A. PENDAHULUAN

PostgreSQL merupakan sebuah *Object-Relational Database Management System* (ORDBMS) berdasarkan pada PostgreSQL Versi 4.2 yang dikembangkan di Universitas California pada Berkeley Computer Science Department. PostgreSQL sebagi pelopor bagi banyak software DBMS lain yang kemudian menjadi komersial.

PostgreSQL memiliki lisensi GPL (*General Public License*) dan oleh karena itu PostgreSQL dapat digunakan, dimodifikasi dan didistribusikan oleh setiap orang tanpa perlu membayar lisensi (*free of charge*) baik untuk keperluan pribadi, pendidikan maupun komersil. PostgreSQL merupakan DBMS yang open-source yang mendukung bahasa SQL secara luas dan menawarkan beberapa fitur-fitur modern seperti :

- Complex Queries
- Foreign Keys
- Triggers
- Views
- Transactional Integrity
- Multiversion Concurrency Control

Selain itu, PostgreSQL telah mendukung teknologi lama dengan menambahkan fitur-fitur baru pada :

- Data Types
- Functions
- Operators
- Aggregate Functions
- Index Methods
- Procedural Languages

NO	ITEM	DESKRIPSI
1	Pengembang	Professor Michael Stonebraker
2	Versi Terakhir	8.2.0
3	Tanggal Direlease	05-Desember-2006
4	Sistem Operasi	Kompatibel Antar Platform
5	Kelompok	Object Relational-DBMS
6	Lisensi	GPL (General Public License)
7	Logo	Elephan
8	Situs	http://www.postgresql.org

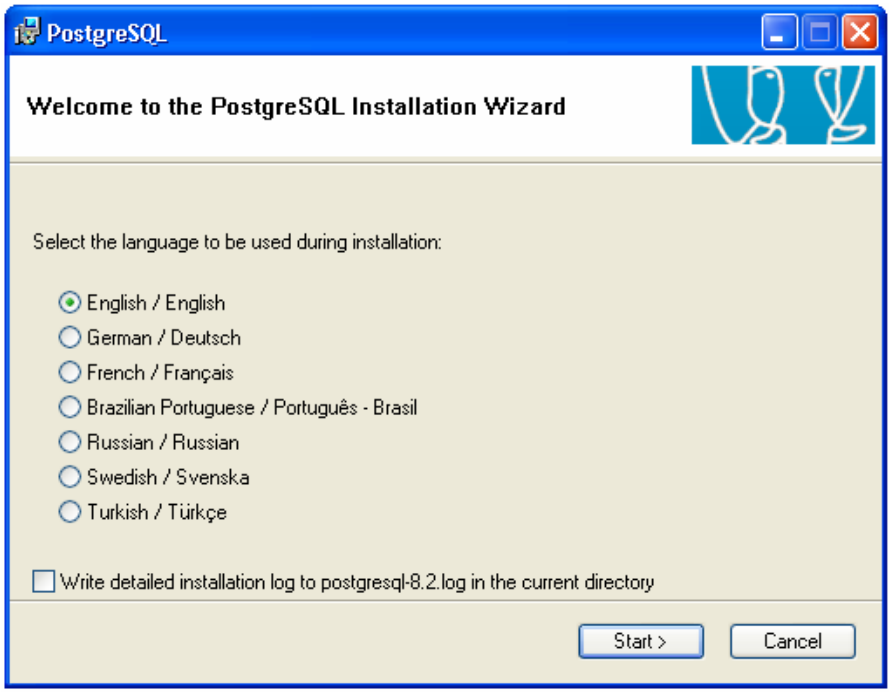
Gambar Logo PostgreSQL

B. SEJARAH SINGKAT POSTGRESQL

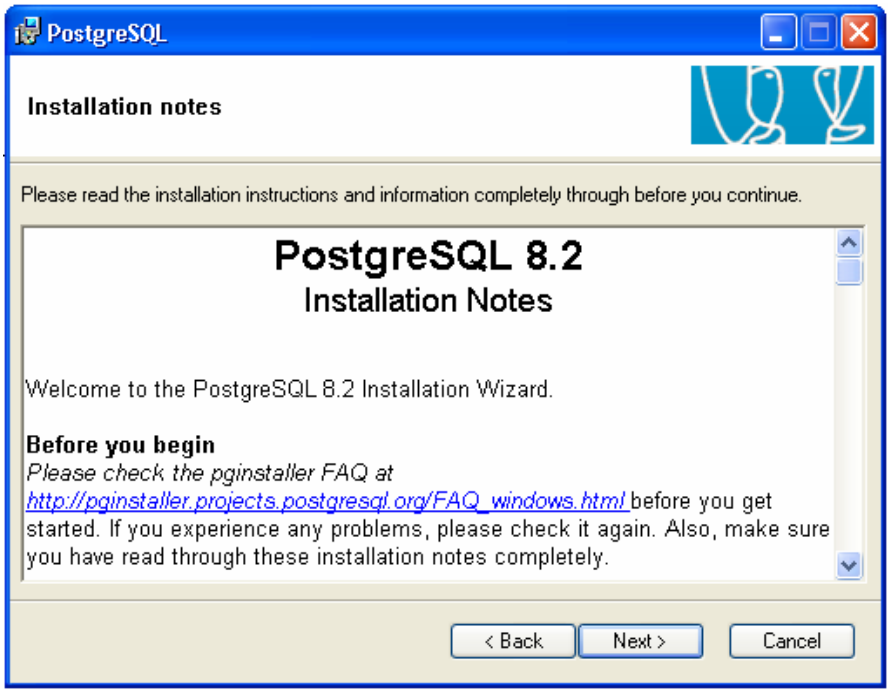
NO	WAKTU	DESKRIPSI
1	1986	Dikembangkan pertama kali oleh Professor Michael Stonebraker, yang disponsori oleh the Defense Advanced Research Projects Agency (DARPA). Tahun ini merupakan inisialisasi konsep untuk pengembangan sistem.
2	1987	Dikembangkan definisi dari model data, pembuatan aturan main, konvensi rasional dan arsitektur dari media penyimpanan.
3	1989	Postgre versi 1 dilaunching dengan banyak kelemahan pada konsep aturan main.
4	1990	Postgre versi 2 dilaunching dengan perbaikan pada aturan main.
5	1991	Postgre versi 3 dilaunching dengan dukungan multiple storage managers, peningkatan pada pengeksekusi query , dan ditulis ulangnya aturan main sistem.
6	1993	Postgre versi 4.2 dilaunching yang merupakan cikal bakal DBMS masa depan dengan fitur yang lengkap.
7	1994	Postgre berubah nama menjadi Postgres95. Andrew Yu dan Jolly Chen, menambahkan sebuah interpreter untuk bahasa SQL.
8	1996	Postgres95 berubah nama menjadi PostgreSQL. Dan versi ini telah mencapai PostgreSQL versi 6.0 dengan kemampuan yang lebih baik dan ditandai dengan dimulainya proyek Postgres pada Berkeley Research.
9	1997-Sekarang	PostgresSQL telah berkembang dan terus dikembangkan sebagai database relasional dengan lisensi GPL. Hingga saat Desember 2006 versi terakhir dari PostgreSQL adalah 8.2.0.

C. INSTALASI POSTGRESQL UNDER WINDOWS

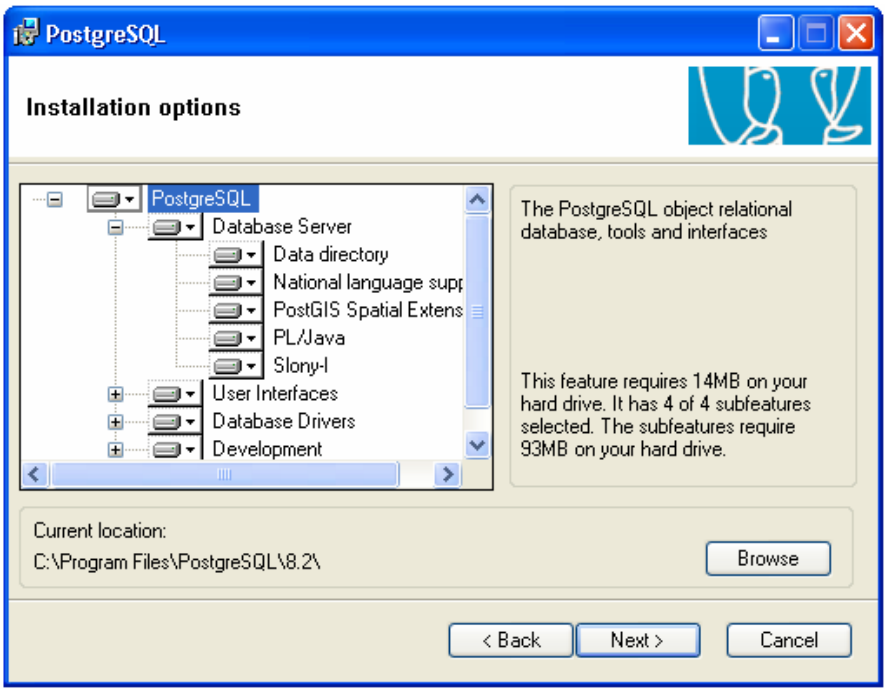
Untuk memulai instalasi silahkan mendownload software PostgreSQL 8.2.0 pada situs http://www.postgresql.org dan pilihlah yang versi Windows.



