ( SELECT f2.state
                FROM friend f2
                WHERE f2.firstname = ‘Dick’ AND
                      f2.lastname  = ‘Cleason’ );
```

- Subqueries as correlated values:

```
SELECT f1.firstname, f1.lastname, f1.age
  FROM friend f1
  WHERE age = (  SELECT MAX(f2.age)
                FROM friend f2
                WHERE f1.state = f2.state );
```

- Multiple columns are supported:
  - `WHERE (uppercol1, uppercol2) IN (SELECT col1, col2 FROM subtable)`
- Subqueries can also be used for DELETE, INSERT & UPDATE
- SELECT INTO creates a new table with the result set

Indexing

- The following indexing algorithms are supported:
  - B-tree (default)
  - R-tree
  - Hash, and
  - GiST

Write-Ahead Logging (WAL)

- Standard technique for transactional logging:
  - Changes in data files can only be written after the changes have been logged and the log has been written to disc
  - No need to flush the data files after each COMMIT

- Advantages:
  - Reduces the number of writes against disk
  - One sync against the log file instead of potentially many against the data files
  - The log file is written sequentially
  - Ensures consistency of the data files
  - Enables online backup and point-in-time recovery

New features in PostgreSQL 8.0

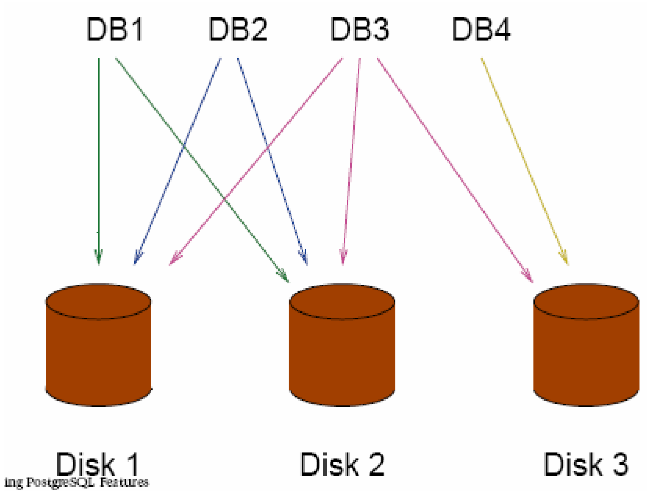
- 7.0 released in mid-2000, so 4.5 years of development
  - 8 months of development of new features compared to 7.x
  - 17 pages of changes
  - 5 months of beta testing
  - Goal: Make PostgreSQL ´enterprise ready´

- Most important new features:
  - Tablespaces: spread data files across disks
  - Savepoints
  - Point-in-time Recovery (PITR)
  - Perl integrated in the server
  - Native support for Windows (~10 man years)

Tablespaces

- Pre 8.0 required symlinking in order to place datafiles in other places than the default

- Tablespaces let us specifiy where to place:
  - Database
  - Schema
  - Tables
  - Indices



ing PostgreSQL Features

- Advantages:
  - Granular to object-level
  - Improves perfomance and control over disc usage
  - Better flexibility to add space when a disk fills up

```
CREATE TABLESPACE fastspace LOCATION ’/disk1/pgsql/data’;
CREATE TABLE foo(i int) TABLESPACE fastspace;
```

Savepoints

- Savepoints gives us the ability to handle error conditions within a transaction in a gracious manner without bailing out of it
  - Changes before a savepoint are implemented even if a rollback occurs later in the transaction
  - Rollbacks within the transaction is not visible outside the transaction

One transaction

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00 WHERE name = ’Alice’;
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00 WHERE name = ’Bob’;
```

- Oops ... use the account of Charlie instead!

```
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00 WHERE name = ’Charlie’;
COMMIT;
```

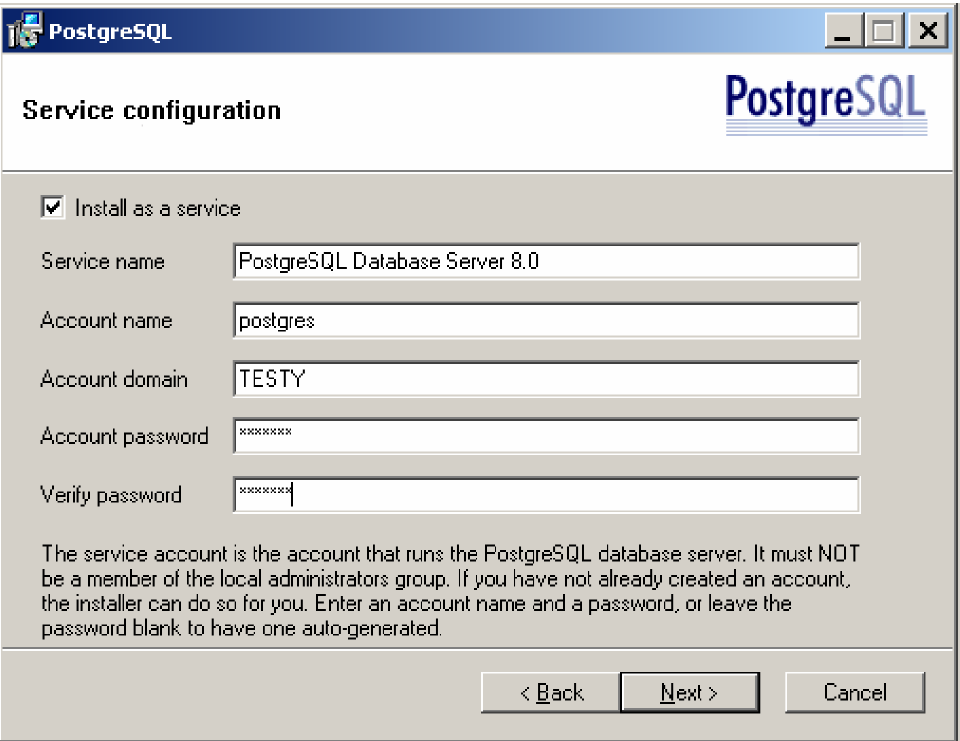
Point-In-Time Recovery

- Prior to PostgreSQL 8, the only way of recovering from a disc crash was to:
  - Recreate data from backup
  - Use replication

- Point-in-time recovery supports continuous backup of the serveren:
  - The Write-Ahead-Log describe all changes; by backup up this, we can fast forward and rewind the database state to a given point in time
  - PITR is based on continous transmission of the WAL to a failover machine, based one a freely chosed archival technique
  - Enable recover from the time of crash, or arbitrary chosen point in time

Native support for Windows #1

- Windows was formerly supported through the use of Cygwin; PostgreSQL 8 includes native support on 2000, XP and 2003.
- Can run as a service
- New, nifty installer:

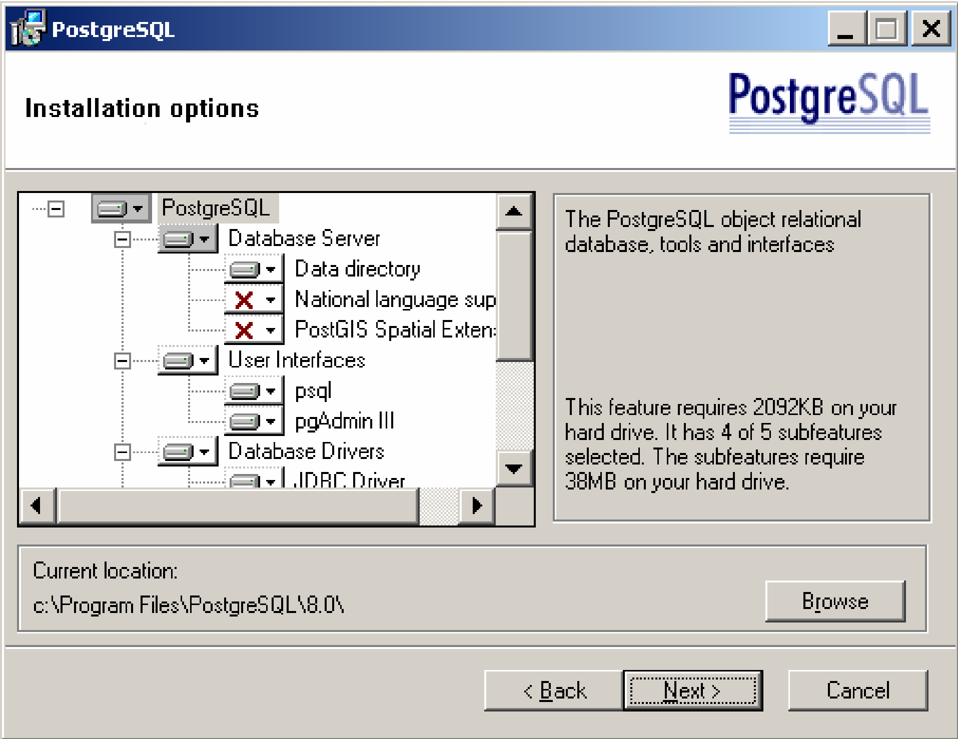


Service configuration

☑ Install as a service

Service name	PostgreSQL Database Server 8.0
Account name	postgres
Account domain	TESTY
Account password	*******
Verify password	*******

The service account is the account that runs the PostgreSQL database server. It must NOT be a member of the local administrators group. If you have not already created an account, the installer can do so for you. Enter an account name and a password, or leave the password blank to have one auto-generated.



Installation options

- PostgreSQL
  - Database Server
    - Data directory
    - ✗ National language sup
    - ✗ PostGIS Spatial Exten
  - User Interfaces
    - psql
    - pgAdmin III
  - Database Drivers
    - JDBC Driver

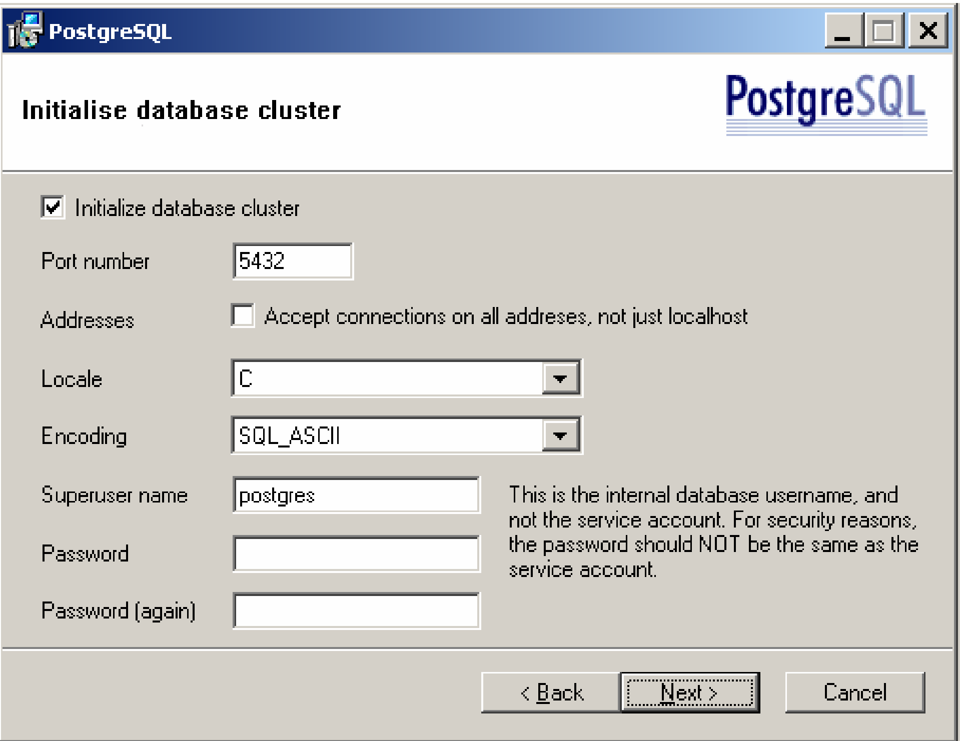
The PostgreSQL object relational database, tools and interfaces

This feature requires 2092KB on your hard drive. It has 4 of 5 subfeatures selected. The subfeatures require 38MB on your hard drive.

Current location:
c:\Program Files\PostgreSQL\8.0\

Native support for Windows #2

- Includes the following add-ons:
  - Npgsql
  - JDBC
  - psqlODBC
  - pgAdmin III



Initialise database cluster

☑ Initialize database cluster

Port number: 5432

Addresses: □ Accept connections on all addreses, not just localhost

Locale: C

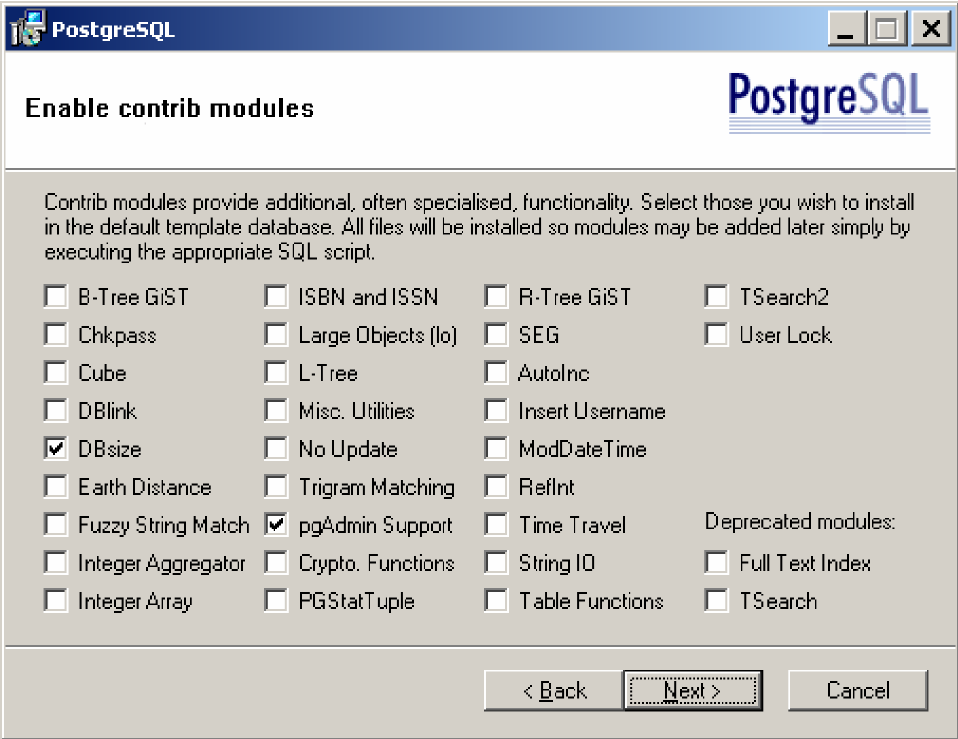
Encoding: SQL_ASCII

Superuser name: postgres

Password:

Password (again):

This is the internal database username, and not the service account. For security reasons, the password should NOT be the same as the service account.



Enable contrib modules

Contrib modules provide additional, often specialised, functionality. Select those you wish to install in the default template database. All files will be installed so modules may be added later simply by executing the appropriate SQL script.

