An Introduction to SQL and PostgreSQL

Computing Laboratory

Department of Computing

Ian Moor

The PostgreSQL System

PostgreSQL is a free relational database system supporting the SQL language available on a variety of hardware platforms.

SQL

SQL (Structured Query Language) is the standard relational query language with statements to create, remove, retrieve and maintain data in a relational database. SQL contains three main types of commands, Data Definition language (DDL), Data Manipulation language (DML), and Data Control language (DCL) statements.

The specific version of SQL used by PostgreSQL can be used interactively via the **psql** front-end, embedded in programming languages like C or called from Java, perl and tcl. PostgreSQL supports a subset of SQL3 with extra types and commands.

SQL syntax

- SQL keywords and names contain alpha-numeric characters (including '_') and must begin with a letter or '_' and are case insensitive. A name containing other characters or which is an SQL keyword must be written in double quotes '"'. Column names, table names and database names are separate, so a column can have the same name as the table it is in. Keywords are usually written in uppercase letters.
- The comment syntax is the same as C : /* ...*/, (but // comments are not allowed). Also '`--`' comments out the rest of the line.
- SQL statements end with ';'.

Simple Data Types

Every column in an SQL table must have a type describing the data in it. SQL supports integer data: as INT4 (4 bytes) and INT2 (2 bytes), Floating point data is stored as FLOAT8 (8 bytes) or FLOAT (4 bytes). Character data is held as fixed length strings padded with blanks: CHAR(n) where n is the length. Variable length strings with a size limit are declared as VARCHAR(n). String contants are written in single quotes `'`.

The types DATE,TIME and DATETIME contain date information, time information or both, constants of these types are written as strings, for example:

```
'5th November 2001'
'2001-11-05 10:20'
'5.11.2001 10:20 gmt'
```

Adding or subtracting a date and and integer is a date, the number is a number of days. Subtracting two dates gives the time between the two dates. PostgreSQL also has the type INTERVAL, for a length of time.

Expressions are type checked, and where possible type conversion is used, for example dates can be written as strings without using the conversion function DATE.

NULL

is a special value that represents the fact that a piece of data that is either unknown or undefined. Note that this is **not** zero, or the empty string. Comparisons with NULL return `unknown`, use the expression IS NULL or IS NOT NULL for example:

```
SELECT 'Unknown' AS born ,name FROM actors WHERE born IS NULL ;
```

Data Definition Statements

CREATE TABLE	creates an empty table defining the structure (see the exercise definition).
DROP TABLE	destroys a table and all the data it contains (if you have permission).
ALTER TABLE	change the definition of a table (column names and types etc.)