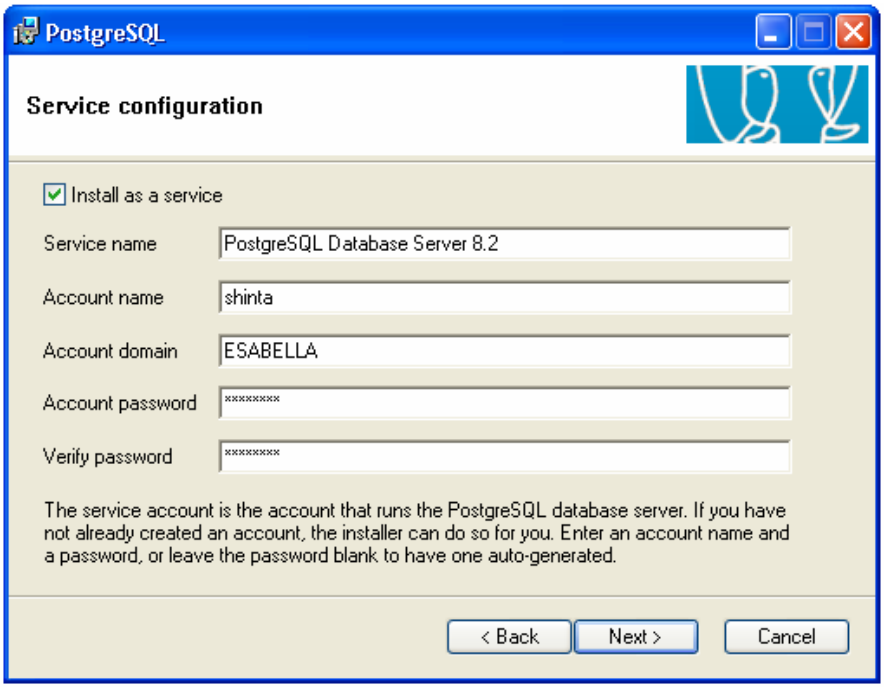
Pilihlah bahasa yang digunakan selama proses installasi. Default adalah English.



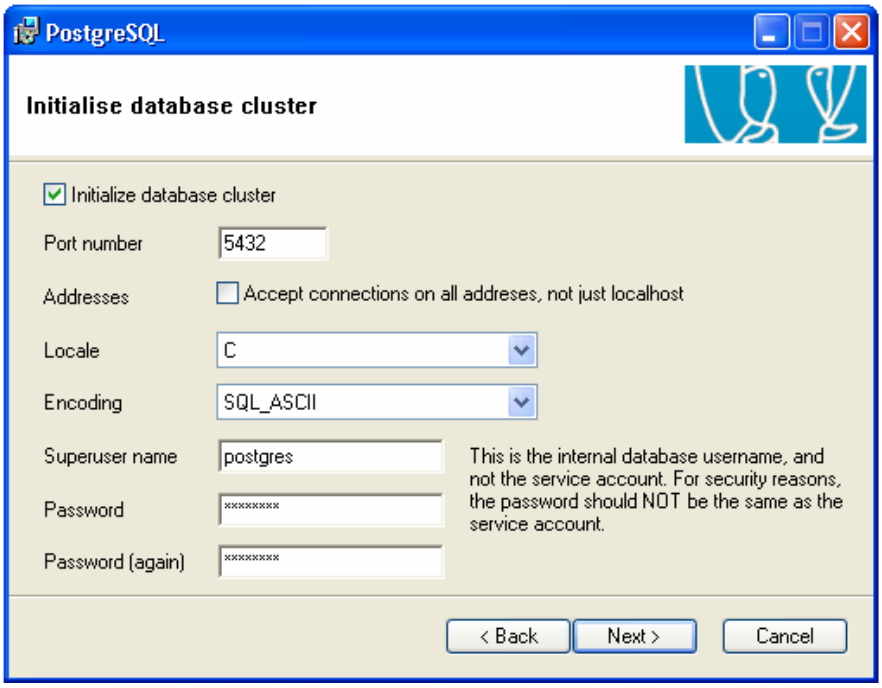
Bacalah catatan installasi dengan seksama demi kemudahan dan keamanan.



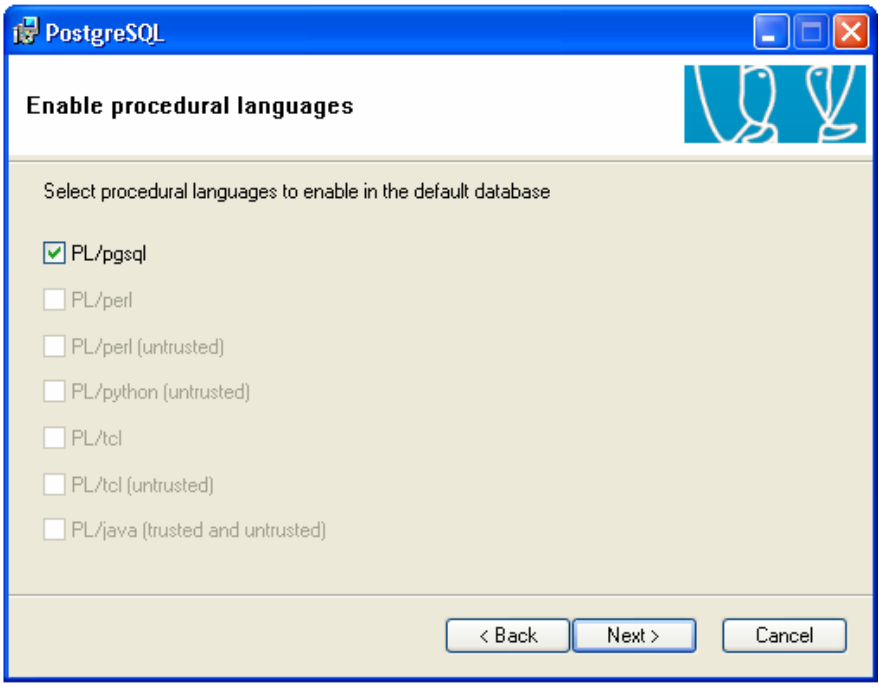
Pada Installation Option, pastikan semua opsi / pilihan terinstall demi kelengkapan dan performa DBMS yang maksimal. Tentukan lokasi untuk melakukan penginstallan. PostgreSQL memungkinkan kita untuk menginstall lebih dari satu versi.



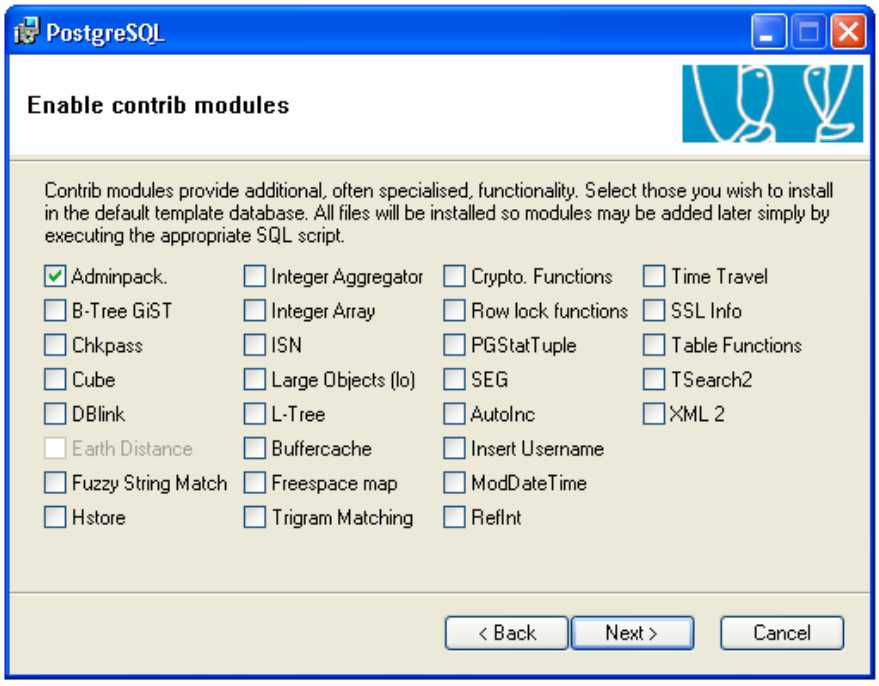
Pada Service Configuration, isikan account name dan password untuk service PostgreSQL. Install as a service maksudnya adalah PostgreSQL akan dijalankan secara otomatis saat Windows startup tanpa perlu melakukan konfigurasi kembali.