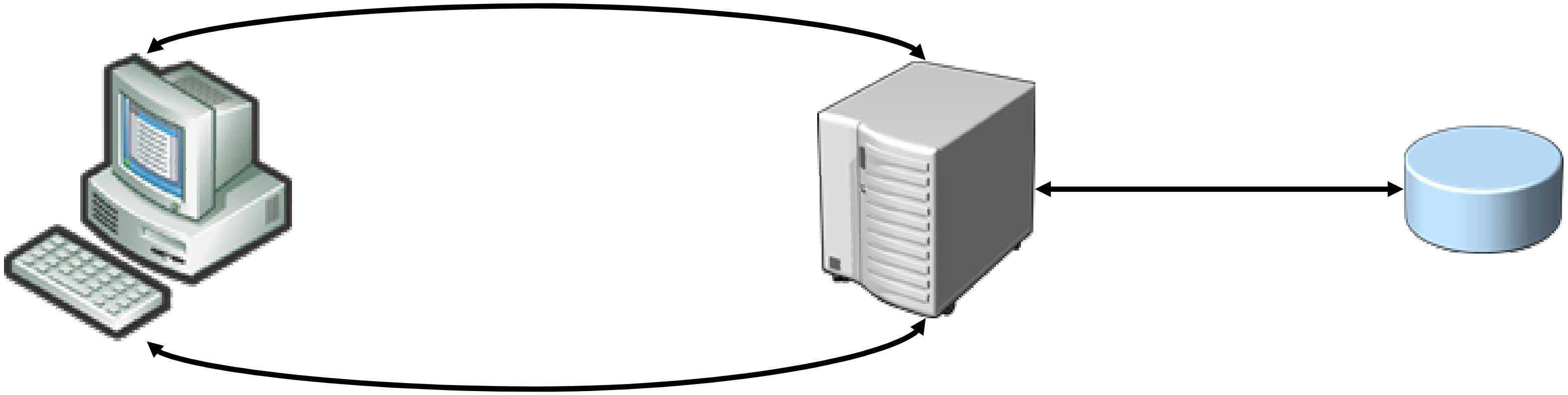
□ B-Tree GiST
□ Chkpass
□ Cube
□ DBlink
☑ DBsize
□ Earth Distance
□ Fuzzy String Match
□ Integer Aggregator
□ Integer Array
□ ISBN and ISSN
□ Large Objects (lo)
□ L-Tree
□ Misc. Utilities
□ No Update
□ Trigram Matching
☑ pgAdmin Support
□ Crypto. Functions
□ PGStatTuple
□ R-Tree GiST
□ SEG
□ AutoInc
□ Insert Username
□ ModDateTime
□ RefInt
□ Time Travel
□ String IO
□ Table Functions
□ TSearch2
□ User Lock

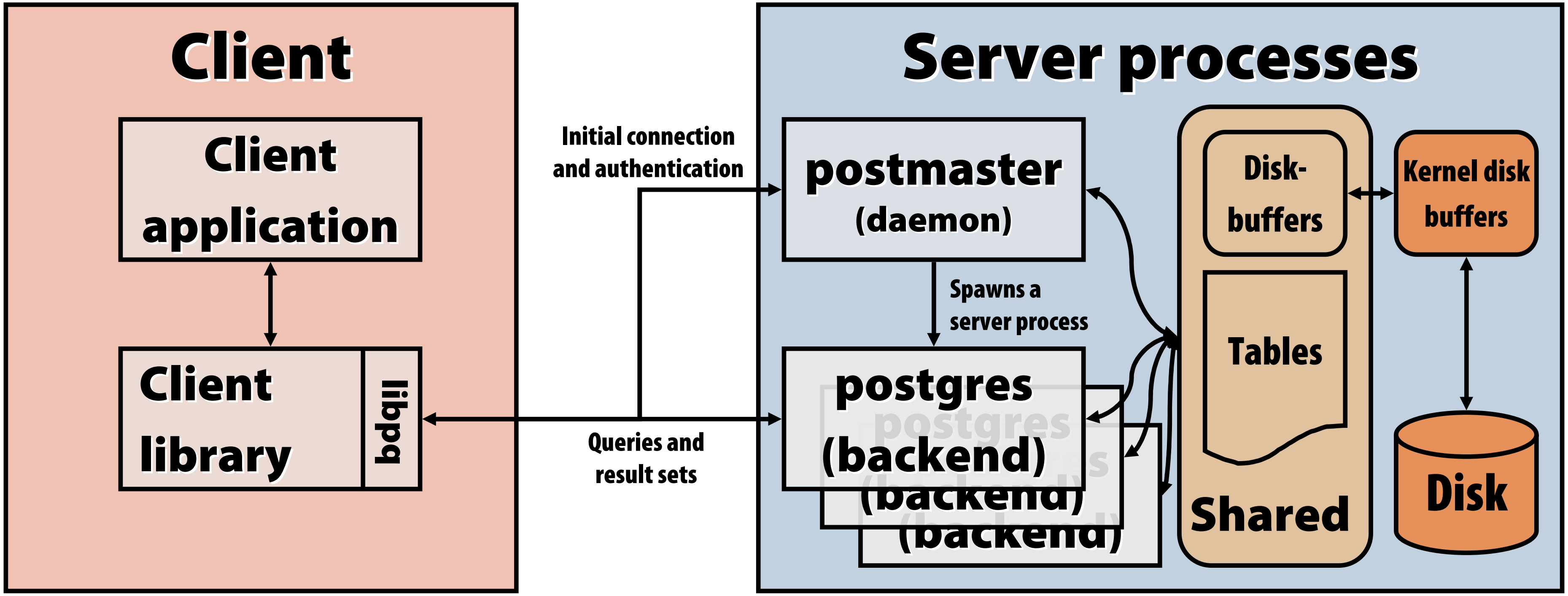
Deprecated modules:

□ Full Text Index
□ TSearch

PostgreSQL behind the scenes

Overall architecture





Client

Client application

Client library

libpq

Initial connection and authentication

Queries and result sets

Server processes

postmaster (daemon)

Spawns a server process

postgres (backend)

postgres (backend)

postgres (backend)

Disk-buffers

Tables

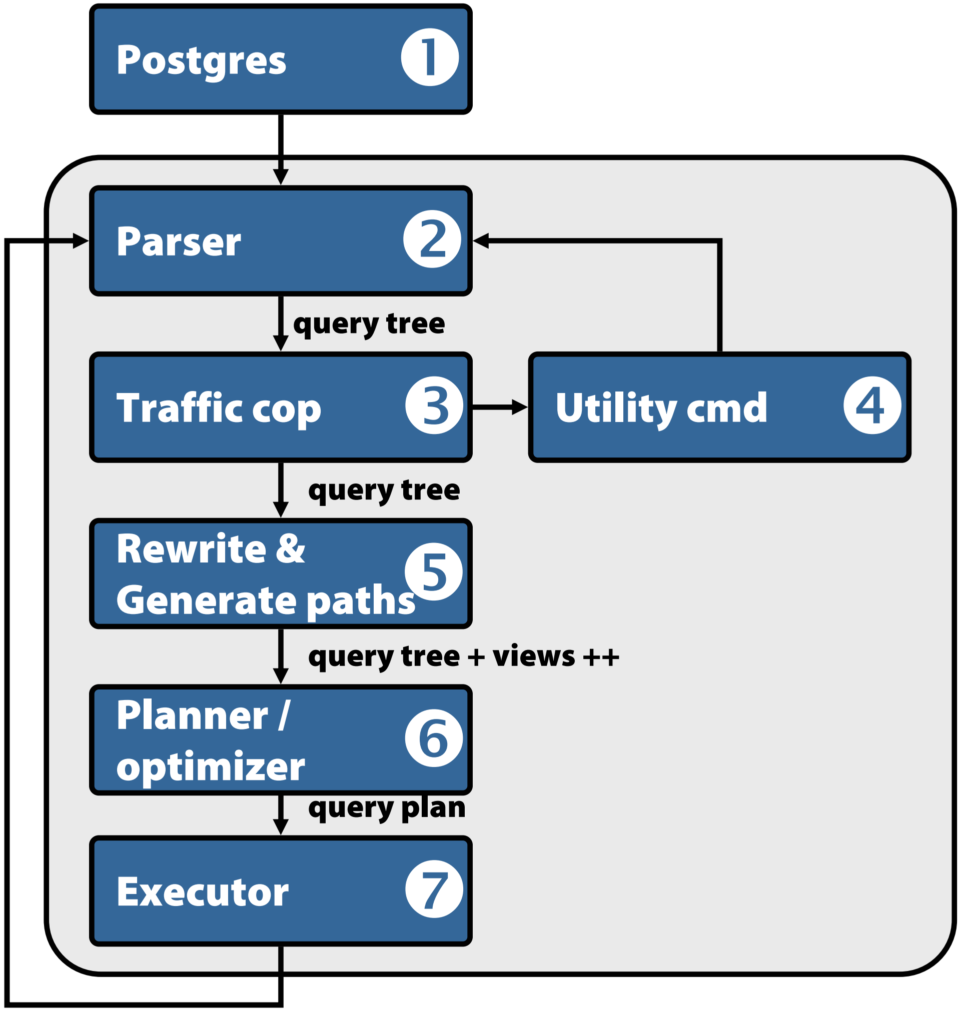
Shared

Kernel disk buffers

Disk

```
23961  Ss     0:05.64 /usr/local/bin/postmaster                 (postgres)
23963  S      0:01.13 postmaster: stats buffer process          (postgres)
23966  S      0:03.24 postmaster: stats collector process       (postgres)
36324  I      0:00.43 postmaster: oddbjorn testdb [local] idle (postgres)
36428  I      0:00.23 postmaster: oddbjorn testdb [local] idle (postgres)
```

What happens during a query?



1. The query arrives by a socket; put into a string
2. Lex/yacc chops up the string, and the type of query is identified
3. Judge whether this is a complex query or the use of a utility command
4. Call respective utility command and return.
5. Apply rules, views and so on
6. Choose optimal plan based upon cost of query tree paths; send it to the executor
7. Execute query, fetch data, sort, perform joins, qualify data and return the result set

Tuning: EXPLAIN

- PostgreSQL creates a *query plan* for each query
- EXPLAIN is an important tool to understand and tune the query plans:

```
testdb=> EXPLAIN SELECT * FROM syslog;
                         QUERY PLAN
------------------------------------------------------------
 Seq Scan on syslog  (cost=0.00..20.00 rows=1000 width=104)
(1 row)
```

1. Estimated startup cost
2. Estimated total cost for all rows
3. Estimated number of rows in the result set
4. Width in number of bytes per row in result set

Kostnadene er målt i antall pages som må hentes fra disk. CPU-kostnadene konverteres til disk-enheter.

- (Much more information): Efficient SQL, OSCON 2003
  - http://www.gtsm.com/oscon2003/toc.html

Tuning: ANALYZE

```
testdb=> ANALYZE VERBOSE syslog;
INFO:  analyzing "public.syslog"
INFO:  "syslog": 3614 pages, 3000 rows sampled, 26243 estimated total rows
ANALYZE
testdb=> EXPLAIN SELECT * from syslog;
                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on syslog  (cost=0.00..3876.43 rows=26243 width=132)
(1 row)
```

- The quality of the plan is dependent upon:
  - The knowledge PostgreSQL has about tables, indices ++
  - combined with the parameter settings in postgresql.conf

Tuning: VACUUM

- VACUUM must be run periodically to:
  1. Free space used by updated or deleted rows
  2. Update the statistics used to create query plans
  3. Protect against loss of data due to wraparound of the transaction ID

- Can be run in parallel with ordinary use of the database

- pg_autovacuum
  - contrib-client monitoring all the databases in an instance of PostgreSQL
  - Use the collection of statistics to monitor, UPDATE- and DELETE-activity
  - Automagically starts VACUUMing when defined thresholds are met

Directory structure

/usr/local/pgsql/data

PG_VERSION	eg."8.0"
postgresql.conf	main config file
postmaster.opts	options
postmaster.pid	PID
pg_hba.conf	access control
pg_ident.conf	mapping between identies
base/	the database files
global/	
pg_log/	application logs
pg_clog/	transaction logs
pg_xlog/	WAL logs
pg_tblspc/	tablespaces

postgresql.conf: Connection Settings

```
tcpip_socket = false
max_connections = 20
#superuser_reserved_connections = 2
port = 5432
[..]
```

postgresql.conf: Resource Settings

```
# - Memory -

shared_buffers = 1000           # min 16, at least max_connections*2, 8KB each
#sort_mem = 1024                # min 64, size in KB
#vacuum_mem = 8192              # min 1024, size in KB

# - Free Space Map -

#max_fsm_pages = 20000          # min max_fsm_relations*16, 6 bytes each
#max_fsm_relations = 1000       # min 100, ~50 bytes each

# - Kernel Resource Usage -

#max_files_per_process = 1000   # min 25
#preload_libraries = ''
```

postgresql.conf: Miscellaneous

- Security & Authentication
- Write Ahead Log
  - Settings
  - Checkpoints
- Query Tuning
  - Planner Method Enabling
  - Planner Cost Constants
  - Genetic Query Optimizer
- Error Reporting and Logging
  - syslog
  - When to log
  - What to log
- Runtime Statistics
  - Statistics Monitoring
  - Query/Index Statistics Collector
- Client Connection Defaults
  - Statement Behaviour
  - Locale and Formatting
- Lock Management
- Version / Platform Compatibility

Access control: pg_hba.conf

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of seven forms:
#
# local      DATABASE  USER  METHOD  [OPTION]
# host       DATABASE  USER  IP-ADDRESS  IP-MASK   METHOD  [OPTION]
# hostssl    DATABASE  USER  IP-ADDRESS  IP-MASK   METHOD  [OPTION]
# hostnossl  DATABASE  USER  IP-ADDRESS  IP-MASK   METHOD  [OPTION]
# host       DATABASE  USER  IP-ADDRESS/CIDR-MASK  METHOD  [OPTION]
# hostssl    DATABASE  USER  IP-ADDRESS/CIDR-MASK  METHOD  [OPTION]
# hostnossl  DATABASE  USER  IP-ADDRESS/CIDR-MASK  METHOD  [OPTION]
#
# [..]
# METHOD can be "trust", "reject","md5", "crypt",
# "password", "krb4", "krb5", "ident", or "pam".
#
# If you want to allow non-local connections, you need to add more
# "host" records.  Also, remember TCP/IP connections are only enabled
# if you enable "tcpip_socket" in postgresql.conf.

# TYPE  DATABASE    USER        IP-ADDRESS        IP-MASK           METHOD
local   all         all                                             trust
host    all         all         127.0.0.1         255.255.255.255   trust
host    all         all         192.168.1.2       255.255.255.255   trust
```

Check of status: pg_controldata