Data Manipulation Statements

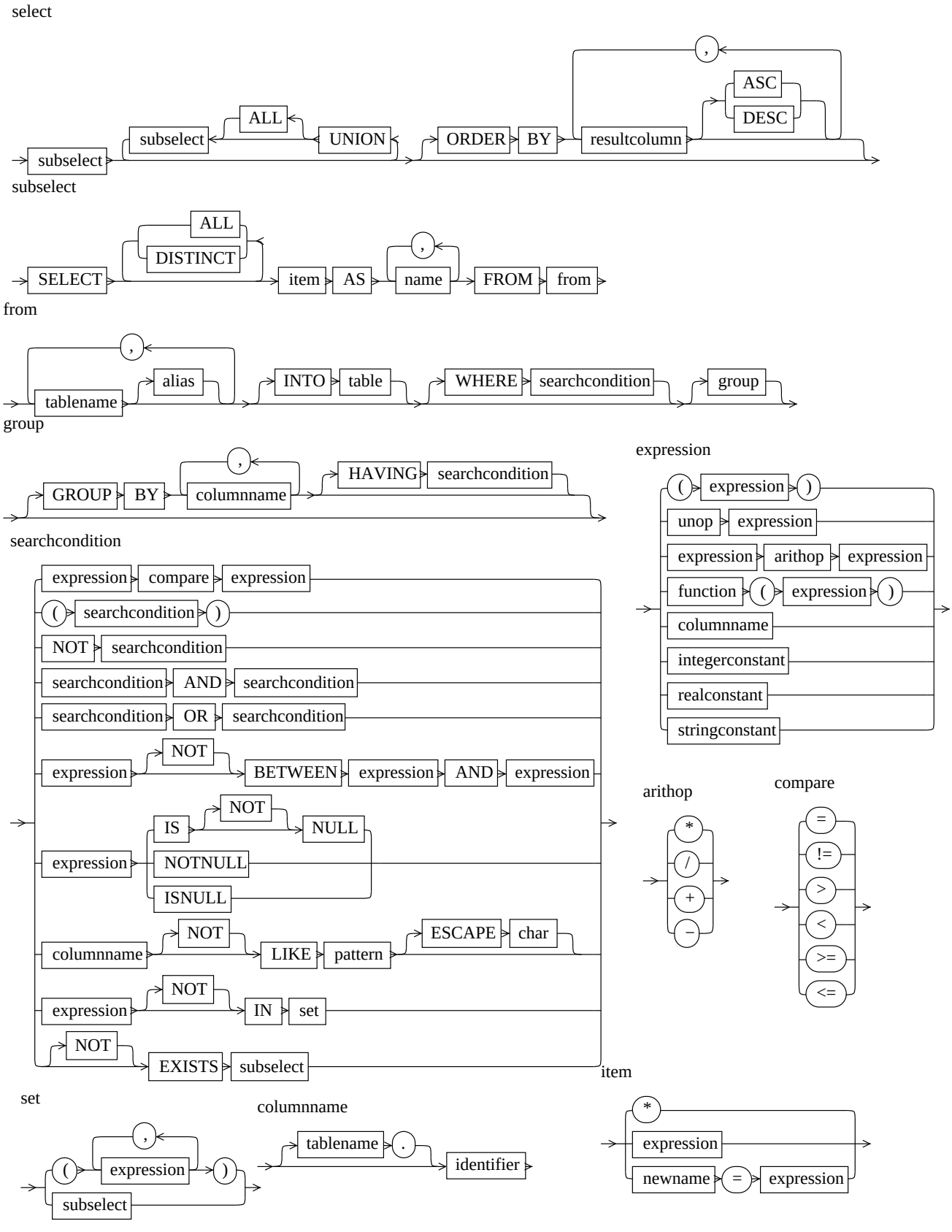
INSERT	add rows or part of rows to a table.
SELECT	perform a query to view some existing data.
UPDATE	modify existing table entries, selecting which rows are changed.
DELETE	remove selected rows from a table.
ROLLBACK	requests that a change be undone.
COMMIT	requests that the database make a permanent record of a change.
COPY	reads or writes data between databases and files.

SELECT : The most used statement

The SELECT statement is the most frequently used statement in SQL, it is used to retrieve data from one or more tables. The names of the tables used in the statement must be listed in the statement. If a column-name appears in more than one table, references to the column must be written as `table.column` to avoid ambiguity. The syntax for SELECT is used in UPDATE and DELETE, for example:

```
SELECT name,pay FROM payroll WHERE name = 'poor man' ;
UPDATE payroll SET pay = pay * 10 WHERE  name = 'poor man' ;
DELETE FROM payroll WHERE pay < 0 ;
```

The railway diagram shows the SQL grammar for select with keywords in uppercase, branches show alternatives in the grammar and loops show repetition. Note that a select statement can contain several sub-selections.



Notes about SELECT

ORDER BY

The result of a select statement is not sorted unless **ORDER BY** is used, when the column(s) to be used for sorting are

given. Ascending order (ASC) is the default, specify DESC for descending order. For example to list films longest first:

```
SELECT title,length FROM films ORDER BY length DESC ;
```

DISTINCT

Duplicate rows in the result of **SELECT** are all shown unless DISTINCT is used. For example to remove duplicate titles:

```
SELECT DISTINCT title FROM Films ;
```

Alias

Tables or columns may have an alias to shorten queries or let the same table to be used several times in a single query. Here `A` is used as a shorthand for the table `actor`

```
SELECT COUNT(A.name) FROM actors A WHERE A.name < 'Zorro' ;
```

GROUP BY

rearranges the original table into logical partitions, in each partition all rows have the same value in the specified column(s). For example to group films by director and give the size of each group:

```
SELECT director,COUNT(director) FROM films GROUP BY director;
```

HAVING

selects groups from the partitions generated by GROUP BY. For example:

```
SELECT part FROM casting GROUP BY part HAVING part > "A" ;
```

Predicates

evaluate to TRUE, FALSE or UNKNOWN. Comparison operations work on dates and times as well as numbers and strings. UNKNOWN is returned if one the operands is NULL.