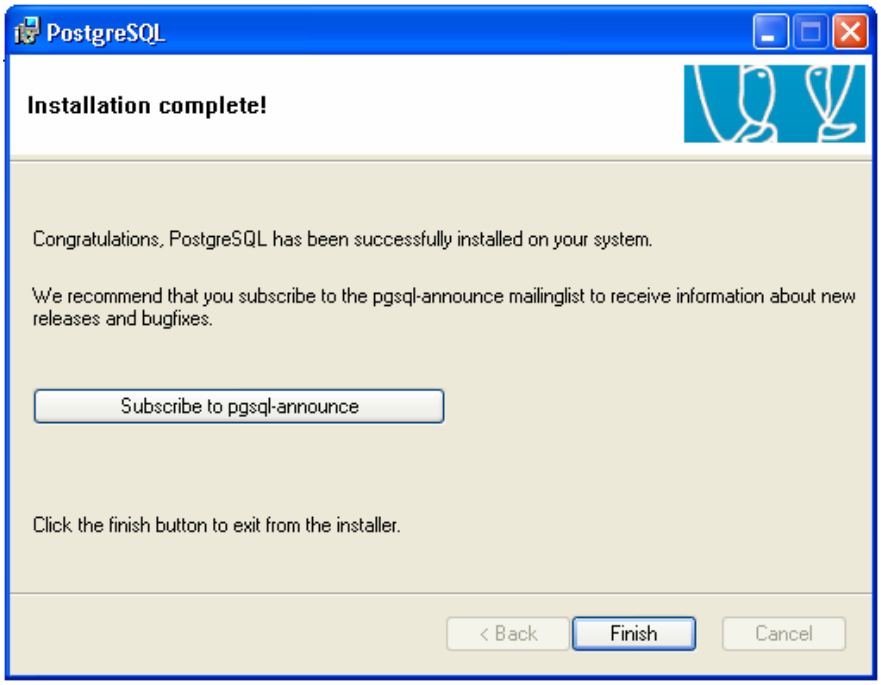
Pada Initialise Database Cluster, maksudnya untuk menginisialisasi datatabase tahap awal. Port default yang digunakan adalah 5432, dan superuser namenya adalah postgres, dan kita wajib mengisi password untuk superuser ini.



Pada Enable Procedural Language, maksudnya mengizinkan PosgtreSQL untuk mengenali Prosedural Language / Structured Query Language (PL/SQL). Default yang terinstall adalah PL/PGSQL. Untuk menginstall Prosedulal Language yang lain, silahkan mendownload dahulu dari situsnya masing-masing.



Pada Enable Contrib Modules, kita dapat memilih modul add-in yang menyediakan fungsi secara lebih spesifik. Semua modul ini dapat ditambahkan nantinya dengan mengeksekusi SQL script yang sesuai dengan fungsinya.

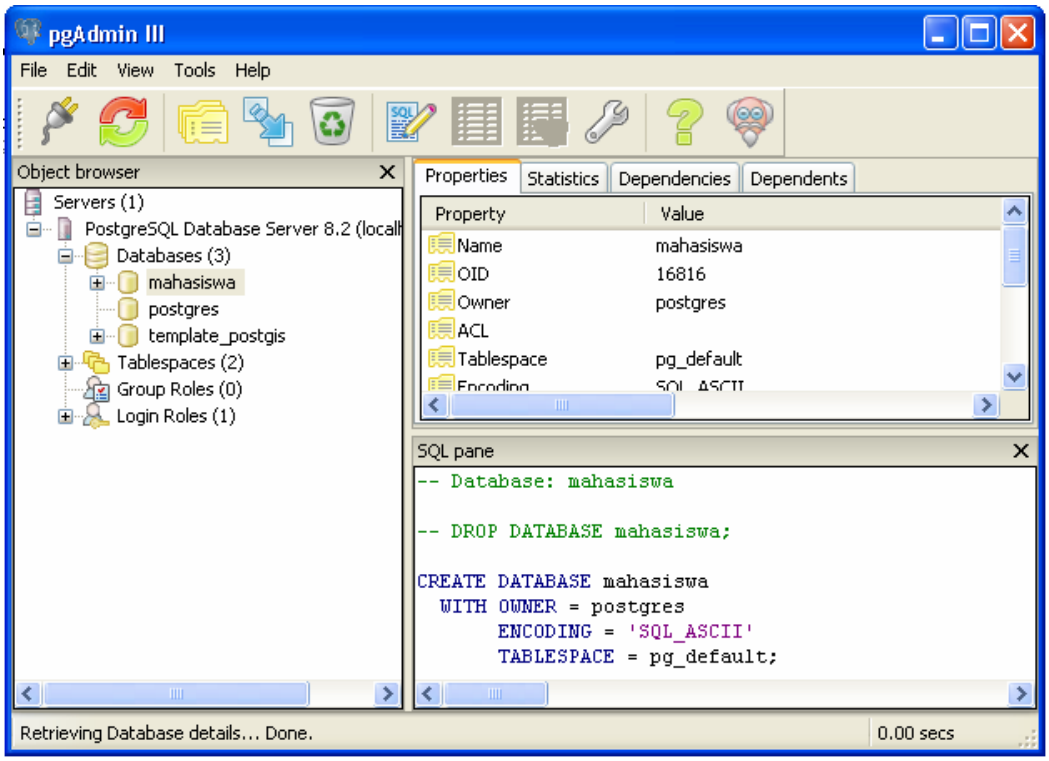


Pada Installation Complete, maksudnya proses installasi telah berjalan dengan baik, dan PostgreSQL siap digunakan. Dan kita dapat pula mendaftakna diri untuk mendapatkan informasi mengenai produk PostgreSQL yang terbaru dengan menekan Subscribe To PgSQL-Announce.

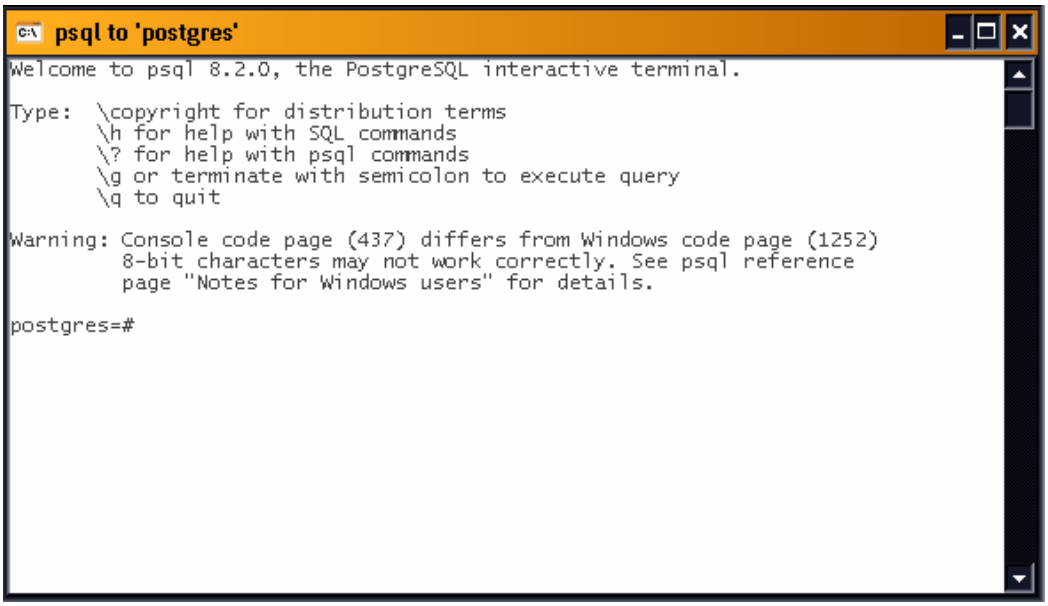
D. TOOLS BUILT-IN POSTGRESQL

Untuk memudahkan administrasi basis data pada PostgeSQL, maka PostgeSQL 8.2 menyediakan alat bantu bagi para pengguna yakni PGAdmin III yang merupakan alat bantu (*tools*) berbasis Graphical User Interface (GUI) dan PSql To Postgres yang merupakan alat bantu (*tools*) bebasis command prompt.