```
home ~> pg_controldata /usr/local/pgsql/data
pg_control version number:            72
Catalog version number:               200310211
Database cluster state:               in production
pg_control last modified:             Sun Jan 30 17:08:32 2005
Current log file ID:                  0
Next log file segment:                57
Latest checkpoint location:           0/3879ABE4
Prior checkpoint location:            0/3879ABA4
Latest checkpoint's REDO location:    0/3879ABE4
Latest checkpoint's UNDO location:    0/0
Latest checkpoint's StartUpID:        78
Latest checkpoint's NextXID:          886791
Latest checkpoint's NextOID:          5065687
Time of latest checkpoint:            Thu Jan 27 16:19:38 2005
Database block size:                  8192
Blocks per segment of large relation: 131072
Maximum length of identifiers:        64
Maximum number of function arguments: 32
Date/time type storage:               floating-point numbers
Maximum length of locale name:        128
LC_COLLATE:                           C
LC_CTYPE:                             C
```

System Catalog + Information schema

- The System Catalog: pg_catalog
  - The system catalog is a schema containing PostgreSQL-specific tables and views describing available tables, data types, functions and operators

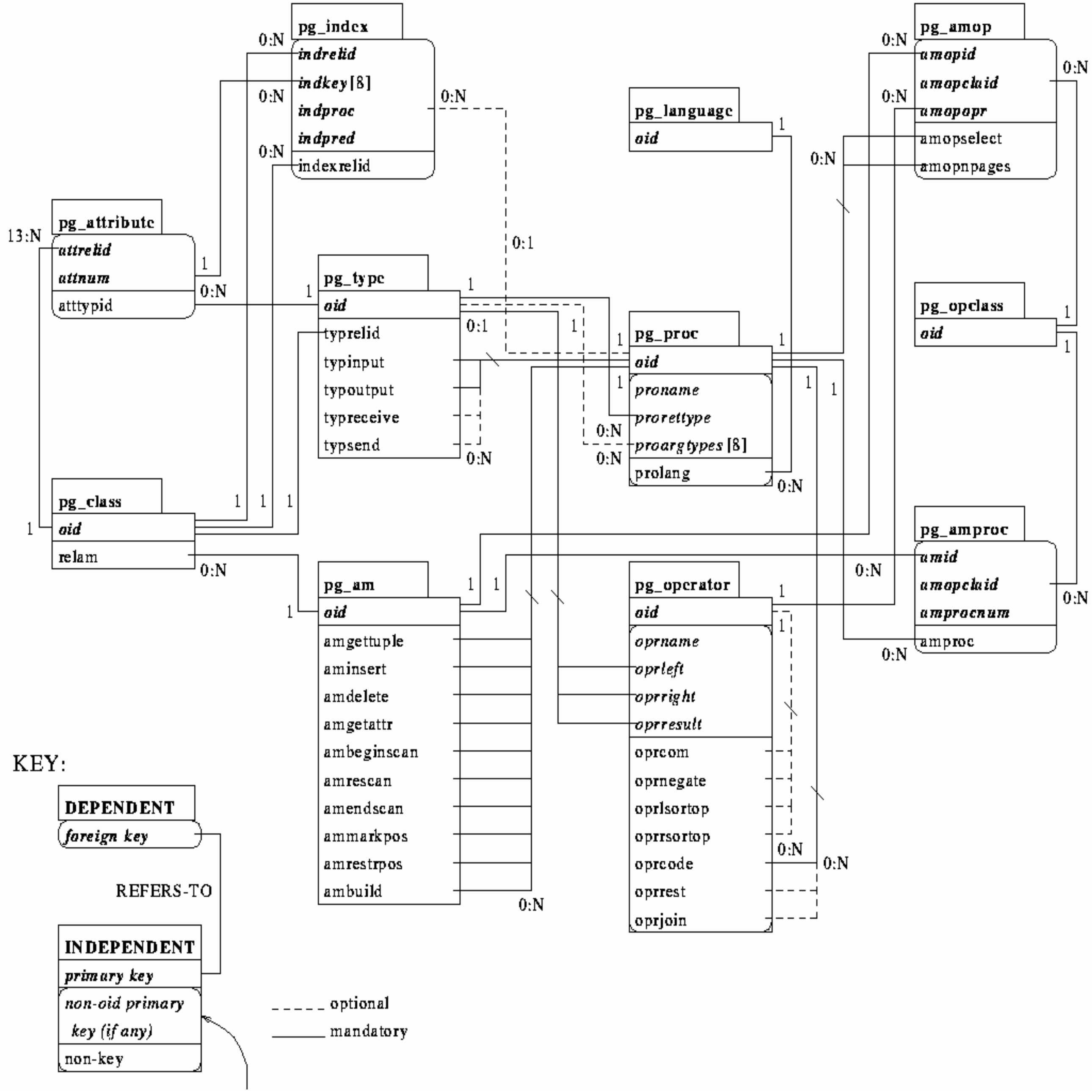
- The Information Schema: information_schema
  - Automatically established in all databases as a subset of pg_catalog
  - Defined in the SQL standarden; stable and portable
  - Does not contain PostgreSQL-specific information

psql: Listing the System Catalog

```
test=> \dS
 List of relations
   Schema    |            Name            |  Type    | Owner
------------+--------------------------+---------+-------
 pg_catalog | pg_aggregate             | table    | pgsql
 pg_catalog | pg_am                    | table    | pgsql
 pg_catalog | pg_amop                  | table    | pgsql
 pg_catalog | pg_amproc                | table    | pgsql
 pg_catalog | pg_attrdef               | table    | pgsql
 pg_catalog | pg_attribute             | table    | pgsql
 pg_catalog | pg_cast                  | table    | pgsql
 pg_catalog | pg_class                 | table    | pgsql
 pg_catalog | pg_constraint            | table    | pgsql
 pg_catalog | pg_conversion            | table    | pgsql
 pg_catalog | pg_database              | table    | pgsql
 pg_catalog | pg_depend                | table    | pgsql
 pg_catalog | pg_description           | table    | pgsql
 pg_catalog | pg_group                 | table    | pgsql
 pg_catalog | pg_index                 | table    | pgsql
 pg_catalog | pg_indexes               | view     | pgsql
 pg_catalog | pg_inherits              | table    | pgsql
 [..]
```

I tabellene i systemkatalogen, lagrer PostgreSQL metadata; f.eks. informasjon om databaser, tabeller, views, brukere og så videre. Ved CREATE DATABASE blir f.eks. pg_database oppdatert, samt databasen skrevet til disk.

ER diagram of the pg_catalog



pg_index
- *indrelid*
- *indkey*[8]
- *indproc*
- *indpred*
- indexrelid

pg_attribute
- *attrelid*
- *attnum*
- atttypid

pg_type
- *oid*
- typrelid
- typinput
- typoutput
- typreceive
- typsend

pg_class
- *oid*
- relam

pg_am
- *oid*
- amgettuple
- aminsert
- amdelete
- amgetattr
- ambeginscan
- amrescan
- amendscan
- ammarkpos
- amrestrpos
- ambuild

pg_language
- *oid*

pg_proc
- *oid*
- *proname*
- *prorettype*
- *proargtypes* [8]
- prolang

pg_operator
- *oid*
- *oprname*
- *oprleft*
- *oprright*
- *oprresult*
- oprcom
- oprnegate
- oprlsortop
- oprrsortop
- oprcode
- oprrest
- oprjoin

pg_amop
- *amopid*
- *amopclaid*
- *amopopr*
- amopselect
- amopnpages

pg_opclass
- *oid*

pg_amproc
- *amid*
- *amopclaid*
- *amprocnum*
- amproc

KEY:

DEPENDENT
- *foreign key*

REFERS-TO

INDEPENDENT
- *primary key*
- *non-oid primary key (if any)*
- non-key

- - - - - optional

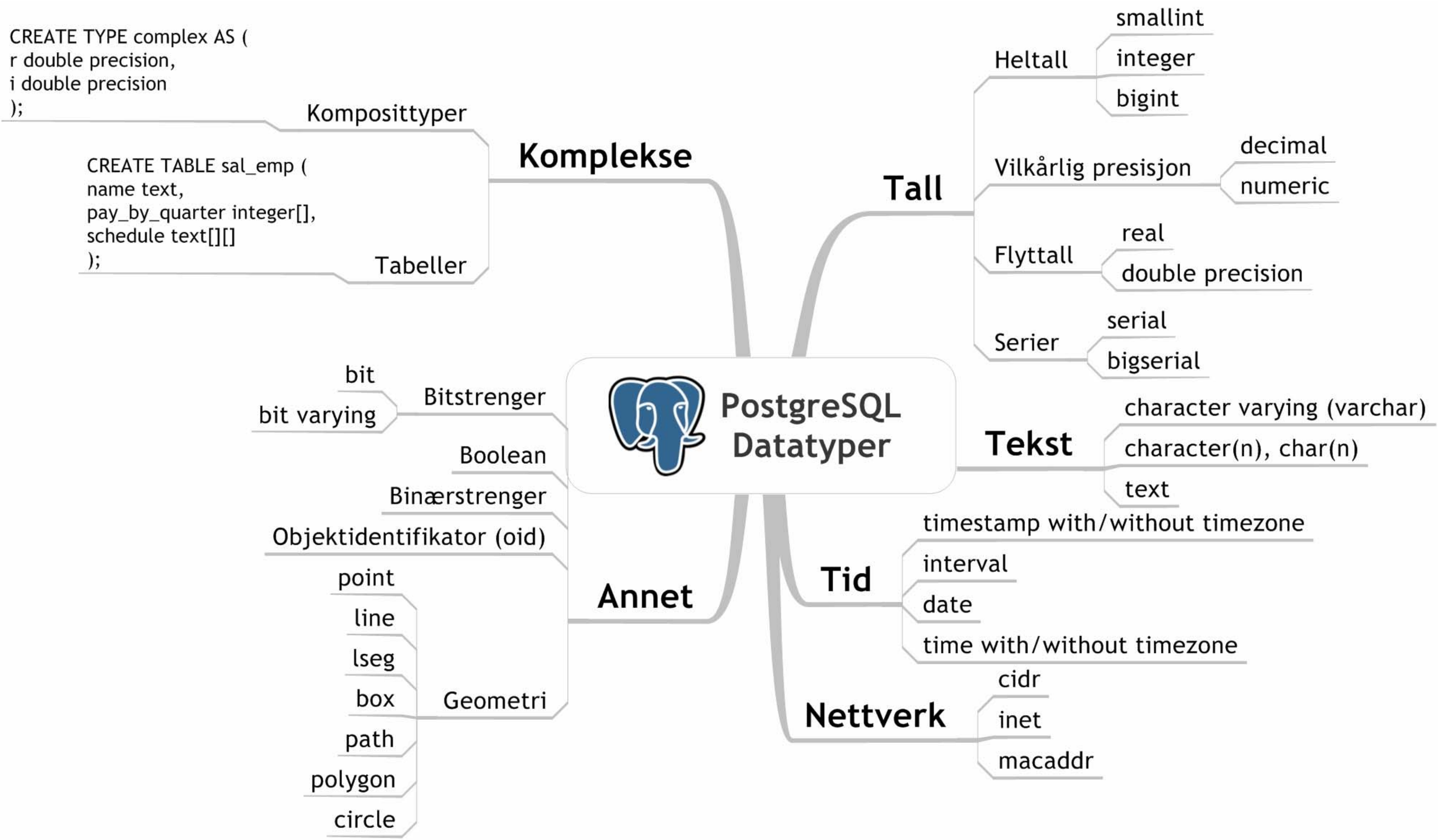
——— mandatory

Available data types: \dT+ in psql

```
                                           List of data types
   Schema   |            Name             |  Internal name   | Size |                              Description
------------+-----------------------------+------------------+------+------------------------------------------------------------------------
 pg_catalog | "SET"                       | SET              | var  | set of tuples
 pg_catalog | "any"                       | any              | 4    |
 pg_catalog | "char"                      | char             | 1    | single character
 pg_catalog | "path"                      | path             | var  | geometric path '(pt1,...)'
 pg_catalog | "trigger"                   | trigger          | 4    |
 pg_catalog | "unknown"                   | unknown          | var  |
 pg_catalog | abstime                     | abstime          | 4    | absolute, limited-range date and time (Unix system time)
 pg_catalog | aclitem                     | aclitem          | 12   | access control list
 pg_catalog | anyarray                    | anyarray         | var  |
 pg_catalog | anyelement                  | anyelement       | 4    |
 pg_catalog | bigint                      | int8             | 8    | ~18 digit integer, 8-byte storage
 pg_catalog | bit                         | bit              | var  | fixed-length bit string
 pg_catalog | bit varying                 | varbit           | var  | variable-length bit string
 pg_catalog | boolean                     | bool             | 1    | boolean, 'true'/'false'
 pg_catalog | box                         | box              | 32   | geometric box '(lower left,upper right)'
 pg_catalog | bytea                       | bytea            | var  | variable-length string, binary values escaped
 pg_catalog | character                   | bpchar           | var  | char(length), blank-padded string, fixed storage length
 pg_catalog | character varying           | varchar          | var  | varchar(length), non-blank-padded string, variable storage length
 pg_catalog | cid                         | cid              | 4    | command identifier type, sequence in transaction id
 pg_catalog | cidr                        | cidr             | var  | network IP address/netmask, network address
 pg_catalog | circle                      | circle           | 24   | geometric circle '(center,radius)'
 pg_catalog | cstring                     | cstring          | var  |
 pg_catalog | date                        | date             | 4    | ANSI SQL date
 pg_catalog | double precision            | float8           | 8    | double-precision floating point number, 8-byte storage
 pg_catalog | inet                        | inet             | var  | IP address/netmask, host address, netmask optional
 pg_catalog | int2vector                  | int2vector       | 64   | array of 32 int2 integers, used in system tables
 pg_catalog | integer                     | int4             | 4    | -2 billion to 2 billion integer, 4-byte storage
 pg_catalog | internal                    | internal         | 4    |
 pg_catalog | interval                    | interval         | 12   | @ <number> <units>, time interval
 pg_catalog | language_handler            | language_handler | 4    |
 pg_catalog | line                        | line             | 32   | geometric line (not implemented)'
 pg_catalog | lseg                        | lseg             | 32   | geometric line segment '(pt1,pt2)'
 pg_catalog | macaddr                     | macaddr          | 6    | XX:XX:XX:XX:XX:XX, MAC address
 pg_catalog | money                       | money            | 4    | monetary amounts, $d,ddd.cc
 pg_catalog | name                        | name             | 64   | 63-character type for storing system identifiers
 pg_catalog | numeric                     | numeric          | var  | numeric(precision, decimal), arbitrary precision number
 pg_catalog | oid                         | oid              | 4    | object identifier(oid), maximum 4 billion
 pg_catalog | oidvector                   | oidvector        | 128  | array of 32 oids, used in system tables
 pg_catalog | opaque                      | opaque           | 4    |
 pg_catalog | point                       | point            | 16   | geometric point '(x, y)'
 pg_catalog | polygon                     | polygon          | var  | geometric polygon '(pt1,...)'
 pg_catalog | real                        | float4           | 4    | single-precision floating point number, 4-byte storage
 pg_catalog | record                      | record           | 4    |
 pg_catalog | refcursor                   | refcursor        | var  |
 pg_catalog | regclass                    | regclass         | 4    | registered class
 pg_catalog | regoper                     | regoper          | 4    |
 pg_catalog | regoperator                 | regoperator      | 4    | registered operator (with args)
 pg_catalog | regproc                     | regproc          | 4    |
 pg_catalog | regprocedure                | regprocedure     | 4    | registered procedure (with args)
 pg_catalog | regtype                     | regtype          | 4    | registered type
 pg_catalog | reltime                     | reltime          | 4    |
 pg_catalog | smallint                    | int2             | 2    | -32 thousand to 32 thousand, 2-byte storage
 pg_catalog | smgr                        | smgr             | 2    |
 pg_catalog | text                        | text             | var  |
 pg_catalog | tid                         | tid              | 6    |
 pg_catalog | time with time zone         | timetz           | 12   |
 pg_catalog | time without time zone      | time             | 8    | hh:mm:ss, ANSI SQL time
 pg_catalog | timestamp with time zone    | timestamptz      | 8    |
 pg_catalog | timestamp without time zone | timestamp        | 8    | date and time
 pg_catalog | tinterval                   | tinterval        | 12   |
 pg_catalog | void                        | void             | 4    |
 pg_catalog | xid                         | xid              | 4    | transaction id
(62 rows)
```

- Operations against columns of the same data type gives consistent results, and are usually the fastest
- Proper use of daat types implies format validation of the data, and rejection of data outside the scope of the data type
- Proper use of data types give the most efficient storage of dataata

Mindmap of the built-in data types (not translated yet)



PostgreSQL Datatyper

- Komplekse
  - Komposittyper

```
CREATE TYPE complex AS (
r double precision,
i double precision
);
```

  - Tabeller