A pattern

is a string constant that may contain wild-card characters: '%' matches any number of characters, '_' (underscore) matches a single character for example to select any actors name containing o.

```
SELECT name FROM actors WHERE name LIKE '%o%';
```

A set

is either a bracketed list of expressions, or a nested SELECT query. The ability to nest queries gives SELECT much of its power, for example to look for actor-directors:

```
SELECT director FROM films   /* Any actor-directors ? */
WHERE director IN (SELECT name FROM actors );
```

EXISTS

is TRUE if the nested query returns any data, for example:

```
SELECT director FROM films
 WHERE EXISTS ( SELECT name FROM actors WHERE born < '1-jan-1900') ;
```

Running SQL on Postgresql

Use the command **psql** followed by a database name such as **psql films** and type sql statements. Other **psql** commands start with '\'

\? lists all **psql** commands.

\h	provides help for sql statements.
\q	quits **psql**.

Online documentation for **psql** is available by typing (**man psql**).

Embedding SQL in other languages

Almost all applications using databases are written in some other programming language to provide for example a graphical user interface or data structures not supported in SQL. There are several standard interfaces to SQL databases: for example

C

A C program using a database is written with special statements, where C and SQL can be mixed. The C program can pass C values to SQL and get result back. In the case of PostgreSQL the command **ecpg** converts each Sql statement into calls to the special PostgreSQL C library and the program can be compiled and run. For example:

```
exec sql begin declare section;
char * name;
char  * result;
exec sql end declare section;
...
strcpy(name,"Alfred Hitchcock");
exec sql select director into :result  from films where title = :name ;
printf("%s made %s",name,result);
```

perl

Perl has a library (DBI) for accessing databases. Although you have to specify what database system is used (Postgresql, mydb, oracle etc) the functions used are independent of the database type.

```
use DBI;
# connect to the database
my $dsn = "DBI:Pg:dbname=films;host=db;port=5432";
my $dbh = DBI->connect($dsn, "lab", "lab",{ RaiseError => 1});
# create a statement
$example = $dbh -> prepare("SELECT name,born FROM ACTORS" );
# run the statement
$example -> execute();
# fetch and print the results
while ( ($actor,$born) = $example -> fetchrow_array()) {
  print "An actor: $actor born on $born\n";
}
$example -> finish();
$dbh -> disconnect();
```

Java

Java has a standard library for accessing databases, an example:

```
import java.sql.*;

public class JDBCSample {
  public static void main(String[] args) {
    // load the postgresql driver
    try {
      Class.forName("org.postgresql.Driver");
    }
```

```
    catch (ClassNotFoundException e) {
      System.err.println("Can't load the postgresql driver");
      return;
    }

    try {
      // connect to the database
      Connection con =
      DriverManager.getConnection("jdbc:postgresql://db/films","lab","lab");
      // create and run an sql query
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT TITLE FROM FILMS");
      // retrieve the results and print them
     while(rs.next()) {
       System.out.println(rs.getString("TITLE"));
     }

     rs.close();
     stmt.close();
     con.close();
    }
    catch (SQLException se) {
      System.err.println("SQL Exception: "+se.getMessage());
      se.printStackTrace(System.err);
    }
  }
}
```

Bibliography

PostgreSQL tutorial and user guide.

on the postgresql site (http://techdocs.postgresql.org/#techguides)

Local information about postsgresl and other databases .

on the CSG QA page (http://www.doc.ic.ac.uk/csg/faqs/postgresql.html)

The Practical SQL Handbook , Bowman et al, 1993 , *Using Structured Query Language* , 3, Judith Bowman, Sandra Emerson, and Marcy Damovsky, 0-201-44787-8, 1996, Addison-Wesley, 1997.

A Guide to the SQL Standard , Date and Darwen, 1997 , *A user's guide to the standard database language SQL* , 4, C. J. Date and Hugh Darwen, 0-201-96426-0, 1997, Addison-Wesley, 1997.

An Introduction to Database Systems , Date, 1994 , 6, C. J. Date, 1, 1994, Addison-Wesley, 1994.

Understanding the New SQL , Melton and Simon, 1993 , *A complete guide*, Jim Melton and Alan R. Simon, 1-55860-245-3, 1993, Morgan Kaufmann, 1993.