GUI-BASED > PGADMIN III



COMMAND BASED > PSQL to POSTGRES



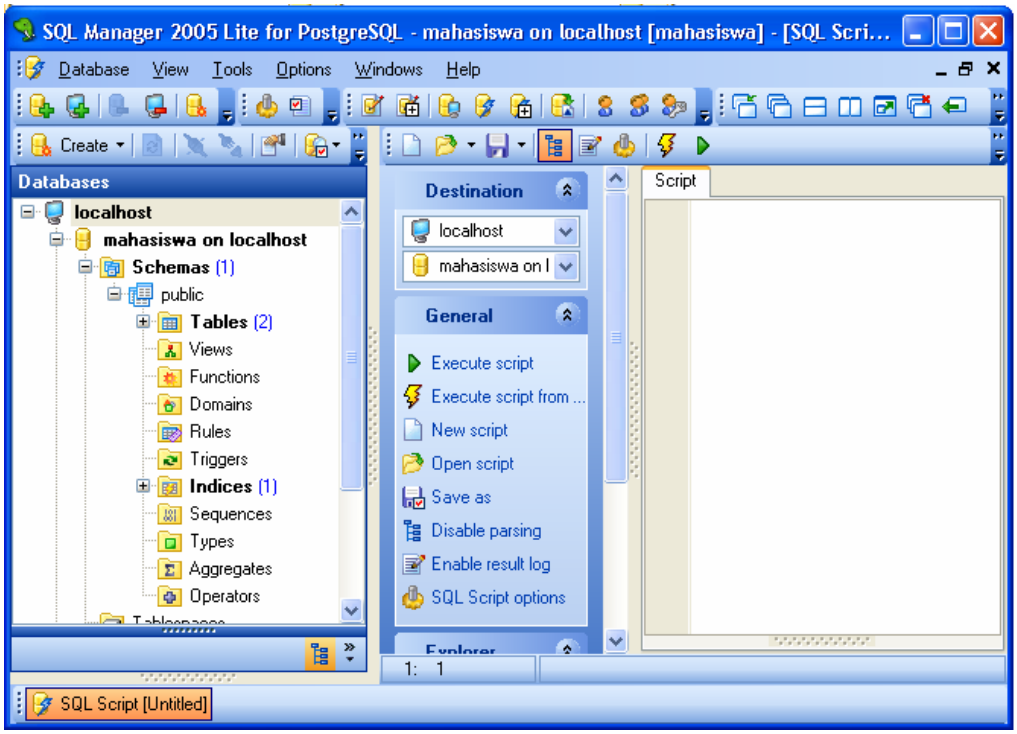
E. THIRD PARTY TOOLS

Bagi pada database administrator yang tidak puas akan alat bantu (*tools*) yang dibawa PostgreSQL 8.2, maka dapat mendownload alat bantu pihak ketiga (*third party tools*) dari Internet. Ada banyak tools yang dapat didownload, seperti :

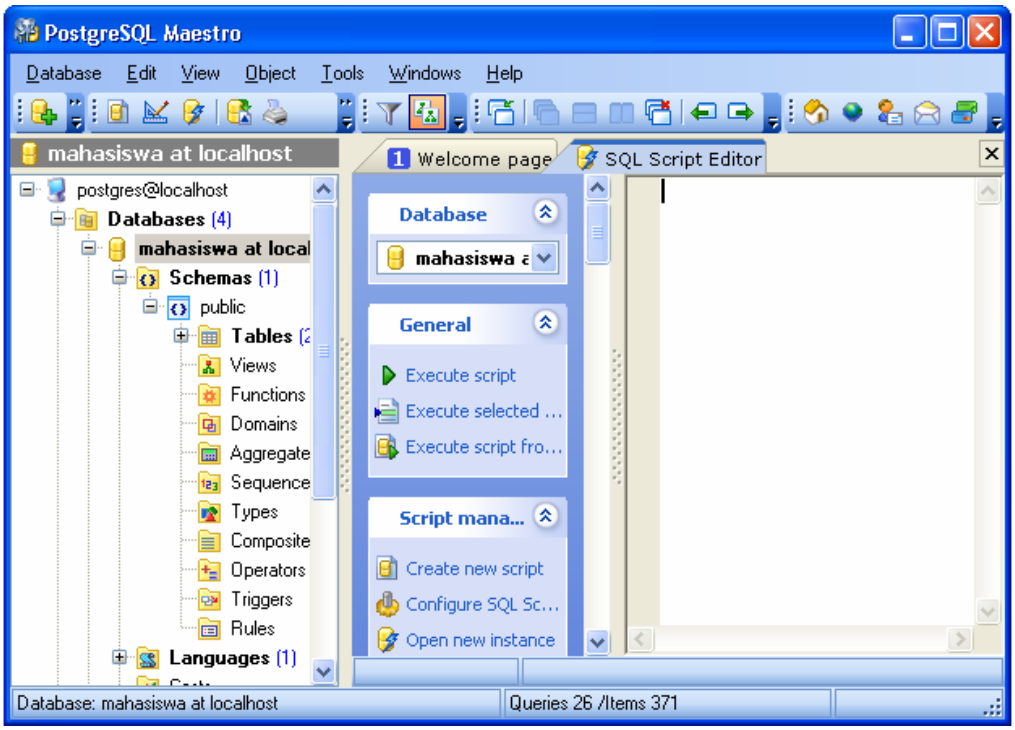
- EMS SQL Manager 2005 For PostgreSQL (http://www.pgsqlmanager.com/), Software ini bersifat Free, kinerja yang ditawarkan juga terbilang baik dan tampilannya yang sangat *user friendly*.
- PostgreSQL Maestro (http://www.sqlmaestro.com/), Software ini bersifat Trial 30 hari, jika ingin menggunakannya lebih lanjut, maka kita diharuskan membeli software ini. Tampilannya sangat mirip dengan EMS SQL Manager.
- Navicat For PostgreSQL (http://pgsqlsupport.navicat.com/), Software ini bersifat Trial 30 hari, jika ingin menggunakannya lebih lanjut, maka kita diharuskan membeli software ini. Tampilannya lebih sederhana untuk pengguna awam. Namun kemampuannya juga tidak kalah dengan software sejenis seperti EMS Manager atau PostgreSQL Maestro.

Tools hanya merupakan alat bantu, yang lebih penting adalah pengetehuan mengenai bagaimana melalukan administrasi pada sebuah DBMS seperti PostgreSQL.