```
CREATE TABLE sal_emp (
name text,
pay_by_quarter integer[],
schedule text[][]
);
```

- Tall
  - Heltall
    - smallint
    - integer
    - bigint
  - Vilkårlig presisjon
    - decimal
    - numeric
  - Flyttall
    - real
    - double precision
  - Serier
    - serial
    - bigserial
- Tekst
  - character varying (varchar)
  - character(n), char(n)
  - text
- Tid
  - timestamp with/without timezone
  - interval
  - date
  - time with/without timezone
- Nettverk
  - cidr
  - inet
  - macaddr
- Annet
  - Bitstrenger
    - bit
    - bit varying
  - Boolean
  - Binærstrenger
  - Objektidentifikator (oid)
  - Geometri
    - point
    - line
    - lseg
    - box
    - path
    - polygon
    - circle

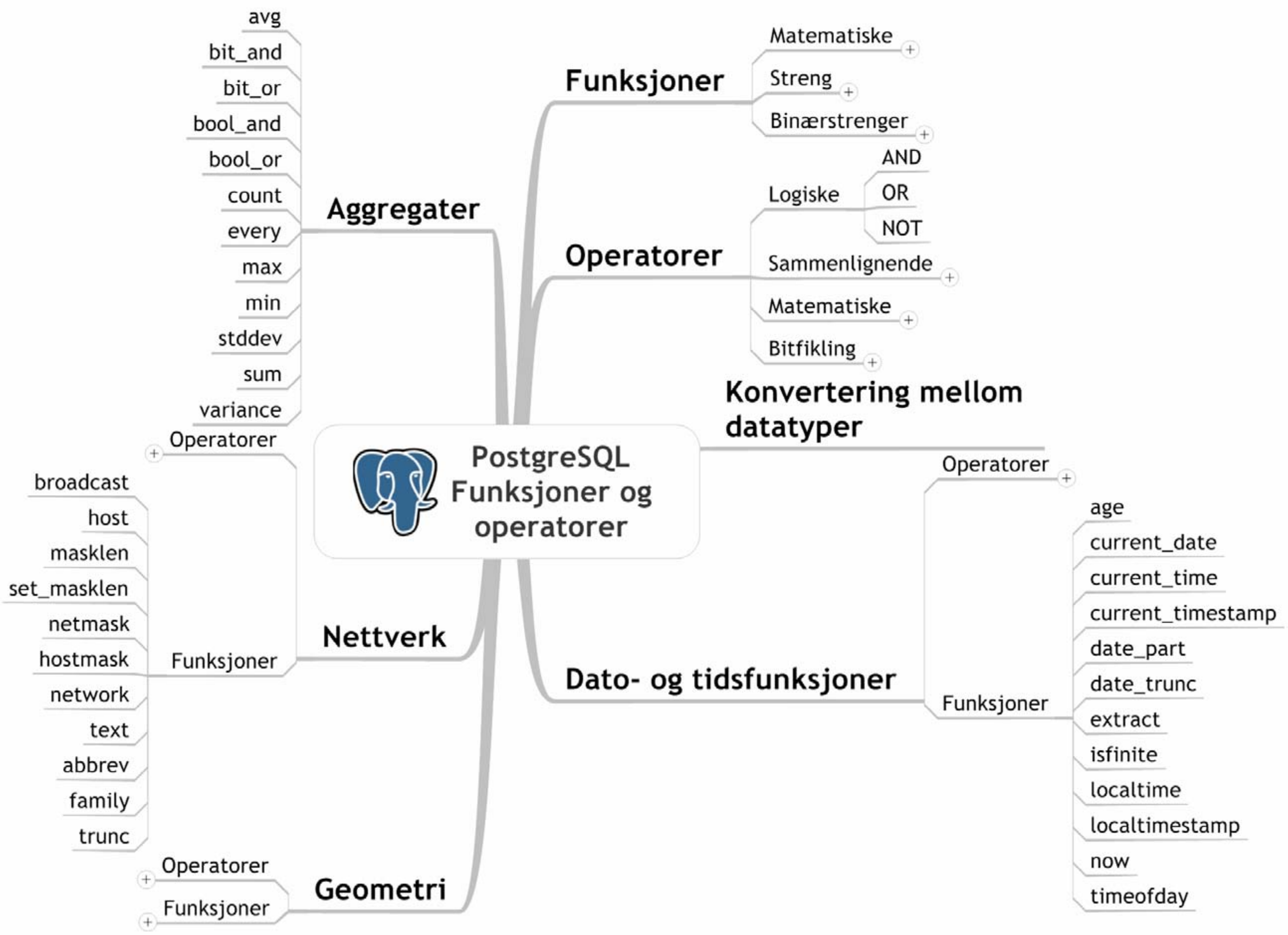
Network data types

- Three data types:
  - inet - host or network mask, eg. 10.0.0.1
  - cidr - network mask, eg. 10.0.0.0/8
  - macaddr - eg. '08:00:2b:01:02:03'

- Very useful when working with network information:

```
1. WHERE ’192.168.1.5’   < ’192.168.1.6’
2. WHERE ’192.168.1/24’ >> ’192.168.1.5’
3. WHERE ip << ’192.168.1.0/24’
4. trunc(macaddr)
```

Functions and operators (not translated yet)



PostgreSQL Funksjoner og operatorer

- Funksjoner
  - Matematiske
  - Streng
  - Binærstrenger
- Operatorer
  - Logiske
    - AND
    - OR
    - NOT
  - Sammenlignende
  - Matematiske
  - Bitfikling
- Konvertering mellom datatyper
- Dato- og tidsfunksjoner
  - Operatorer
  - Funksjoner
    - age
    - current_date
    - current_time
    - current_timestamp
    - date_part
    - date_trunc
    - extract
    - isfinite
    - localtime
    - localtimestamp
    - now
    - timeofday
- Aggregater
  - avg
  - bit_and
  - bit_or
  - bool_and
  - bool_or
  - count
  - every
  - max
  - min
  - stddev
  - sum
  - variance
- Nettverk
  - Operatorer
  - Funksjoner
    - broadcast
    - host
    - masklen
    - set_masklen
    - netmask
    - hostmask
    - network
    - text
    - abbrev
    - family
    - trunc
- Geometri
  - Operatorer
  - Funksjoner

Support for regular expressions

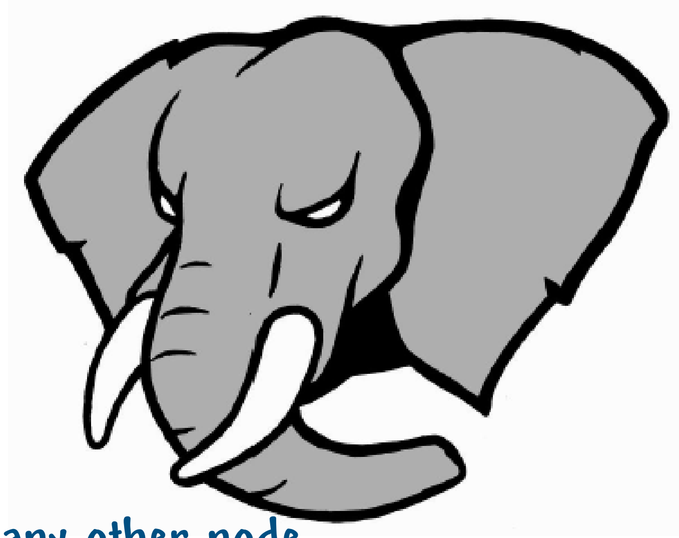
- Support for three kinds of pattern matching:
  - The SQL LIKE operator
  - The SQL99 SIMILAR TO-operator
  - POSIX-style regular expressions

- Example of the latter:
  - ’abc’ ~ ’abc’ *true*
  - ’abc’ ~ ’^a’ *true*
  - ’abc’ ~ ’(b|d)’ *true*
  - ’abc’ ~ ’^(b|c)’ *false*

Replication solutions

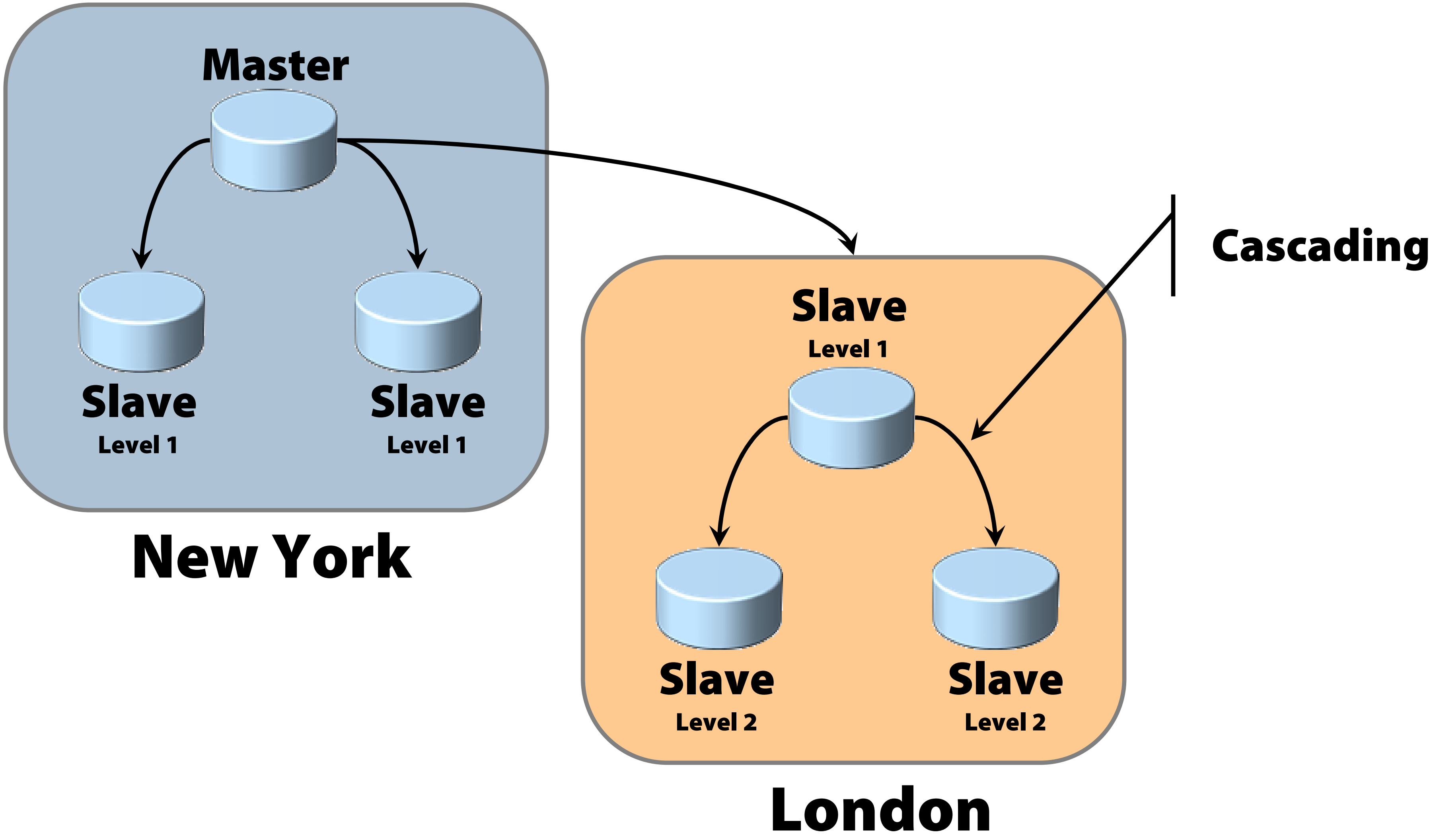
Slony-I

- "Master to multiple slaves" replication
  - Developed by Jan Wieck
  - Slony is Russian plural for elephant
  - Arguably the coolest mascot
  - http://www.slony.info

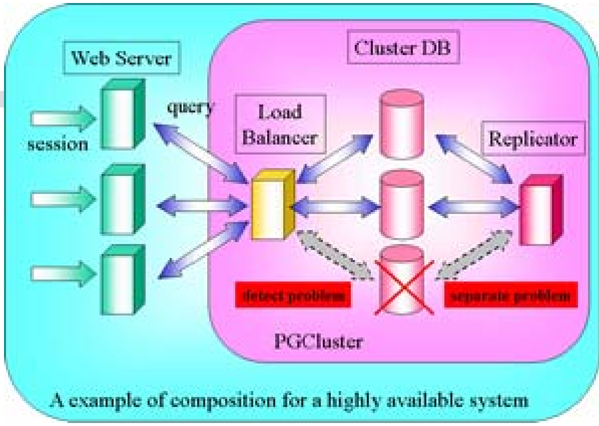


- Supports:
  - Establishing a replica while running
  - Asynchrounous replication
  - Any replica can take on the duties of any other node
    - Mechanism for promoting a slave to master if master dies
- Slony-2 is going to support multi-master replication
- *Introducing Slony & Building and Configuring Slony*
  - A. Elein Mustain
  - http://www.onlamp.com/lpt/a/{5328,5486}

Slony-I: Graphical description



Other replication solutions



- pgcluster
  - Synchronous replication including load balancing
  - http://pgcluster.projects.postgresql.org/
- pgpool
  - Connection-pool-server; implemented as a layer between clients and up to two PostgreSQL servers
  - Caches connections for improved performance
  - Automatic failover to secondary server if/when the primary fails
  - pgpool sends the transactions in parallel to each server
- eRServer
  - Trigger-based single-master/multi-slave asynchronous replication
  - No longer alive?
  - http://www.erserver.com/
- pgreplicator
  - "Store and forward" asynchronous replication
  - Two-way synchronization, differential replication
  - No longer developed?
  - http://pgreplicator.sourceforge.net

Programming PostgreSQL

Languages: Frontend versus backend

- Frontend:
  - Languages to access data from the 'outside', for example scripts or applications
- Backend:
  - Languages to extend the functionality of the database server
- Practically all the languages can be used in both roles.
- Classical balancing between functionality within the database or in the application.

Functions in other languages

- PostgreSQL supports user-defined functions in an assorted array of languages beyond SQL og C:
  - PL/pgSQL
  - PL/Tcl
  - PL/Perl
  - PL/Python
  - PL/PHP
  - PL/Java / pl-j

- PL = procedural languages
- Other languages can be defined by the user

- PostgreSQL does not care about the source code itself; it just transfer the procedure call to a handler which invoke the respective interpreter and receive the results back.

Use of procedural languages

```
createlang plperl dbname
```

```
CREATE FUNCTION perl_max (integer, integer) RETURNS integer AS $$
if ($_[0] > $_[1]) { return $_[0]; }
return $_[1];
$$ LANGUAGE plperl;
```

```
CREATE TABLE employee (
   name text,
   basesalary integer,
   bonus integer
);
```

```
CREATE FUNCTION empcomp(employee) RETURNS integer AS $$
   my ($emp) = @_;
   return $emp->{basesalary} + $emp->{bonus};
$$ LANGUAGE plperl;
```

```
SELECT name, empcomp(employee.*) FROM employee;
```

PL/pgSQL

- PL/pgSQL is a loadable procedural language
- Supports:
  - Defining functions and triggers
  - Control structures
  - Calculations
  - Reuses all data types, functions and operators available in PostgreSQL
  - Grouping of transactions in one procedure invocation, reducing client/server overhead

SQL-based functions: Example

```
CREATE FUNCTION tax(numeric)
   RETURNS numeric
   AS ‘SELECT ($1 * 0.06::numeric(8,2))::numeric(8,2);’
   LANGUAGE ‘sql’;

CREATE FUNCTION shipping(numeric)
   RETURNS numeric
   AS ‘SELECT CASE
        WHEN $1 < 2                THEN CAST(3.00 AS numeric(8,2))
        WHEN $1 >= 2 AND $1 < 4 THEN CAST(5.00 AS numeric(8,2))
        WHEN $1 >=4                THEN CAST(6.00 AS numeric(8,2))
   END;’
   LANGUAGE ‘sql’;

SELECT part_id, trim(name) AS name, cost, tax(cost), cost +
   tax(cost) AS subtotal, shipping(weight), cost + tax(cost) +
   shipping(weight) AS total
   FROM part
   ORDER BY part_id;
```

PL/pgSQL: Example

```
CREATE TABLE emp (empname text, salary int4,
                  last_date datetime, last_user name);

CREATE FUNCTION emp_stamp () RETURNS OPAQUE AS
   BEGIN
   -- Check that empname and salary are given
   IF NEW.empname ISNULL THEN
         RAISE EXCEPTION ''empname cannot be NULL value' ';
    END IF;
   IF NEW.salary ISNULL THEN
         RAISE EXCEPTION ''% cannot have NULL salary'', NEW.empname;
   END IF;
   -- Who works for us when she must pay for?
   IF NEW.salary < 0 THEN
         RAISE EXCEPTION ''% cannot have a negative salary'',
   NEW.empname;
   END IF;
   -- Remember who changed the payroll when
   NEW.last_date :=  ' 'now' ';
   NEW.last_user := getpgusername();
   RETURN NEW;
   END; '
LANGUAGE 'plpgsql';

CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
```

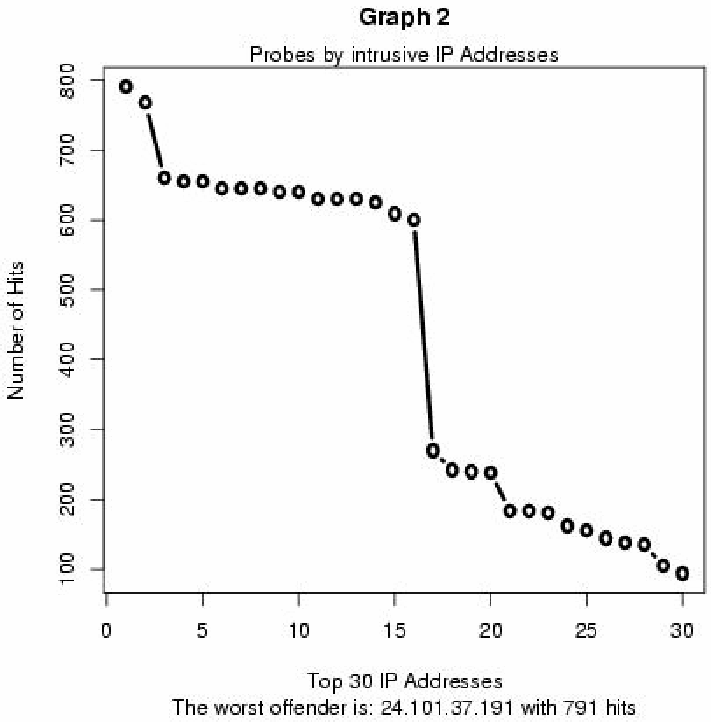
pl/R

- R is an integrated environment for manipulating, calulating and displaying data
  - Based upon AT&Ts S

- R includes:
  - efficient management and storage of data
  - operators for manipulating tables and matrices
  - large number of functions and tools to analyze data
  - tool to create high quality graphs, both for screen and print
  - a mature programming language to tie the above together

- PL/R is a loadable procedural language which enable functions and triggers in PostgreSQL to be expressed in R:
  - Written by Joe Conway
  - *How to Graph data in PostgreSQL* by Robert Bernier:
    - http://www.varlena.com/varlena/GeneralBits/Tidbits/ +
    - bernier/art_66/graphingWithR.html

pl/R: Plotting of firewall logs



```
BEGIN;

CREATE TEMPORARY TABLE
  mytemp(id serial, hit int, source_ip inet)
  ON COMMIT DROP;

INSERT INTO mytemp(hit,source_ip)
  SELECT count(*) AS counterhits, source_ip
  FROM firewall
  GROUP BY source_ip
  ORDER BY counterhits DESC;

CREATE OR REPLACE FUNCTION f_graph2() RETURNS text AS '
sql <- paste("SELECT id as x,hit as y FROM mytemp LIMIT 30",sep="");
str <- c(pg.spi.exec(sql));

mymain <- "Graph 2";
mysub <- paste("The worst offender is: ",str[1,3]," with ",str[1,2]," hits",sep="");
myxlab <- "Top 30 IP Addresses";
myylab <- "Number of Hits";

pdf(''/tmp/graph2.pdf'');
plot(str,type="b",main=mymain,sub=mysub,xlab=myxlab,ylab=myylab,lwd=3);
mtext("Probes by intrusive IP Addresses",side=3);
dev.off();

print(''DONE'');
' LANGUAGE plr;

-- now generating the graph
SELECT f_graph2();
COMMIT;
```

Other interfaces

- psqlODBC
  - This is the most common interface for Windows applications.
- pgjdbc
  - A JDBC interface.
- Npgsql
  - .Net interface for more recent Windows applications.
- libpqxx
  - A newer C++ interface.
- libpq++
  - An older C++ interface.
- pgperl
  - A Perl interface with an API similar to libpq.
- DBD-Pg
  - A Perl interface that uses the DBD-standard API.
- pgtclng
  - A newer version of the Tcl interface.
- pgtcl
  - The original version of the Tcl interface.
- PyGreSQL
  - A Python interface library.

Use of PostgreSQL from Perl

- DBI / DBD::Pg / DBD::PgPP (not libpq-based)

```
#!/usr/local/bin/perl –w

use DBI;

$dbh = DBI->connect('dbi:Pg:dbname=testdb;', ‘username', '');

$sth = $dbh->prepare(“SELECT id,news from news”);
$sth->execute;

while (@news = $sth->fetchrow) {
   $date    = $news[0];
   $article = $news[1];

   print(“$date:\t $article\n”);
}
```

Use of PostgreSQL from Python #1

- PygreSQL
  - The oldest and most tested
  - http://www.pygresql.org

- psycopg
  - Based upon libpq, with DB API-interface
  - Used a lot by Zope
  - Smart reuse of connections
  - http://initd.org/software/initd/psycopg

- and others (pyPgSQL, DB-API)

Use of PostgreSQL from Python #2

```
import psycopg

o = psycopg.connect('dbname=mydb user=fog')

c = o.cursor()
c.execute('SELECT * FROM addressbook WHERE name = %s', [‘Bob'])
data = c.fetchone()

print "Saving image of %s %s" % (data[0], data[1])
open(data[0]+".png", 'w').write(data[3])
```

Use of PostgreSQL from PHP

- http://www.php.net/manual/en/ref.pgsql.php

```
$conn = pg_connect("dbname=testdb");

if (!$conn) {
   print("Connection Failed.");
   exit;
}

$query = “SELECT posted_date,posted_time,news FROM news”;
$news  = pg_query($conn, $query);

echo "<table border=1>\n";

for($i = 0; $i < pg_num_rows($news); $i++) {
   echo "<tr>\n";
   echo "<td>” . pg_result($news, $i, 0) . "</td>\n";
   echo "<td>" . pg_result($news, $i, 1) . "</td>\n";
   echo "<td>" . pg_result($news, $i, 2) . "</td>\n";
   echo "</tr>";
}

echo "</table>";
```

ODBC & JDBC

- ODBC
  - http://odbc.postgresql.org/

- JDBC
  - Pure Java-implementation
  - Supports JDBC v3 + extensions
  - http://jdbc.postgresql.org/

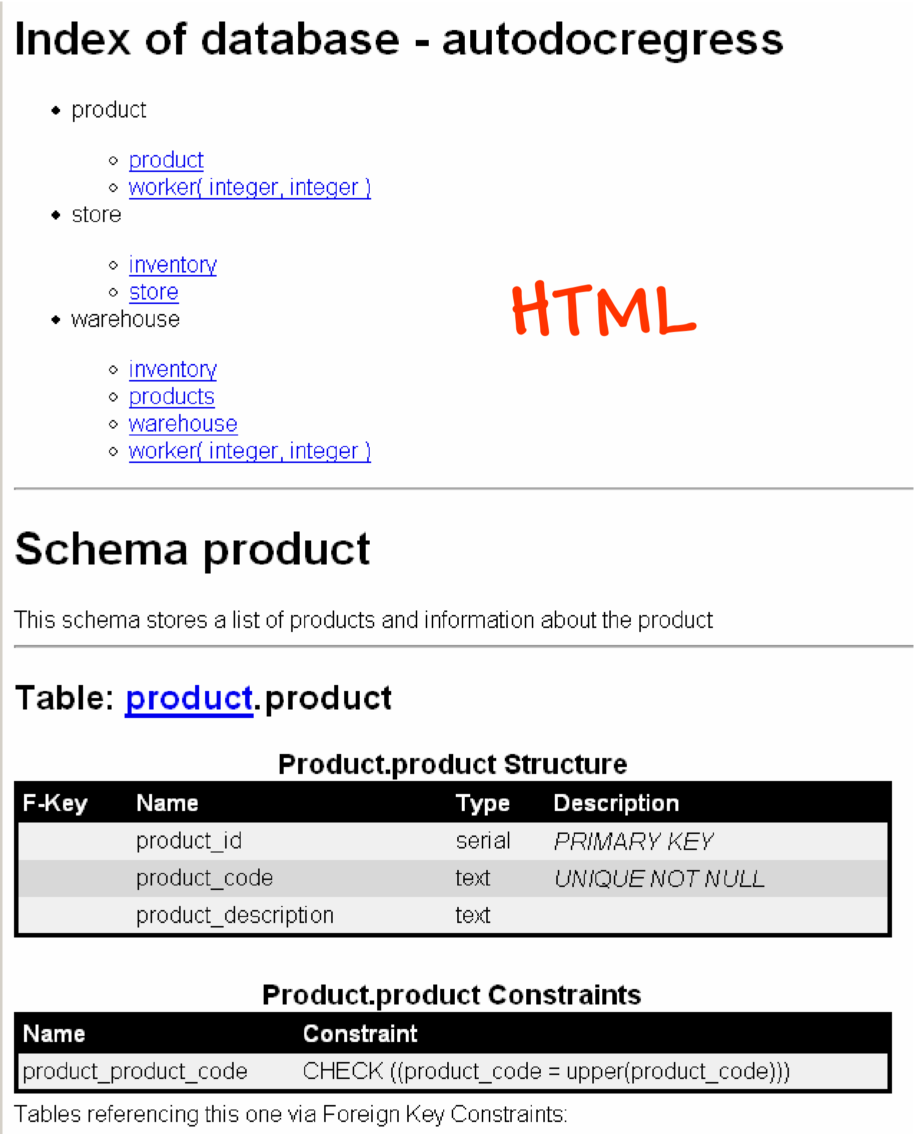
- Both available as FreeBSD-ports

Third party tools

Autodoc

- Tool to automagically document a database
- Template-based reporting to the following formats:
  - HTML
  - Dot
  - Dia
  - Docbook XML

Autodoc: Examples #1



Index of database - autodocregress

Schema product

This schema stores a list of products and information about the product

Table: product.product

Product.product Structure

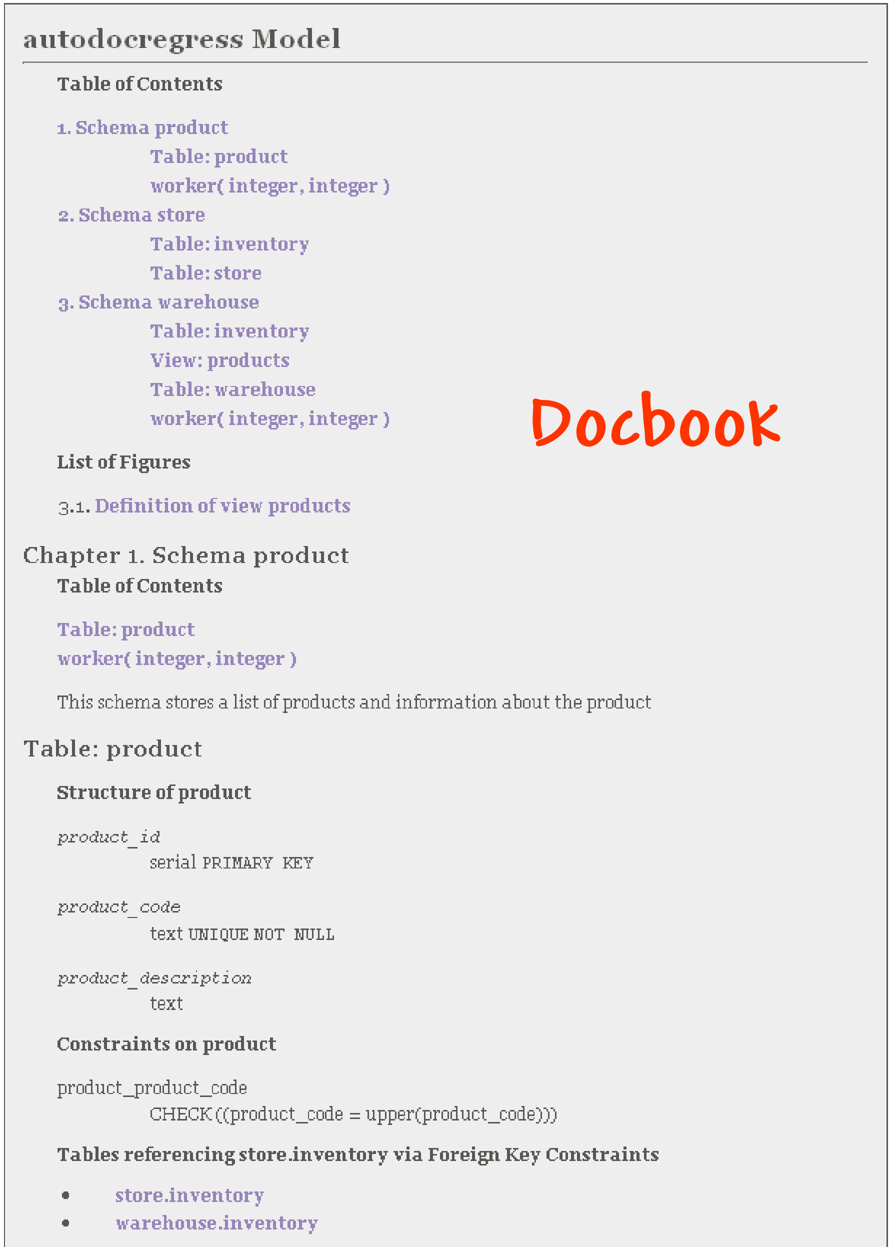
F-Key	Name	Type	Description
	product_id	serial	*PRIMARY KEY*
	product_code	text	*UNIQUE NOT NULL*
	product_description	text	

Product.product Constraints

Name	Constraint
product_product_code	CHECK ((product_code = upper(product_code)))

Tables referencing this one via Foreign Key Constraints:

HTML



autodocregress Model

Table of Contents

List of Figures

Chapter 1. Schema product

Table of Contents

This schema stores a list of products and information about the product

Table: product

Structure of product

product_id
serial PRIMARY KEY

product_code
text UNIQUE NOT NULL

product_description
text

Constraints on product

product_product_code
CHECK ((product_code = upper(product_code)))

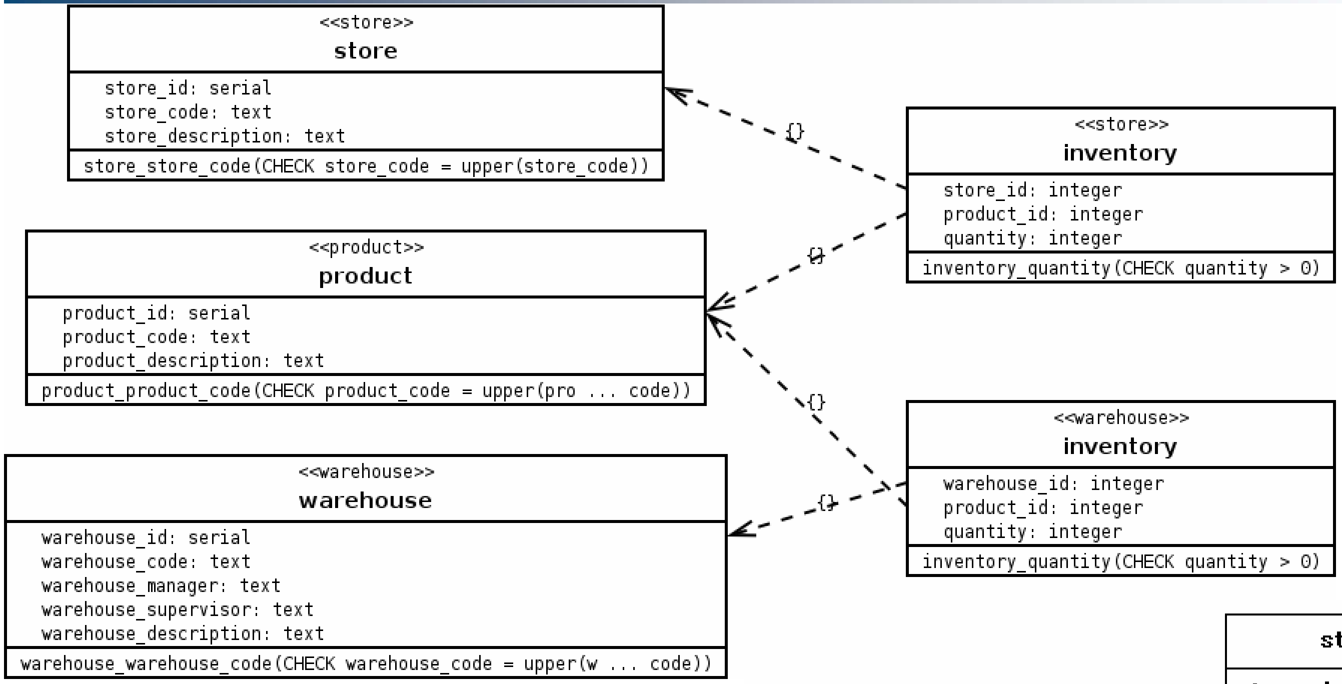
Tables referencing store.inventory via Foreign Key Constraints

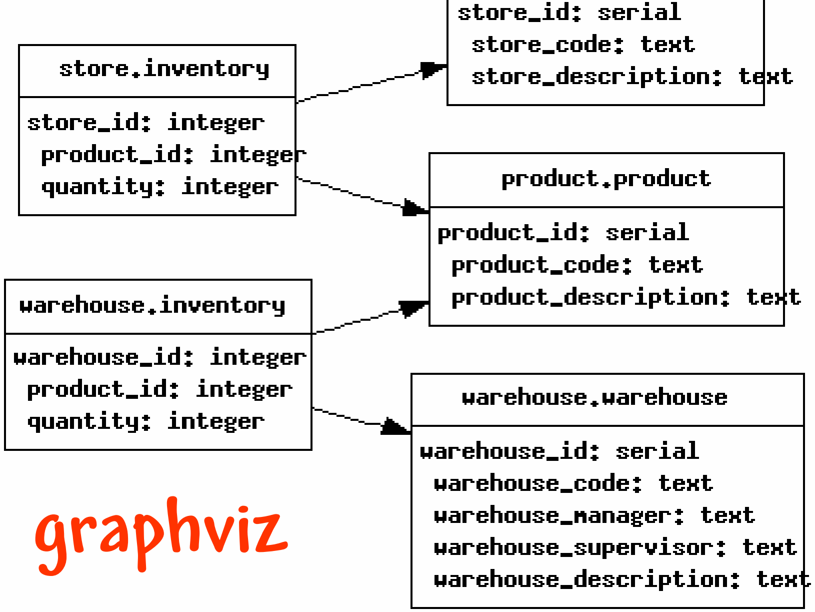
- store.inventory
- warehouse.inventory

Docbook

Autodoc: Examples #2

dia





graphviz

PostGIS

- PostGIS implements support for spatial data, ie. data which describe a location or shape:
    - Points
    - Lines
    - Polygons

- plus functions related to these:
    - Distance
    - Proximity ("touching" and "connectivity")
    - Containing ("inside" and "overlapping")

PostGIS-example: Optimized pub searches