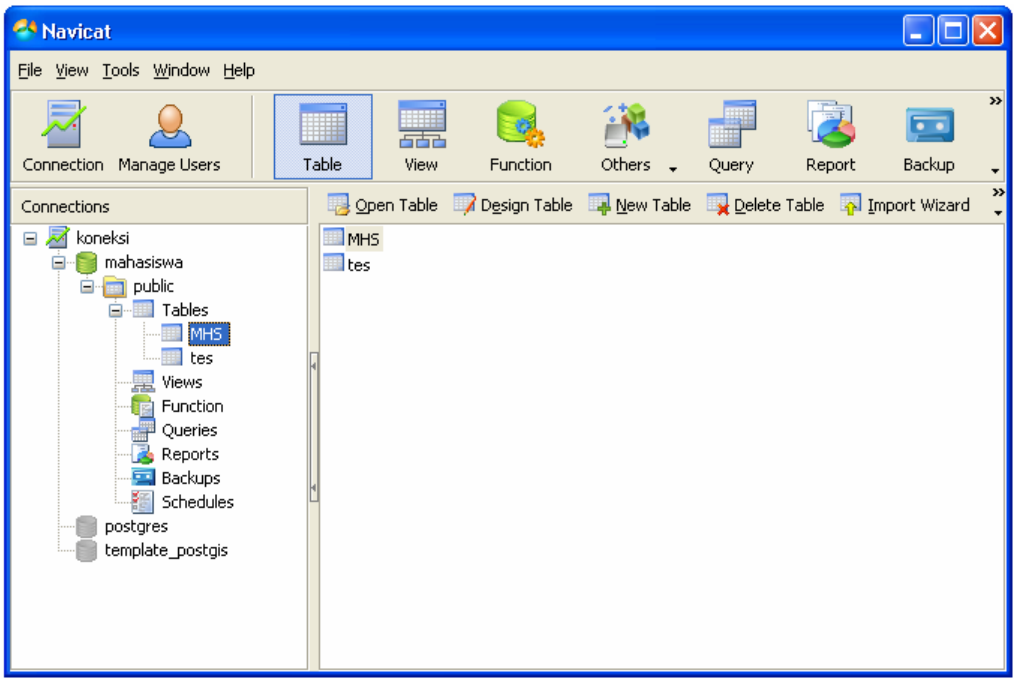
EMS SQL Manager 2005 For PostgreSQL



PostgreSQL Maestro



Navicat For PostgreSQL



F. PERINTAH YANG BERHUBUNGAN DENGAN SQL COMMAND

Berikut ini daftar perintah bantuan yang tersedia di PostgeSQL 8.2 jika kita menggunakan Command Prompt. Jika kita ingin menggunakan salah satu perintah namun lupa sintaks penulisannya, maka kita dapat menggunakan **\h <nama_perintah>** dan PostgreSQL akan menampilkan sintaks penulisan perintah.

```
postgres=# \h
Available help :
  ABORT                            CREATE OPERATOR CLASS      END
  ALTER AGGREGATE                  CREATE OPERATOR            EXECUTE
  ALTER CONVERSION                 CREATE ROLE                EXPLAIN
  ALTER DATABASE                   CREATE RULE                FETCH
  ALTER DOMAIN                     CREATE SCHEMA              GRANT
  ALTER FUNCTION                   CREATE SEQUENCE            INSERT
  ALTER GROUP                      CREATE TABLE               LISTEN
  ALTER INDEX                      CREATE TABLE AS            LOAD
  ALTER LANGUAGE                   CREATE TABLESPACE          LOCK
  ALTER OPERATOR CLASS             CREATE TRIGGER             MOVE
  ALTER OPERATOR                   CREATE TYPE                NOTIFY
  ALTER ROLE                       CREATE USER                PREPARE
  ALTER SCHEMA                     CREATE VIEW                PREPARE TRANSACTION
  ALTER SEQUENCE                   DEALLOCATE                 REASSIGN OWNED
  ALTER TABLE                      DECLARE                    REINDEX
  ALTER TABLESPACE                 DELETE                     RELEASE SAVEPOINT
  ALTER TRIGGER                    DROP AGGREGATE             RESET
  ALTER TYPE                       DROP CAST                  REVOKE
  ALTER USER                       DROP CONVERSION            ROLLBACK
  ANALYZE                          DROP DATABASE              ROLLBACK PREPARED
  BEGIN                            DROP DOMAIN                ROLLBACK TO SAVEPOINT
  CHECKPOINT                       DROP FUNCTION              SAVEPOINT
  CLOSE                            DROP GROUP                 SELECT
  CLUSTER                          DROP INDEX                 SELECT INTO
  COMMENT                          DROP LANGUAGE              SET
  COMMIT                           DROP OPERATOR CLASS        SET CONSTRAINTS
  COMMIT PREPARED                  DROP OPERATOR              SET ROLE
  COPY                             DROP OWNED           SET SESSION AUTHORIZATION
  CREATE AGGREGATE                 DROP ROLE                  SET TRANSACTION
  CREATE CAST                      DROP RULE                  SHOW
  CREATE CONSTRAINT TRIGGER        DROP SCHEMA                START TRANSACTION
  CREATE CONVERSION                DROP SEQUENCE              TRUNCATE
  CREATE DATABASE                  DROP TABLE                 UNLISTEN
  CREATE DOMAIN                    DROP TABLESPACE            UPDATE
  CREATE FUNCTION                  DROP TRIGGER               VACUUM
  CREATE GROUP                     DROP TYPE                  VALUES
  CREATE INDEX                     DROP USER
  CREATE LANGUAGE                  DROP VIEW
```

G. PERINTAH YANG BERHUBUNGAN DENGAN PSQL COMMAND

Berikut ini daftar perintah yang berhubungan dengan penggunaan PSQL (Prosedural SQL) Pada PostgreSQL 8.2. Perintah-perintah di bawah ini bertujuan untuk memudahkan database administrator dalam mengelola basis data dengan PostgreSQL.

```
postgres=# \?
General
  \c                      [connect] [DBNAME|- USER|- HOST|- PORT|-]
                          connect to new database (currently "postgres")
  \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
  \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
  \db [PATTERN]           list tablespaces (add "+" for more detail)
  \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)
\dg [PATTERN]	list groups
\dn [PATTERN]	list schemas (add "+" for more detail)
\do [NAME]	list operators
\dl	list large objects, same as \lo_list
\dp [PATTERN]	list table, view, and sequence 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, view, and sequence 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| numericlocale|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 LOBOID FILE \lo_import FILE [COMMENT] \lo_list \lo_unlink LOBOID	large object operations

H. TIPE DATA PADA POSTGRESQL

Untuk melihat tipe data yang didikung PostgreSQL, maka dapat menulis perintah berikut ini pada console PostgreSQL.

postgres=# \dT

```
                        List of data types
------------------------------------------------------------------------
   Schema   |        Name        |               Description
------------+--------------------+----------------------------------------
 pg_catalog | "any"              |
 pg_catalog | "char"             | single character
 pg_catalog | "trigger"          |
 pg_catalog | "unknown"          |
 pg_catalog | abstime            | absolute, limited-range date and time
 pg_catalog | aclitem            | access control list
 pg_catalog | anyarray           |
 pg_catalog | anyelement         |
 pg_catalog | bigint             | ~18 digit integer, 8-byte storage
 pg_catalog | bit                | fixed-length bit string
 pg_catalog | bit varying        | variable-length bit string
 pg_catalog | boolean            | boolean, 'true'/'false'
 pg_catalog | box                | geometric box '(lower left,upper right)'
 pg_catalog | bytea              | variable-length string, binary values
 pg_catalog | character          | char(length), blank-padded string,
                                   fixed storage length
 pg_catalog | character varying| varchar(length), non-blank-padded
                                   string, variable storage length
 pg_catalog | cid                | command identifier type, sequence in
                                   transaction id
 pg_catalog | cidr               | networkIP address/netmask,networkaddress
 pg_catalog | circle             | geometric circle '(center,radius)'
 pg_catalog | cstring            |
 pg_catalog | date               | ANSI SQL date
 pg_catalog | double precision   | double-precision floating point
                                   number, 8-byte storage
 pg_catalog | inet               | IP address/netmask, host address,
                                   netmask optional
 pg_catalog | int2vector         | array of int2, used in system tables
 pg_catalog | integer            | -2 billion to 2 billion integer, 4-byte
 pg_catalog | internal           |
 pg_catalog | interval           | @ <number> <units>, time interval
 pg_catalog | language_handler   |
 pg_catalog | line               | geometric line (not implemented)'
 pg_catalog | lseg               | geometric line segment '(pt1,pt2)'
 pg_catalog | macaddr            | XX:XX:XX:XX:XX:XX, MAC address
 pg_catalog | money              | monetary amounts, $d,ddd.cc
 pg_catalog | name               | 63-character type for storing system id
 pg_catalog | numeric            | numeric(precision, decimal), arbitrary
 pg_catalog | oid                | object identifier(oid), maximum 4billion
 pg_catalog | oidvector          | array of oids, used in system tables
 pg_catalog | opaque             |
 pg_catalog | path               | geometric path '(pt1,...)'
 pg_catalog | point              | geometric point '(x, y)'
 pg_catalog | polygon            | geometric polygon '(pt1,...)'
 pg_catalog | real               | single-precision floating point
```

```
                                  number, 4-byte storage
 pg_catalog | record           |
 pg_catalog | refcursor       | reference cursor (portal name)
 pg_catalog | regclass        | registered class
 pg_catalog | regoper         | registered operator
 pg_catalog | regoperator     | registered operator (with args)
 pg_catalog | regproc         | registered procedure
 pg_catalog | regprocedure    | registered procedure (with args)
 pg_catalog | regtype         | registered type
 pg_catalog | reltime         | relative, limited-range time interval
 pg_catalog | smallint        | -32 thousand to 32 thousand, 2-byte
 pg_catalog | smgr            | storage manager
 pg_catalog | text            | variable-length string, nolimitspecified
 pg_catalog | tid             | (Block,offset),physical location oftuple
 pg_catalog | time with time zone         | hh:mm:ss, ANSI SQL time
 pg_catalog | time without time zone      | hh:mm:ss, ANSI SQL time
 pg_catalog | timestamp with time zone    | date and time with time zone
 pg_catalog | timestamp without time zone | date and time
 pg_catalog | tinterval       | (abstime,abstime), time interval
 pg_catalog | void            |
 pg_catalog | xid             | transaction id
(61 rows)
```

Terlihat tipe data yang didukung jumlahnya ada 61 buah. Tipe data di atas merupakan tipe data tambahan dari tipe data default yang didukung PostgreSQL berdasarkan Standar ANSI 92. Jadi tidak terlihat tipe data seperti VARCHAR, VARCHAR(2) atau INT. Karena secara default PostgreSQL sudah mendukung hampir semua tipe data yang digunakan pada DBMS modern pada umumnya.

I. IMPLEMENTASI DDL PADA POSTGRESQL

Dalam *Data Definition Language* (DDL) perintah yang biasa digunakan seperti CREATE, DROP dan ALTER. Adapun penjelasan singkatnya sebagai berikut :

- CREATE, merupakan perintah yang digunakan untuk membuat struktur objek pada database, yang dapat berupa database, table, view, procedure, trigger dan sebagainya.
- DROP, merupakan perintah yang digunakan untuk menghapus struktur objek pada database.
- ALTER, merupakan perintah yang digunakan untuk mengubah struktur objek yang telah ada pada database.

CREATE

1. DATABASE

   Sintaks Membuat Database

```
CREATE DATABASE name
    [ [ WITH ] [ OWNER [=] dbowner ]
    [ TEMPLATE [=] template ]
    [ ENCODING [=] encoding ]
    [ TABLESPACE [=] tablespace ]
    [ CONNECTION LIMIT [=] connlimit ] ]
```

   Masuk ke console PosgtresSQL, lalu ketik :

```
postgres=# CREATE DATABASE Penjualan
postgres-# WITH OWNER=postgres
postgres-# ENCODINg='SQL-ASCII'
postgres-# TABLESPACE=pg_default;
CREATE DATABASE
```

   Melihat database yang telah dibuat :

```
postgres=# \l
           List of databases
       Name       |  Owner   | Encoding
------------------+----------+-----------
 mahasiswa        | postgres | SQL_ASCII
 penjualan        | postgres | SQL_ASCII
 postgres         | postgres | SQL_ASCII
 template0        | postgres | SQL_ASCII
 template1        | postgres | SQL_ASCII
 template_postgis | postgres | SQL_ASCII
(6 rows)
```

2. TABLE

Sintaks Membuat Tabel

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name ( [
  { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE parent_table [ { INCLUDING | EXCLUDING } { DEFAULTS | CONSTRAINTS } ] ... }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace ]
where column_constraint is:
[ CONSTRAINT constraint_name ]
{ NOT NULL |
  NULL |
  UNIQUE index_parameters |
  PRIMARY KEY index_parameters |
  CHECK ( expression ) |
  REFERENCES reftable [ ( refcolumn ) ] [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE
]
    [ ON DELETE action ] [ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
and table_constraint is:
[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] ) index_parameters |
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |
  CHECK ( expression ) |
  FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]
    [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON
UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
index_parameters in UNIQUE and PRIMARY KEY constraints are:
[ WITH ( storage_parameter [= value] [, ... ] ) ]
[ USING INDEX TABLESPACE tablespace
```

Masuk ke console PosgtresSQL, lalu ketik :

```
postgres=# \c penjualan
You are now connected to database "penjualan".
```

Membuat Tabel Barang

```
penjualan=# CREATE TABLE barang
penjualan-# (
penjualan(# KodeBarang CHAR(5) PRIMARY KEY,
penjualan(# NamaBarang VARCHAR(50),
penjualan(# Harga FLOAT,
penjualan(# Keterangan VARCHAR(20)
penjualan(# );
```

```
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"barang_pkey" for
 table "barang"
CREATE TABLE
```

Melihat Struktur Tabel Barang

```
penjualan=# \d barang
             Table "public.barang"
   Column   |         Type          | Modifiers
------------+-----------------------+-----------
 kodebarang | character(5)          | not null
 namabarang | character varying(50) |
 harga      | double precision      |
 keterangan | character varying(20) |
Indexes:
    "barang_pkey" PRIMARY KEY, btree (kodebarang)
```

Membuat Tabel Pelanggan

```
penjualan=# CREATE TABLE pelanggan
penjualan-# (
penjualan(# KodePelanggan CHAR(5) PRIMARY KEY,
penjualan(# NamaPelanggan VARCHAR(50),
penjualan(# Alamat VARCHAR(50),
penjualan(# ContactPerson VARCHAR(20),
penjualan(# Telepon VARCHAR(20)
penjualan(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"pelanggan_pkey"
for table "pelanggan"
CREATE TABLE
```

Melihat Struktur Tabel Pelanggan

```
penjualan=# \d pelanggan
             Table "public.pelanggan"
    Column     |         Type          | Modifiers
---------------+-----------------------+-----------
 kodepelanggan | character(5)          | not null
 namapelanggan | character varying(50) |
 alamat        | character varying(50) |
 contactperson | character varying(20) |
 telepon       | character varying(20) |
Indexes:
    "pelanggan_pkey" PRIMARY KEY, btree (kodepelanggan)
```

Membuat Tabel JualHeader

```
penjualan=# CREATE TABLE jualheader
penjualan-# (
penjualan(# NoFaktur CHAR(5) PRIMARY KEY,
penjualan(# Tanggal DATE,
penjualan(# KodePelanggan CHAR(5),
penjualan(# SubTotal FLOAT,
penjualan(# Diskon FLOAT,
```

```
penjualan(# Pajak FLOAT,
penjualan(# GrandTotal FLOAT
penjualan(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"jualheader_pkey"
 for table "jualheader"
```

Melihat Struktur Tabel JualHeader

```
penjualan=# \d jualdetail
           Table "public.jualdetail"
   Column   |         Type          | Modifiers
------------+-----------------------+-----------
 nofaktur   | character(5)          |
 kodebarang | character(5)          |
 namabarang | character varying(50) |
 harga      | double precision      |
 qty        | smallint              |
 total      | double precision      |
```

Membuat Tabel JualDetail

```
CREATE TABLE
penjualan=# CREATE TABLE jualdetail
penjualan-# (
penjualan(# NoFaktur CHAR(5),
penjualan(# KodeBarang CHAR(5),
penjualan(# NamaBarang VARCHAR(50),
penjualan(# Harga FLOAT,
penjualan(# QTY SMALLINT,
penjualan(# Total FLOAT
penjualan(# );
CREATE TABLE
```

Melihat Struktur Tabel JualDetail

```
penjualan=# \d jualheader
          Table "public.jualheader"
    Column     |       Type       | Modifiers
---------------+------------------+-----------
 nofaktur      | character(5)     | not null
 tanggal       | date             |
 kodepelanggan | character(5)     |
 subtotal      | double precision |
 diskon        | double precision |
 pajak         | double precision |
 grandtotal    | double precision |
Indexes:
    "jualheader_pkey" PRIMARY KEY, btree (nofaktur)
```

3. VIEW

Sintaks Membuat View

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] VIEW name [ ( column_name [, ...] ) ]
  AS query Statement;
```

Masuk ke console PosgtresSQL, lalu ketik :

Membuat ViewPenjualan

```
penjualan=# CREATE VIEW viewjual AS
penjualan-# (
penjualan(# SELECT H.NoFaktur, H.Tanggal, H.KodePelanggan,
penjualan(# P.NamaPelanggan, D.KodeBarang, D.NamaBarang,
penjualan(# D.Harga, D.QTY, D.Total
penjualan(# FROM Barang B, Pelanggan P, JualHeader H, JualDetail
D
penjualan(# WHERE B.KodeBarang=D.KodeBarang AND
penjualan(# D.NoFaktur=H.Nofaktur AND
penjualan(# H.KodePelanggan=P.KodePelanggan
penjualan(# );
CREATE VIEW
```

Melihat Struktur ViewPenjualan

```
penjualan=# \d viewjual
              View "public.viewjual"
    Column     |          Type          | Modifiers
---------------+------------------------+-----------
 nofaktur      | character(5)           |
 tanggal       | date                   |
 kodepelanggan | character(5)           |
 namapelanggan | character varying(50)  |
 kodebarang    | character(5)           |
 namabarang    | character varying(50)  |
 harga         | double precision       |
 qty           | smallint               |
 total         | double precision       |
```

DROP

Untuk menghapus Objek, maka perintah yang dapat digunakan adalah :

NAMA OBJEK	SINTAKS
Database	DROP DATABASE <nama_database>
Table	DROP TABLE <nama_table>
View	DROP VIEW <nama_view>
Index	DROP INDEX <nama_index>
Procedure	DROP PROCEDURE <nama_procedure>
Trigger	DROP TRIGGER <nama_triger>

ALTER

Alter pada PostgreSQL ada 18 jenis. Namun kita hanya membatasi pada ALTER TABLE. Berikut sintaks ALTER TABLE :

```
ALTER TABLE [ ONLY ] name [ * ]
    action [, ... ]
ALTER TABLE [ ONLY ] name [ * ]
    RENAME [ COLUMN ] column TO new_column
ALTER TABLE name
    RENAME TO new_name
ALTER TABLE name
    SET SCHEMA new_schema
Where Action Is One Of :
    ADD [ COLUMN ] column type [ column_constraint [ ... ] ]
    DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
    ALTER [ COLUMN ] column TYPE type [ USING expression ]
    ALTER [ COLUMN ] column SET DEFAULT expression
    ALTER [ COLUMN ] column DROP DEFAULT
    ALTER [ COLUMN ] column { SET | DROP } NOT NULL
    ALTER [ COLUMN ] column SET STATISTICS integer
    ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
    ADD table_constraint
    DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
    DISABLE TRIGGER [ trigger_name | ALL | USER ]
    ENABLE TRIGGER [ trigger_name | ALL | USER ]
    CLUSTER ON index_name
    SET WITHOUT CLUSTER
    SET WITHOUT OIDS
    SET ( storage_parameter = value [, ... ] )
    RESET ( storage_parameter [, ... ] )
    INHERIT parent_table
    NO INHERIT parent_table
    OWNER TO new_owner
    SET TABLESPACE new_tablespace
```

Masuk ke console PosgtresSQL, lalu ketik :

Menambah Constraint Pada Tabel JualDetail :

```
penjualan=# ALTER TABLE JualDetail
penjualan-# ADD CONSTRAINT FK_JualDetail FOREIGN KEY (NoFaktur)
penjualan-# REFERENCES JualHeader (NoFaktur);
ALTER TABLE
```

Menambah Column Pada Tabel Barang :