-
  ```
  CREATE TABLE pubs (name VARCHAR, beer_price FLOAT4);
  ```

-
  ```
  ADDGEOMETRYCOLUMN (‘beer_db’,'pubs','location’ ,2167,'POINT',3);
  ```

-
  ```
  INSERT INTO pubs VALUES ('Garricks Head',4.50,GeometryFromText('POINT
  (1196131 383324)’,2167));
  ```

-
  ```
  SELECT name, beer_price,
         DISTANCE(location, GeometryFromText('POINT(1195722 383854)',2167))
         FROM pubs ORDER BY beer_price;
  ```

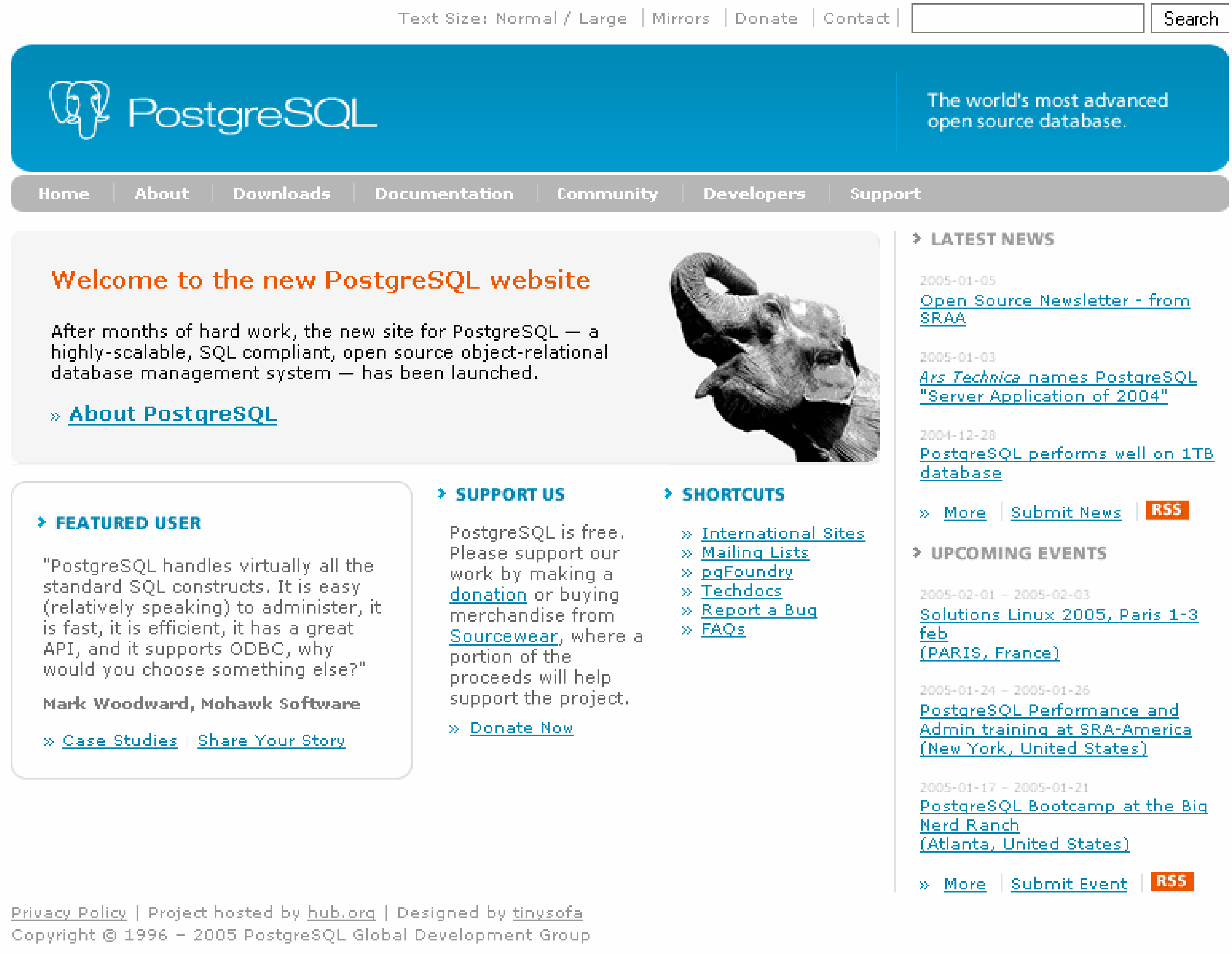
```
     name      | beer_price |     distance
---------------+------------+------------------
 Fireside      |       4.25 | 1484.10275160491
 The Forge     |       4.33 | 1533.06561109862
 Rumours       |       4.46 | 2042.00094093097
 Garricks Head |        4.5 | 669.389105609889
 Slap Happy    |        4.5 | 1882.31910168298
 Old Bailys    |       4.55 | 1147.20900404641
 Black Sheep   |       4.66 | 536.859935972633
 Big Bad Daves |       4.75 | 907.446543878884
```

-
  ```
  SELECT name, beer_price + 0.001 * DISTANCE(location,
         GeometryFromText('POINT(1195722 383854)',2167))
         AS net_price FROM pubs ORDER BY price;
  ```