```
penjualan=# ALTER TABLE Barang
penjualan-# ADD STOK SMALLINT DEFAULT NULL;
ALTER TABLE
```

J. IMPLEMENTASI DML PADA POSTGRESQL

Data Manipulation Language terdiri dari perintah : SELECT, INSERT, UPDATE dan DELETE. Biasanya perintah DML dilakukan terhadap tabel atau view dalam database MySQL. Adapun penjelasan singkatnya adalah sebagai berikut :

- SELECT, merupakan perintah yang digunakan untuk menampilkan data yang berasal dari tabel atau view.
- INSERT, merupakan perintah yang digunakan untuk memasukkan data atau record ke dalam tabel.
- UPDATE, merupakan perintah yang digunakan untuk memperbarui data atau record pada tabel.
- DELETE, merupakan perintah yang digunakan untuk menghapus data atau record yang ada pada tabel.

SELECT

Sintaks Melakukan Select Pada Tabel :

```
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 | SHARE } [ OF table_name [, ...] ] [ NOWAIT ] [...] ]

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 [, ...] ) ]
```

Tampilkan semua data yang harganya lebih dari 5000000 dari tabel Barang

```
penjualan=# SELECT * FROM barang
penjualan-# WHERE harga > 5000000;
 kodebarang |     namabarang      |  harga  | keterangan
------------+---------------------+---------+------------
 B0003      | HIS ATI X1950XTX    | 7000000 | VGA Card
 B0004      | ASUS NVIDIA 7950GX2 | 6500000 | VGA Card
(2 rows)
```

Tampilkan semua data harganya antara 1000000 sampai 5000000 dari tabel Barang

```
penjualan=# SELECT * FROM Barang
penjualan-# WHERE harga BETWEEN 1000000 AND 3000000;
 kodebarang |     namabarang      |  harga  | keterangan
------------+---------------------+---------+-------------
 B0005      | MSI A9N Deluxe AM2  | 2300000 | Motherboard
 B0006      | ASUS P5 WAD2 Deluxe | 3000000 | Motherboard
 B0007      | Seagate SATA 200 GB | 1500000 | Harddisk
 B0008      | Maxtor SATA 200 GB  | 1600000 | Harddisk
 B0009      | Kingston DDR2 1 GB  | 1100000 | Memory
 B0010      | Corsair DDR2 1 GB   | 1300000 | Memory
(6 rows)
```

Tampilkan semua data yang NamaBarangnya mengandung kata IN

```
penjualan=# SELECT * FROM Barang
penjualan-# WHERE namabarang LIKE '%in%';
 kodebarang |     namabarang     |  harga  | keterangan
------------+--------------------+---------+------------
 B0009      | Kingston DDR2 1 GB | 1100000 | Memory
(1 row)
```

Tampilkan Keterangan Sebagai JenisBarang Tanpa Redundant

```
penjualan=# SELECT DISTINCT(Keterangan) AS JenisBarang
penjualan-# FROM Barang;
 jenisbarang
-------------
 Harddisk
 Memory
 Motherboard
 Processor
 VGA Card
(5 rows)
```

Tampilkan semua data yang NamaBarangnya huruf depannya adalah A

```
penjualan=# SELECT * FROM barang
penjualan-# WHERE namabarang LIKE 'A%';
 kodebarang |      namabarang      |  harga  | keterangan
------------+----------------------+---------+-------------
 B0002      | AMD Athlon AM2 4000+ | 4000000 | Processor
 B0004      | ASUS NVIDIA 7950GX2  | 6500000 | VGA Card
 B0006      | ASUS P5 WAD2 Deluxe  | 3000000 | Motherboard
(3 rows)
```

INSERT

Sintaks Melakukan Insert Pada Tabel :

```
INSERT INTO table [ ( column [, ...] ) ]
  { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | que
ry }
  [ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

Masukan Data Pada Tabel Barang :

```
penjualan=# INSERT INTO Barang VALUES
penjualan-# ('B0001','Intel Dual Core 3.4 GHz',3200000,'Processor'),
penjualan-# ('B0002','AMD Athlon AM2 4000+',4000000,'Processor'),
penjualan-# ('B0003','HIS ATI X1950XTX',7000000,'VGA Card'),
penjualan-# ('B0004','ASUS NVIDIA 7950GX2',6500000,'VGA Card'),
penjualan-# ('B0005','MSI A9N Deluxe AM2',2300000,'Motherboard'),
penjualan-# ('B0006','ASUS P5 WAD2 Deluxe',3000000,'Motherboard'),
penjualan-# ('B0007','Seagate SATA 200 GB',1500000,'Harddisk'),
penjualan-# ('B0008','Maxtor SATA 200 GB',1600000,'Harddisk'),
penjualan-# ('B0009','Kingston DDR2 1 GB',1100000,'Memory'),
penjualan-# ('B0010','Corsair DDR2 1 GB',1300000,'Memory');
INSERT 0 10

penjualan=# select * from barang;
 kodebarang |       namabarang        |  harga  | keterangan
------------+-------------------------+---------+-------------
 B0001      | Intel Dual Core 3.4 GHz | 3200000 | Processor
 B0002      | AMD Athlon AM2 4000+    | 4000000 | Processor
 B0003      | HIS ATI X1950XTX        | 7000000 | VGA Card
 B0004      | ASUS NVIDIA 7950GX2     | 6500000 | VGA Card
 B0005      | MSI A9N Deluxe AM2      | 2300000 | Motherboard
 B0006      | ASUS P5 WAD2 Deluxe     | 3000000 | Motherboard
 B0007      | Seagate SATA 200 GB     | 1500000 | Harddisk
 B0008      | Maxtor SATA 200 GB      | 1600000 | Harddisk
 B0009      | Kingston DDR2 1 GB      | 1100000 | Memory
 B0010      | Corsair DDR2 1 GB       | 1300000 | Memory
(10 rows)
```

Masukan Data Pada Tabel Pelanggan

```
penjualan=# INSERT INTO Pelanggan VALUES
penjualan-# ('P0001','PT.Sumbawa Indah','Jakarta','Shinta','08123701639'),
penjualan-# ('P0002','PT.Alam Raya','Bogor','Esabella','08569787865'),
penjualan-# ('P0003','PT.Mekar Sari','Bekasi','Rini','08138765432'),
penjualan-# ('P0004','PT.Sejahtera','Jakarta','Andri','08529876578'),
penjualan-# ('P0005','PT.Kost Palm','Jakarta','Nova','08120897776'),
penjualan-# ('P0006','PT.Maju Mundur','Tangerang','Tomi','08658765478'),
penjualan-# ('P0007','PT.Senyum Selalu','Bogor','Andi','08179876876'),
penjualan-# ('P0008','PT.Untung Terus','Bekasi','Bella','08188768567'),
penjualan-# ('P0009','PT.Surya Tenggelam','Depok','Riyani','08157887654'),
penjualan-# ('P0010','PT.Kesepian Kita','Tangerang','Budi','08190987687');
INSERT 0 10

penjualan=# Select * From Pelanggan;
 kodepelanggan |   namapelanggan    |  alamat   | contactperson |   telepon
---------------+--------------------+-----------+---------------+-------------
 P0001         | PT.Sumbawa Indah   | Jakarta   | Shinta        | 08123701639
 P0002         | PT.Alam Raya       | Bogor     | Esabella      | 08569787865
 P0003         | PT.Mekar Sari      | Bekasi    | Rini          | 08138765432
```

```
 P0004          | PT.Sejahtera        | Jakarta   | Andri          | 08529876578
 P0005          | PT.Kost Palm        | Jakarta   | Nova           | 08120897776
 P0006          | PT.Maju Mundur      | Tangerang | Tomi           | 08658765478
 P0007          | PT.Senyum Selalu    | Bogor     | Andi           | 08179876876
 P0008          | PT.Untung Terus     | Bekasi    | Bella          | 08188768567
 P0009          | PT.Surya Tenggelam | Depok     | Riyani         | 08157887654
 P0010          | PT.Kesepian Kita    | Tangerang | Budi           | 08190987687
(10 rows)
```

UPDATE

Sintaks Melakukan Update Pada Tabel :

```
UPDATE [ ONLY ] table [ [ AS ] alias ]
   SET { column = { expression | DEFAULT } |
        ( column [, ...] ) = ( { expression | DEFAULT } [, ...] ) } [, ...]
   [ FROM fromlist ]
   [ WHERE condition ]
   [ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

Ubah harga barang yang kodebarangnya = B0001

```
penjualan=# UPDATE barang
penjualan-# SET harga=500000
penjualan-# WHERE kodebarang='B0001';
UPDATE 1

penjualan=# SELECT * FROM barang
penjualan-# WHERE kodebarang='B0001';
 kodebarang |       namabarang        | harga  | keterangan
------------+-------------------------+--------+------------
 B0001      | Intel Dual Core 3.4 GHz | 500000 | Processor
(1 row)
```

Ubah contactperson = Esabella yang Alamatnya = Jakarta

```
penjualan=# UPDATE pelanggan
penjualan-# SET contactperson='Esabella'
penjualan-# WHERE alamat='Jakarta';
UPDATE 3

penjualan=# SELECT * FROM pelanggan
penjualan-# WHERE alamat='Jakarta';
 kodepelanggan |  namapelanggan   | alamat  | contactperson |   telepon
---------------+-----------------+---------+---------------+------------
 P0001         | PT.Sumbawa Indah | Jakarta | Esabella      | 08123701639
 P0004         | PT.Sejahtera     | Jakarta | Esabella      | 08529876578
 P0005         | PT.Kost Palm     | Jakarta | Esabella      | 08120897776
(3 rows)
```

DELETE

Sintaks Melakukan Delete Pada Tabel :

```
DELETE FROM [ ONLY ] table [ [ AS ] alias ]
  [ USING usinglist ]
  [ WHERE condition ]
  [ RETURNING * | output_expression [ AS output_name ] [, ...] ]
```

Hapus Record dari tabel barang yang harganya < 3000000

```
penjualan=# DELETE FROM Barang
penjualan-# WHERE harga < 3000000;
DELETE 6
penjualan=# SELECT * FROM barang;
 kodebarang |      namabarang       |  harga   | keterangan
------------+-----------------------+----------+-------------
 B0002      | AMD Athlon AM2 4000+  | 4000000  | Processor
 B0003      | HIS ATI X1950XTX      | 7000000  | VGA Card
 B0004      | ASUS NVIDIA 7950GX2   | 6500000  | VGA Card
 B0006      | ASUS P5 WAD2 Deluxe   | 3000000  | Motherboard
(4 rows)
```

Hapus Record dari tabel pelanggan yang alamtnya = Tangerang

```
penjualan=# DELETE FROM Pelanggan
penjualan-# WHERE alamat = 'Tangerang';
DELETE 2
penjualan=# SELECT * FROM pelanggan;
 kodepelanggan |   namapelanggan    | alamat  | contactperson |   telepon
---------------+--------------------+---------+---------------+-------------
 P0002         | PT.Alam Raya       | Bogor   | Esabella      | 08569787865
 P0003         | PT.Mekar Sari      | Bekasi  | Rini          | 08138765432
 P0007         | PT.Senyum Selalu   | Bogor   | Andi          | 08179876876
 P0008         | PT.Untung Terus    | Bekasi  | Bella         | 08188768567
 P0009         | PT.Surya Tenggelam | Depok   | Riyani        | 08157887654
 P0001         | PT.Sumbawa Indah   | Jakarta | Esabella      | 08123701639
 P0004         | PT.Sejahtera       | Jakarta | Esabella      | 08529876578
 P0005         | PT.Kost Palm       | Jakarta | Esabella      | 08120897776
(8 rows)
```

K. IMPLEMENTASI DCL PADA POSTGRESQL

Data Control Language merupakan kontrol keamanan terhadap database dan tabelnya, yaitu mengatur hak akses dengan cara mencabut hak akses atau memberi hak akses pada user, tujuannya agar tabel-tabel tertentu hanya bisa diakses oleh orang-orang yang dikehendaki. DCL terdiri dari perintah GRANT dan REVOKE. Adapun penjelasan singkatnya adalah sebagai berikut :

- GRANT, merupakan perintah yang digunakan untuk mengizinkan seorang user untuk mengakses tabel pada database tertentu.
- REVOKE, merupakan perintah yang digunakan untuk mencabut hak akses seorang user pada tabel dalam database tertentu.

Memberikan Hak Akses

Seorang administrator database biasanya akan melakukan pembatasan hak akses user terhadap tabel dalam database. Tujuan akhirnya adalah untuk manajemen keamanan database. Hak Akses adalah hak-hak yang diberikan server administrator kepada user, antara lain : ALTER, CREATE, DELETE, DROP, UPDATE, INSERT, FILE, PROCESS, RELOAD, REFERENCES,LOAD, SHUTDOWN DAN USAGE. Pemakai adalah nama user yang akan diberi hak, dengan ketentuan nama pemakai diikuti nama dari host diawali tanda @.

Adapun sintaks GRANT adalah sebagai berikut :

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRIGGER }
   [,...] | ALL [ PRIVILEGES ] }
   ON [ TABLE ] tablename [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
   [,...] | ALL [ PRIVILEGES ] }
   ON SEQUENCE sequencename [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
   ON DATABASE dbname [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
   ON FUNCTION funcname ( [ [ argmode ] [ argname ] argtype [, ...] ] ) [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
   ON LANGUAGE langname [, ...]
```

```
TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
   ON SCHEMA schemaname [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] }
   ON TABLESPACE tablespacename [, ...]
   TO { username | GROUP groupname | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT role [, ...] TO username [, ...] [ WITH ADMIN OPTION ]
```

Berikan seluruh hak akses pada user Shinta atas database penjualan

```
postgres=# GRANT ALL PRIVILEGES
postgres-# ON DATABASE penjualan
postgres-# TO shinta;
GRANT
```

Mencabut Hak Akses

REVOKE digunakan untuk mencabut hak akses seorang user mengakses tabel dalam database tertentu. Pencabutan hak akses ini dengan klausa REVOKE. Pemakai adalah nama user yang akan dicabut hak aksesnya, dengan ketentuan nama pemakai diikuti nama dari host diawali tanda @.

Adapun sintaks REVOKE adalah sebagai berikut :

```
REVOKE [ GRANT OPTION FOR ]
   { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRIGGER }
   [,...] | ALL [ PRIVILEGES ] }
   ON [ TABLE ] tablename [, ...]
   FROM { username | GROUP groupname | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { USAGE | SELECT | UPDATE }
   [,...] | ALL [ PRIVILEGES ] }
   ON SEQUENCE sequencename [, ...]
   FROM { username | GROUP groupname | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
   { { CREATE | CONNECT | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
   ON DATABASE dbname [, ...]
   FROM { username | GROUP groupname | PUBLIC } [, ...]
   [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
    { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTION funcname ( [ [ argmode ] [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | GROUP groupname | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE langname [, ...]
    FROM { username | GROUP groupname | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schemaname [, ...]
    FROM { username | GROUP groupname | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespacename [, ...]
    FROM { username | GROUP groupname | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]

REVOKE [ ADMIN OPTION FOR ]
    role [, ...] FROM username [, ...]
    [ CASCADE | RESTRICT ]
```

Cabut semua hak akses dari user Shinta atas database penjualan

```
postgres=# REVOKE ALL PRIVILEGES
postgres-# ON DATABASE PENJUALAN
postgres-# FROM SHINTA;
REVOKE
```

L. IMPLEMENTASI FUNGSI AGREGAT PADA DATABASE PENJUALAN

Selain mengambil data dengan kriteria tertentu, sering juga diperlukan berbagai perhitungan yang bersifat ringkasan. Fungsi agregat merupakan sekumpulan fungsi yang siap dipakai untuk mendapatkan hasil penjumlahan, penghitungan frekuensi, rata-rata, dan lain-lain.

FUNGSI AGREGAT	KETERANGAN
AVG	Memperoleh Nilai Rata-Rata
COUNT	Menghitung Cacah Data
MAX	Mencari Nilai Terbesar
MIN	Mencari Nilai Terkecil
SUM	Memperoleh Penjumlahan Data

Tabel Fungsi Agregat

1. AVG

Mencari nilai rata-rata harga barang dari tabel barang

```
penjualan=# SELECT AVG(HARGA) AS HARGA_RATA2
penjualan-# FROM BARANG;
  PENJUALAN
 harga_rata2
-------------
     3150000
(1 row)
```

2. COUNT

Mencari jumlah Record dari tabel pelanggan

```
penjualan=# SELECT COUNT(*) AS JUMLAH_RECORD
penjualan-# FROM PELANGGAN;
   PENJUALAN
 jumlah_record
---------------
            10
(1 row)
```

3. MAX

Mencari harga tertinggi dari tabel barang

```
penjualan=# SELECT MAX(HARGA) AS HARGA_TERTINGGI
penjualan-# FROM BARANG;
    PENJUALAN
 harga_tertinggi
-----------------
         7000000
(1 row)
```

4. MIN

Mencari harga terendah dari tabel barang

```
penjualan=# SELECT MIN(HARGA) AS HARGA_TERENDAH
penjualan-# FROM BARANG;
   PENJUALAN
 harga_terendah
----------------
        1100000
(1 row)
```

5. SUM

Mencari jumlah penjualan dari masing-masing keterangan/jenis dari tabel barang

```
penjualan=# SELECT DISTINCT(KETERANGAN) AS JENIS,
penjualan-# SUM(HARGA) AS HARGA_BARANG
penjualan-# FROM BARANG
penjualan-# GROUP BY KETERANGAN;
         PENJUALAN
    jenis     | harga_barang
--------------+--------------
 Harddisk     |      3100000
 Memory       |      2400000
 Motherboard  |      5300000
 Processor    |      7200000
 VGA Card     |     13500000
(5 rows)
```

M. DAFTAR PUSTAKA

- Kadir, Abdul. 2002. **Penuntun Praktis Belajar SQL**. Yogyakarta : Penerbit Andi.
- Manual Books PostgreSQL 8.2.0
- Nugroho, Adi. 2004. **Konsep Pengembangan Sistem Basis Data**. Bandung : Penerbit Informatika.
- www.postgresql.org
- http://www.bogor.net/idkf/idkf/aplikasi/linux/postgres-double-A4.pdf