```
     name      |     net_price
---------------+------------------
 Garricks Head | 5.16938910560989
 Black Sheep   | 5.19685978338474
 Big Bad Daves | 5.65744654387888
 Old Bailys    | 5.69720919478127
 Fireside      | 5.73410275160491
 The Forge     | 5.86306553480468
 Slap Happy    | 6.38231910168298
 Rumours       | 6.50200097907794
```

How to get started?

www.postgresql.org



The world's most advanced open source database.

Welcome to the new PostgreSQL website

After months of hard work, the new site for PostgreSQL — a highly-scalable, SQL compliant, open source object-relational database management system — has been launched.

» About PostgreSQL

FEATURED USER

"PostgreSQL handles virtually all the standard SQL constructs. It is easy (relatively speaking) to administer, it is fast, it is efficient, it has a great API, and it supports ODBC, why would you choose something else?"

Mark Woodward, Mohawk Software

» Case Studies | Share Your Story

SUPPORT US

PostgreSQL is free. Please support our work by making a donation or buying merchandise from Sourcewear, where a portion of the proceeds will help support the project.

» Donate Now

SHORTCUTS

- » International Sites
- » Mailing Lists
- » pgFoundry
- » Techdocs
- » Report a Bug
- » FAQs

LATEST NEWS

2005-01-05
Open Source Newsletter - from SRAA

2005-01-03
Ars Technica names PostgreSQL "Server Application of 2004"

2004-12-28
PostgreSQL performs well on 1TB database

» More | Submit News | RSS

UPCOMING EVENTS

2005-02-01 – 2005-02-03
Solutions Linux 2005, Paris 1-3 feb
(PARIS, France)

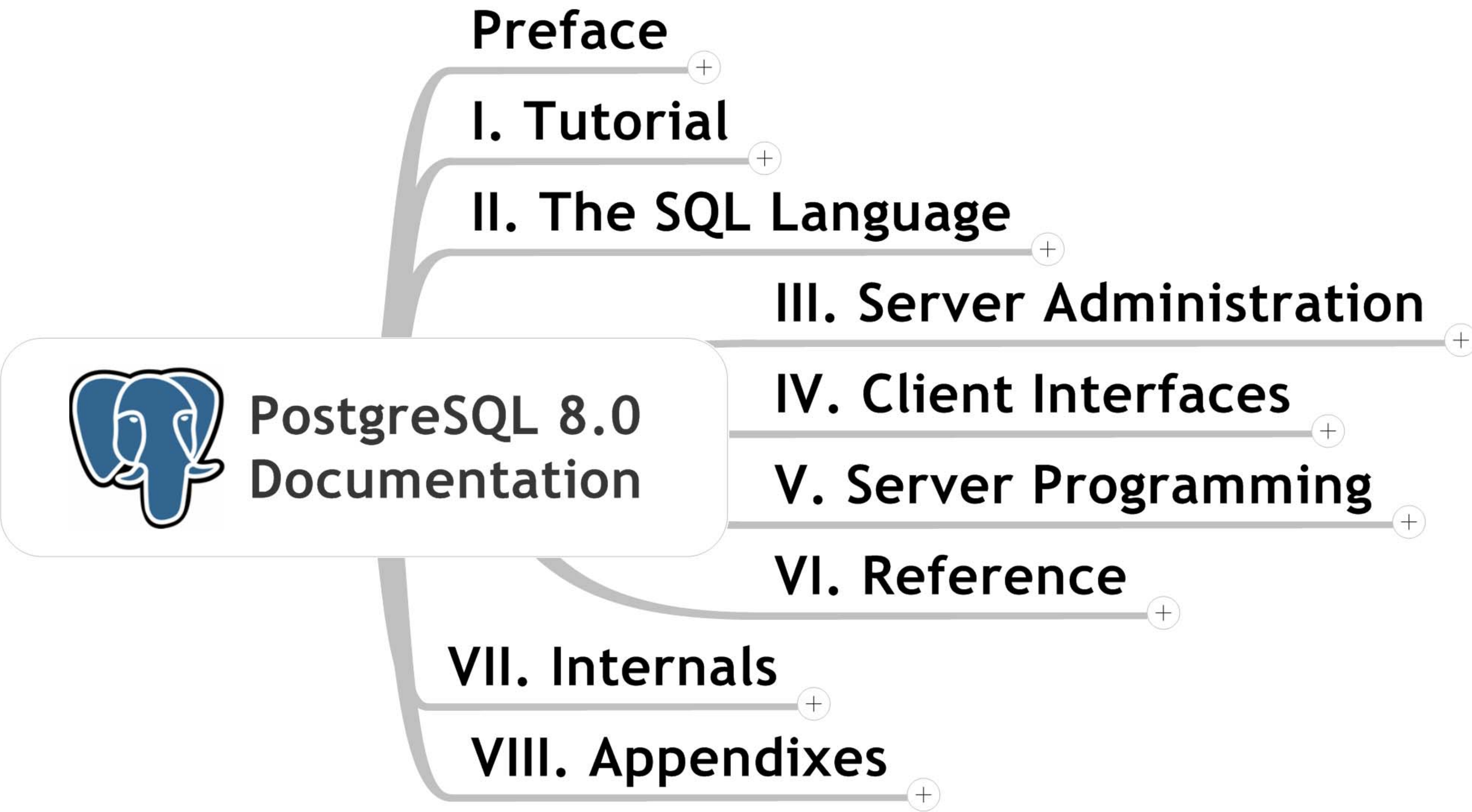
2005-01-24 – 2005-01-26
PostgreSQL Performance and Admin training at SRA-America
(New York, United States)

2005-01-17 – 2005-01-21
PostgreSQL Bootcamp at the Big Nerd Ranch
(Atlanta, United States)

» More | Submit Event | RSS

Privacy Policy | Project hosted by hub.org | Designed by tinysofa
Copyright © 1996 – 2005 PostgreSQL Global Development Group

Documentation #1



PostgreSQL 8.0 Documentation

Preface

I. Tutorial

II. The SQL Language

III. Server Administration

IV. Client Interfaces

V. Server Programming

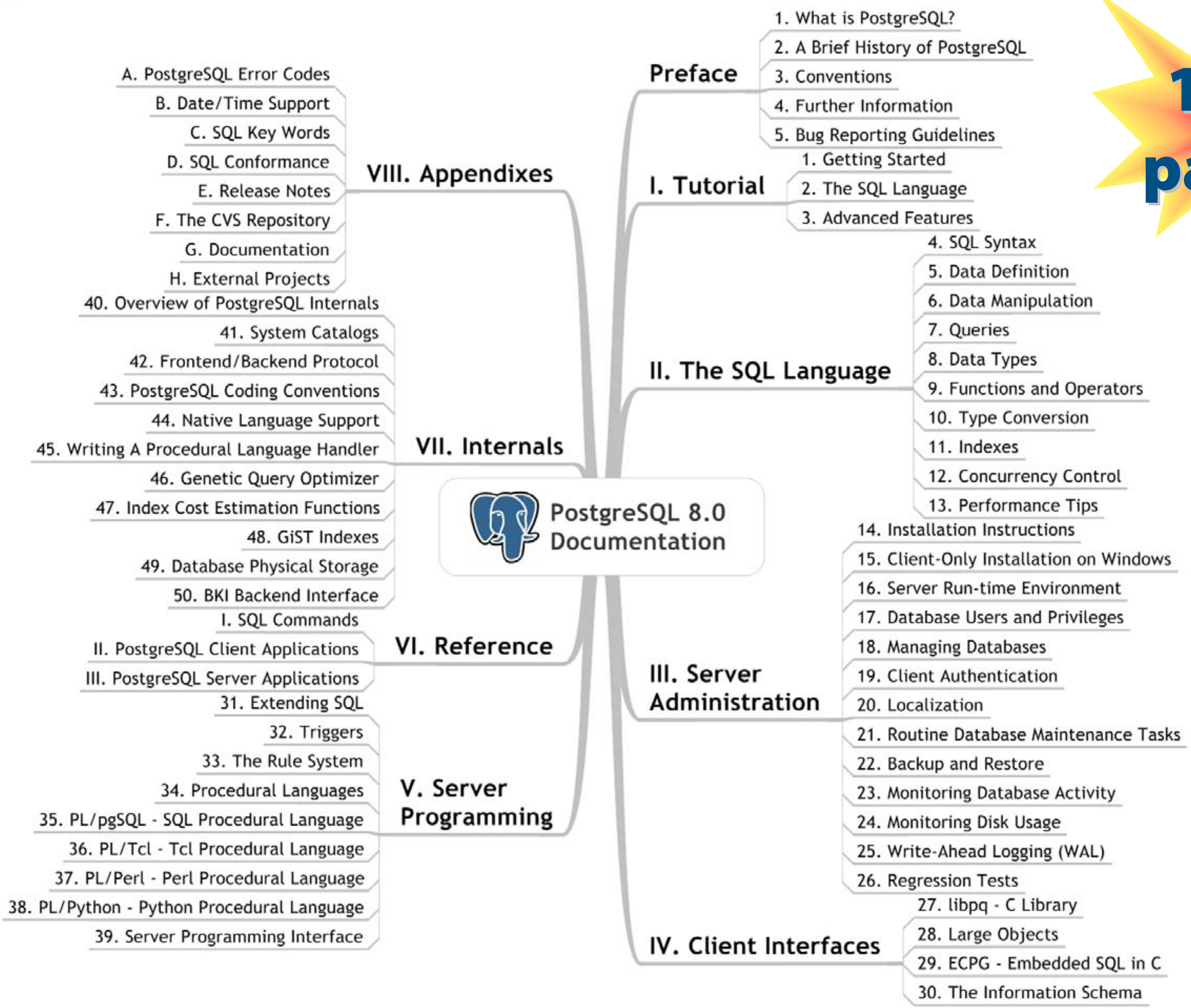
VI. Reference

VII. Internals

VIII. Appendixes

Documentation #2

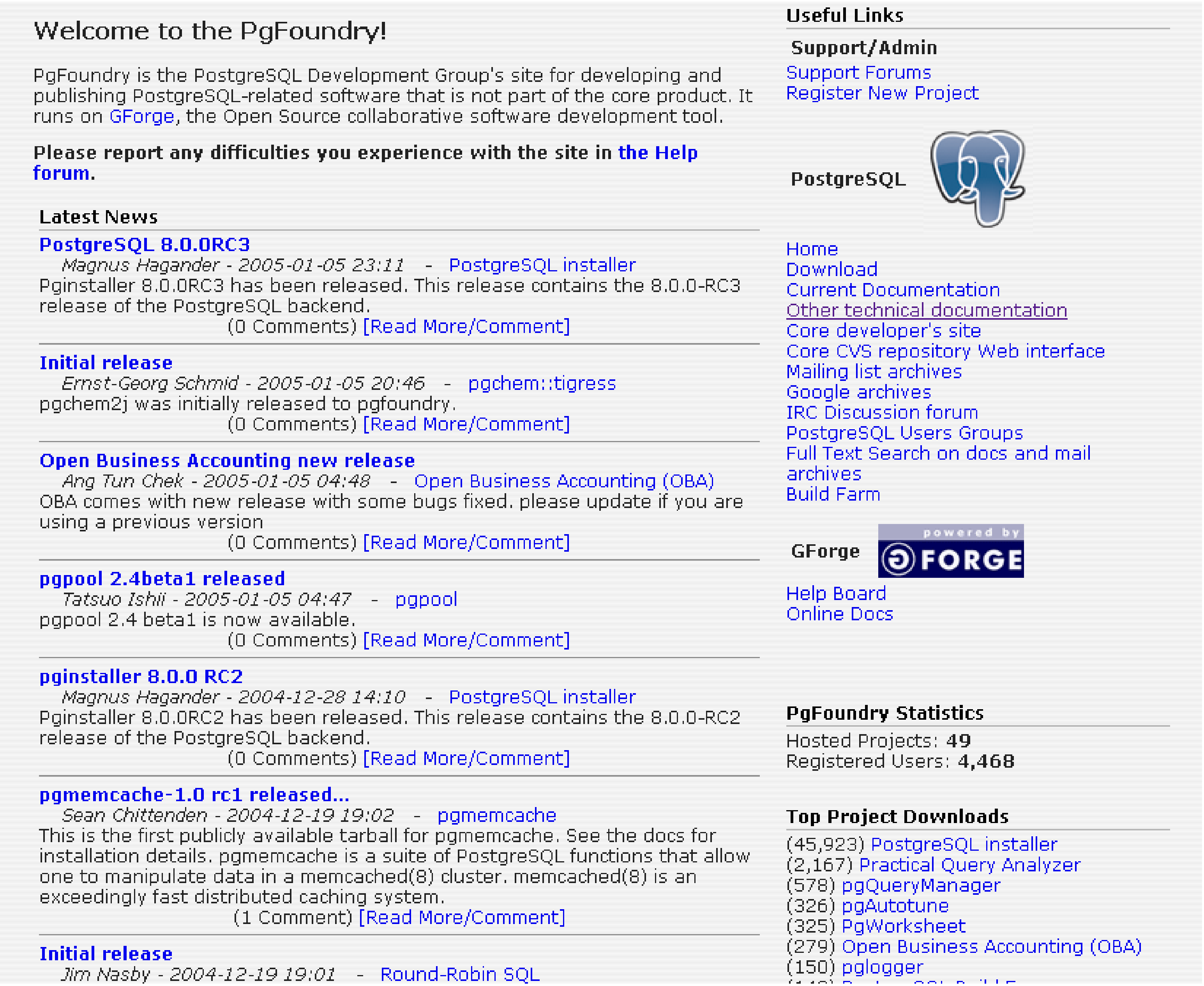
1332 pages!



PostgreSQL 8.0 Documentation

- Preface
  - 1. What is PostgreSQL?
  - 2. A Brief History of PostgreSQL
  - 3. Conventions
  - 4. Further Information
  - 5. Bug Reporting Guidelines
- I. Tutorial
  - 1. Getting Started
  - 2. The SQL Language
  - 3. Advanced Features
- II. The SQL Language
  - 4. SQL Syntax
  - 5. Data Definition
  - 6. Data Manipulation
  - 7. Queries
  - 8. Data Types
  - 9. Functions and Operators
  - 10. Type Conversion
  - 11. Indexes
  - 12. Concurrency Control
  - 13. Performance Tips
- III. Server Administration
  - 14. Installation Instructions
  - 15. Client-Only Installation on Windows
  - 16. Server Run-time Environment
  - 17. Database Users and Privileges
  - 18. Managing Databases
  - 19. Client Authentication
  - 20. Localization
  - 21. Routine Database Maintenance Tasks
  - 22. Backup and Restore
  - 23. Monitoring Database Activity
  - 24. Monitoring Disk Usage
  - 25. Write-Ahead Logging (WAL)
  - 26. Regression Tests
- IV. Client Interfaces
  - 27. libpq - C Library
  - 28. Large Objects
  - 29. ECPG - Embedded SQL in C
  - 30. The Information Schema
- V. Server Programming
  - 31. Extending SQL
  - 32. Triggers
  - 33. The Rule System
  - 34. Procedural Languages
  - 35. PL/pgSQL - SQL Procedural Language
  - 36. PL/Tcl - Tcl Procedural Language
  - 37. PL/Perl - Perl Procedural Language
  - 38. PL/Python - Python Procedural Language
  - 39. Server Programming Interface
- VI. Reference
  - I. SQL Commands
  - II. PostgreSQL Client Applications
  - III. PostgreSQL Server Applications
- VII. Internals
  - 40. Overview of PostgreSQL Internals
  - 41. System Catalogs
  - 42. Frontend/Backend Protocol
  - 43. PostgreSQL Coding Conventions
  - 44. Native Language Support
  - 45. Writing A Procedural Language Handler
  - 46. Genetic Query Optimizer
  - 47. Index Cost Estimation Functions
  - 48. GiST Indexes
  - 49. Database Physical Storage
  - 50. BKI Backend Interface
- VIII. Appendixes
  - A. PostgreSQL Error Codes
  - B. Date/Time Support
  - C. SQL Key Words
  - D. SQL Conformance
  - E. Release Notes
  - F. The CVS Repository
  - G. Documentation
  - H. External Projects

http://pgfoundry.org



Welcome to the PgFoundry!

PgFoundry is the PostgreSQL Development Group's site for developing and publishing PostgreSQL-related software that is not part of the core product. It runs on GForge, the Open Source collaborative software development tool.

Please report any difficulties you experience with the site in the Help forum.

Latest News

PostgreSQL 8.0.0RC3
Magnus Hagander - 2005-01-05 23:11 - PostgreSQL installer
Pginstaller 8.0.0RC3 has been released. This release contains the 8.0.0-RC3 release of the PostgreSQL backend.
(0 Comments) [Read More/Comment]

Initial release
Ernst-Georg Schmid - 2005-01-05 20:46 - pgchem::tigress
pgchem2j was initially released to pgfoundry.
(0 Comments) [Read More/Comment]

Open Business Accounting new release
Ang Tun Chek - 2005-01-05 04:48 - Open Business Accounting (OBA)
OBA comes with new release with some bugs fixed. please update if you are using a previous version
(0 Comments) [Read More/Comment]

pgpool 2.4beta1 released
Tatsuo Ishii - 2005-01-05 04:47 - pgpool
pgpool 2.4 beta1 is now available.
(0 Comments) [Read More/Comment]

pginstaller 8.0.0 RC2
Magnus Hagander - 2004-12-28 14:10 - PostgreSQL installer
Pginstaller 8.0.0RC2 has been released. This release contains the 8.0.0-RC2 release of the PostgreSQL backend.
(0 Comments) [Read More/Comment]

pgmemcache-1.0 rc1 released...
Sean Chittenden - 2004-12-19 19:02 - pgmemcache
This is the first publicly available tarball for pgmemcache. See the docs for installation details. pgmemcache is a suite of PostgreSQL functions that allow one to manipulate data in a memcached(8) cluster. memcached(8) is an exceedingly fast distributed caching system.
(1 Comment) [Read More/Comment]

Initial release
Jim Nasby - 2004-12-19 19:01 - Round-Robin SQL

Useful Links

Support/Admin
Support Forums
Register New Project

PostgreSQL
Home
Download
Current Documentation
Other technical documentation
Core developer's site
Core CVS repository Web interface
Mailing list archives
Google archives
IRC Discussion forum
PostgreSQL Users Groups
Full Text Search on docs and mail archives
Build Farm

GForge
powered by
FORGE
Help Board
Online Docs

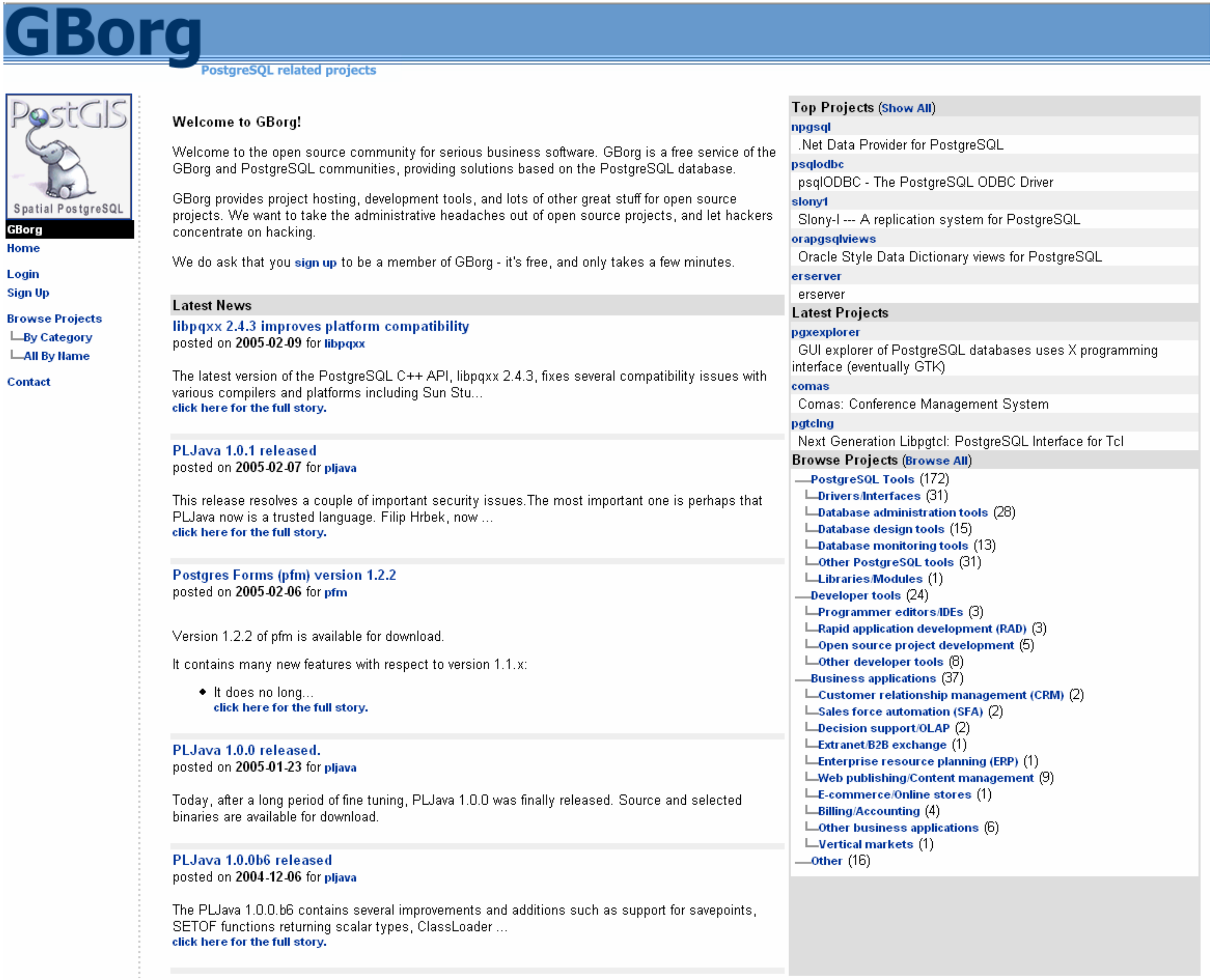
PgFoundry Statistics

Hosted Projects: **49**
Registered Users: **4,468**

Top Project Downloads

(45,923) PostgreSQL installer
(2,167) Practical Query Analyzer
(578) pgQueryManager
(326) pgAutotune
(325) PgWorksheet
(279) Open Business Accounting (OBA)
(150) pglogger

http://gborg.postgresql.org/



GBorg

PostgreSQL related projects

GBorg

- Home
- Login
- Sign Up
- Browse Projects
  - By Category
  - All By Name
- Contact

Welcome to GBorg!

Welcome to the open source community for serious business software. GBorg is a free service of the GBorg and PostgreSQL communities, providing solutions based on the PostgreSQL database.

GBorg provides project hosting, development tools, and lots of other great stuff for open source projects. We want to take the administrative headaches out of open source projects, and let hackers concentrate on hacking.

We do ask that you sign up to be a member of GBorg - it's free, and only takes a few minutes.

Latest News

libpqxx 2.4.3 improves platform compatibility
posted on 2005-02-09 for libpqxx

The latest version of the PostgreSQL C++ API, libpqxx 2.4.3, fixes several compatibility issues with various compilers and platforms including Sun Stu...
click here for the full story.

PLJava 1.0.1 released
posted on 2005-02-07 for pljava

This release resolves a couple of important security issues.The most important one is perhaps that PLJava now is a trusted language. Filip Hrbek, now ...
click here for the full story.

Postgres Forms (pfm) version 1.2.2
posted on 2005-02-06 for pfm

Version 1.2.2 of pfm is available for download.

It contains many new features with respect to version 1.1.x:

- It does no long...
  click here for the full story.

PLJava 1.0.0 released.
posted on 2005-01-23 for pljava

Today, after a long period of fine tuning, PLJava 1.0.0 was finally released. Source and selected binaries are available for download.

PLJava 1.0.0b6 released
posted on 2004-12-06 for pljava

The PLJava 1.0.0.b6 contains several improvements and additions such as support for savepoints, SETOF functions returning scalar types, ClassLoader ...
click here for the full story.

Top Projects (Show All)

npgsql
.Net Data Provider for PostgreSQL

psqlodbc
psqlODBC - The PostgreSQL ODBC Driver

slony1
Slony-I --- A replication system for PostgreSQL

orapgsqlviews
Oracle Style Data Dictionary views for PostgreSQL

erserver
erserver

Latest Projects

pgxexplorer
GUI explorer of PostgreSQL databases uses X programming interface (eventually GTK)

comas
Comas: Conference Management System

pgtclng
Next Generation Libpgtcl: PostgreSQL Interface for Tcl

Browse Projects (Browse All)

- PostgreSQL Tools (172)
  - Drivers/Interfaces (31)
  - Database administration tools (28)
  - Database design tools (15)
  - Database monitoring tools (13)
  - Other PostgreSQL tools (31)
  - Libraries/Modules (1)
- Developer tools (24)
  - Programmer editors/IDEs (3)
  - Rapid application development (RAD) (3)
  - Open source project development (5)
  - Other developer tools (8)
- Business applications (37)
  - Customer relationship management (CRM) (2)
  - Sales force automation (SFA) (2)
  - Decision support/OLAP (2)
  - Extranet/B2B exchange (1)
  - Enterprise resource planning (ERP) (1)
  - Web publishing/Content management (9)
  - E-commerce/Online stores (1)
  - Billing/Accounting (4)
  - Other business applications (6)
  - Vertical markets (1)
- Other (16)

Mailing lists & IRC

- An assortment of mailing lists are available:
  - http://www.postgresql.org/community/lists/subscribe
  - High volume
  - High level of competence
  - User-friendly
- Archives available from:
  - http://archives.postgresql.org/
- IRC: irc.freenode.net/#postgresql
  - An unique mix of competence and friendliness

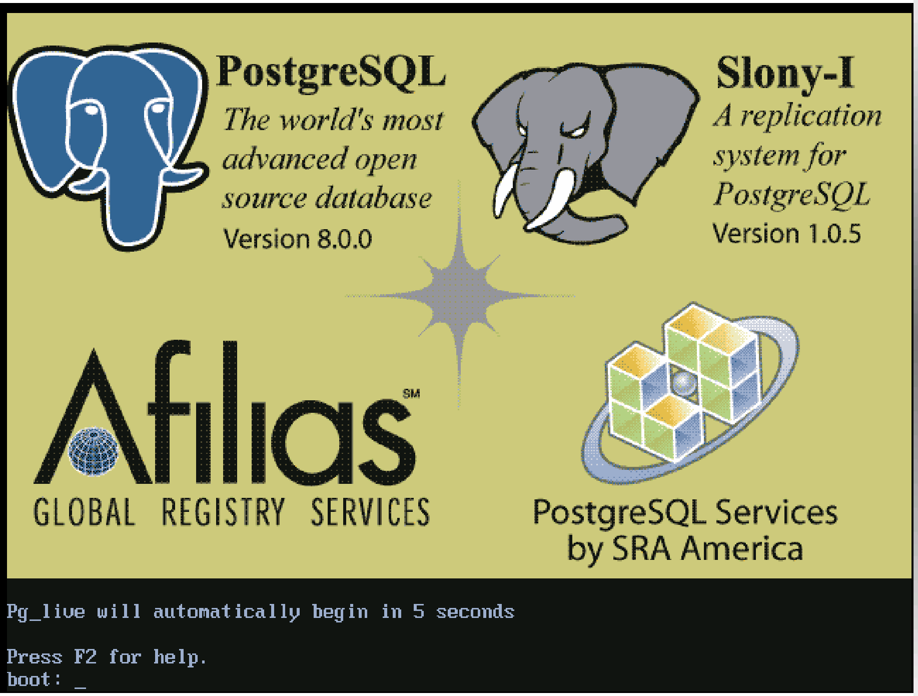
- pgsql-admin
- pgsql-advocacy
- pgsql-announce
- pgsql-bugs
- pgsql-docs
- pgsql-general
- pgsql-hackers
- pgsql-interfaces
- pgsql-jdbc
- pgsql-novice
- pgsql-odbc
- pgsql-performance
- pgsql-php
- pgsql-sql

Web resources

- http://techdocs.postgresql.org/
  - Technical articles and miscellaneous information
- General Bits by A. Elein Mustain
  - http://www.varlena.com/GeneralBits
  - Weekly summary of the pgsql-general mailing list
- PGSearch:
  - http://www.pgsql.ru/db/pgsearch
  - Search engine based on PostgreSQL and TSearch2

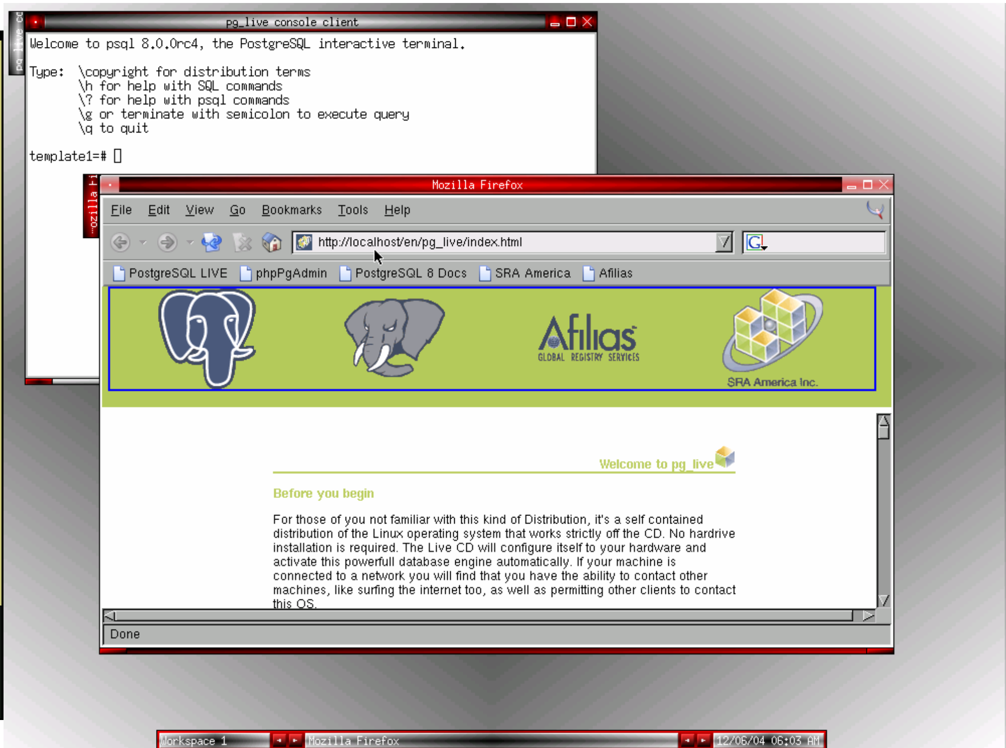
pg_live

- Knoppix-based live-CD with PostgreSQL
  - Compiled by Robert Bernier
  - Newest version is 1.3.3, released 8. februar 2005
  - http://www.sraapowergres.com + /en/newsletter/issue_02/pg_live/pg_live.1.3.3.iso


PostgreSQL
The world's most advanced open source database
Version 8.0.0
Slony-I
A replication system for PostgreSQL
Version 1.0.5
Afilias SM
GLOBAL REGISTRY SERVICES
PostgreSQL Services by SRA America

```
Pg_live will automatically begin in 5 seconds

Press F2 for help.
boot: _
```



```
Welcome to psql 8.0.0rc4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

template1=# 
```

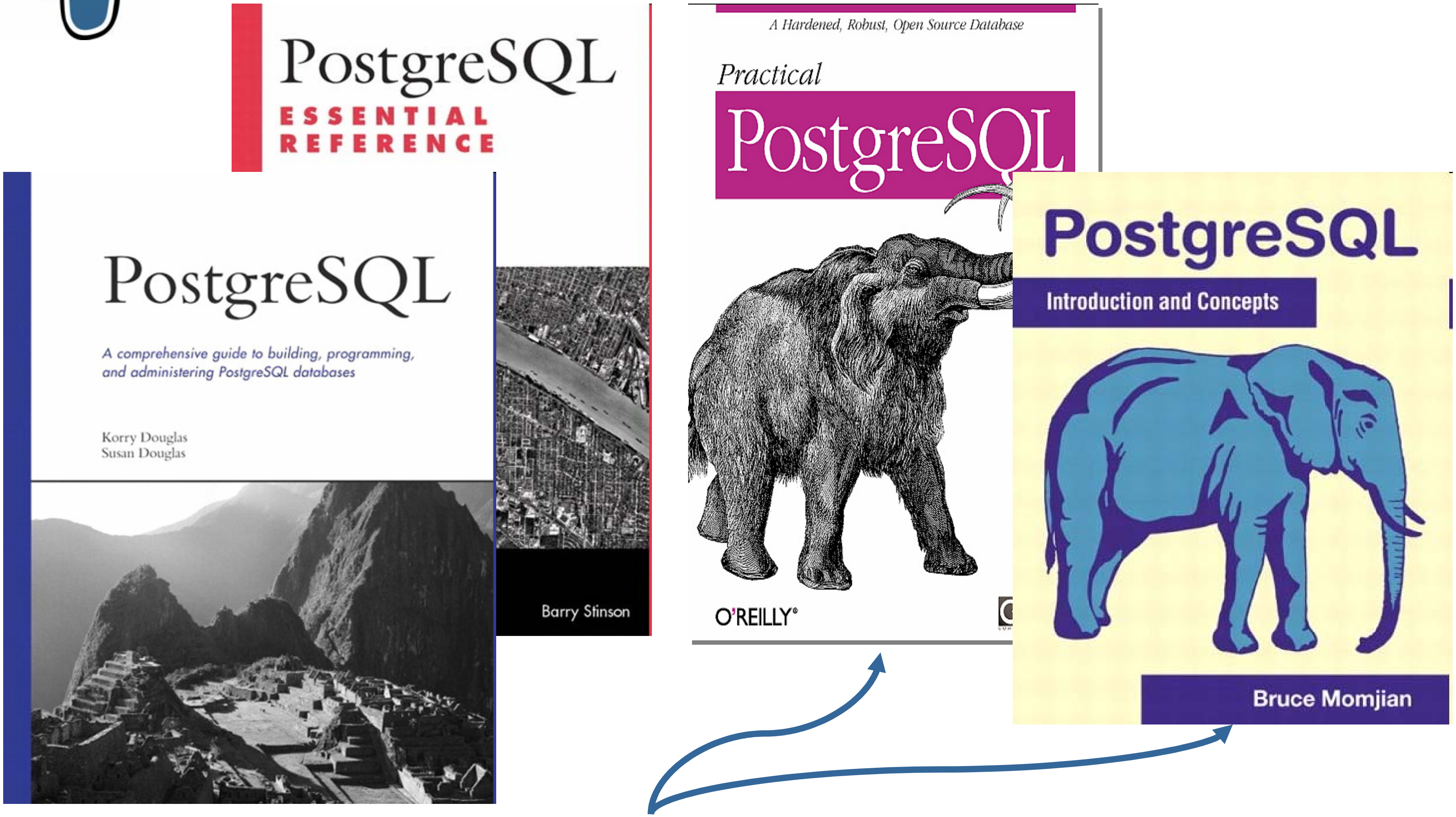
http://localhost/en/pg_live/index.html

Welcome to pg_live

Before you begin

For those of you not familiar with this kind of Distribution, it's a self contained distribution of the Linux operating system that works strictly off the CD. No hardrive installation is required. The Live CD will configure itself to your hardware and activate this powerfull database engine automatically. If your machine is connected to a network you will find that you have the ability to contact other machines, like surfing the internet too, as well as permitting other clients to contact this OS.

Books about PostgreSQL


PostgreSQL
ESSENTIAL REFERENCE
Barry Stinson
A Hardened, Robust, Open Source Database
Practical PostgreSQL
O'REILLY
PostgreSQL
A comprehensive guide to building, programming, and administering PostgreSQL databases
Korry Douglas
Susan Douglas
PostgreSQL
Introduction and Concepts
Bruce Momjian

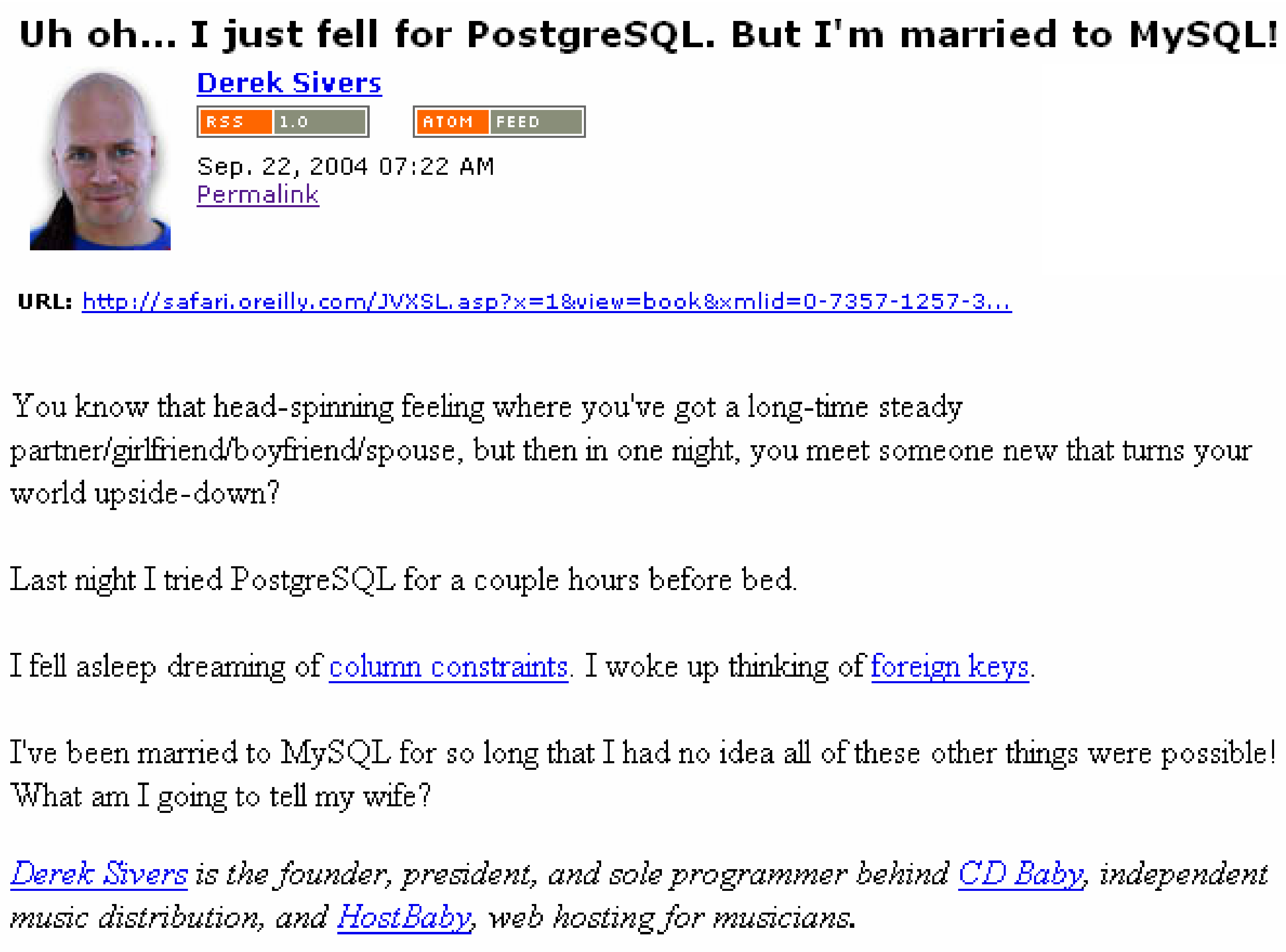
- **http://www.postgresql.org/docs/books/awbook.html**
- **http://www.commandprompt.com/ppbook/**

Questions?

- The presentation is available from:
  - http://www.tricknology.org/foilware/

oddbjorn@tricknology.org

Uh oh...



Uh oh... I just fell for PostgreSQL. But I'm married to MySQL!

Derek Sivers

RSS 1.0 ATOM FEED

Sep. 22, 2004 07:22 AM
Permalink

URL: http://safari.oreilly.com/JVXSL.asp?x=1&view=book&xmlid=0-7357-1257-3...

You know that head-spinning feeling where you've got a long-time steady partner/girlfriend/boyfriend/spouse, but then in one night, you meet someone new that turns your world upside-down?

Last night I tried PostgreSQL for a couple hours before bed.

I fell asleep dreaming of column constraints. I woke up thinking of foreign keys.

I've been married to MySQL for so long that I had no idea all of these other things were possible! What am I going to tell my wife?

Derek Sivers is the founder, president, and sole programmer behind CD Baby, independent music distribution, and HostBaby, web hosting for musicians.