Review on Database SQL skills ISYS2120

Harry

November 16, 2018

Abstract

This note is for reviewing the key SQL skills (postreSQL) as well as some other basic DBMS knowledge involved in unit ISYS2120.

1 Lectrue 1: Intro

The GrokLearning websites for all the SQL exercises and corresponding knowledge points. Weekly lab quizzes which used for testing the corresponding knowledge taught that week by providing feedback.

•**Instance** a snapshot of the contents of a database at a single time.

•**Schema** Describe the structure of a bunch of databases, the instance above must follow the schema at any time.

•**DDL** Data Definition Language (or Data Description language) is for coding the database structure. (e.g. create, drop etc.)

•**DML** Data Manipulation Language is for accessing the data. (e.g. select ... from ... where ...)

2 Lecture 2: SQL and Conceptual Data Design

All the statements below are concerning Data Manipulation Language.

•**Select** Lists the columns (and expressions) that should be returned from the query.

•**From** Indicate the table(s) from which data will be obtained.

•**Where** Indicate the conditions to include a tuple in the result.

•**Group by** Indicate the categorization of tuples.

•**Having** Indicate the conditions to include a category.

•**Order by** Sorts the result according to specified criteria.

An example for the most basic query: **SELECT** name **FROM** Student **WHERE** country = 'AUS' **ORDER BY** name. That means that List all students (name) from Australia in alphabetical order.
The '**ORDER BY**' here can include more than one condition. For example, you can make the result ordered by the ascending student ID when there are some students share the same priority in name order.

The '**DISTINCT**' and '**all**' keywords specify that whether the results should eliminate duplicates if exist. If you use the 'distinct', there will be no duplicates in your result table.

The asterisk in the query means 'all attributes', which usually appear in the selection statement targeting the attribute. (e.g. **SELECT** * **FROM** Student)

The '**AS**' key word denotes that you use another name which you prefer to name the attribute appear in one particular column. (e.g. **SELECT** SID **AS** student_ID **FROM** Student)

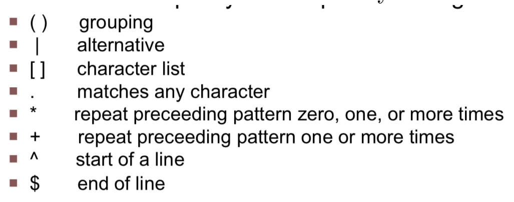
The where statement shows the requirements that the result must satisfy. There are also some comparison operators here, including =,¿,¿=,¡,¡=,!=,¡¿ (the last two are the same). Also, three connectives '**AND**', '**OR**' and '**NOT**'. (e.g. **SELECT** * **FROM** Student **WHERE** gender = 'Male')

The '**BETWEEN**' key word for stating a range. (e.g. **SELECT** * **FROM** Student **WHERE** age **BETWEEN** 18 **AND** 20)

The string operations '**LIKE**' is for matching, its successor should be a string using possibly two symbols % and _. The percent character matches any substring. And on the other hand, the underscore character matches any character. The most distinct difference between these two symbols are how many characters they can match. (e.g. **SELECT** * **FROM** Student **WHERE** first_name **LIKE** 'Ha%').

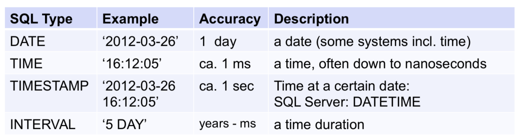
There are some other string operations such as concatenation (using "||"), converting from upper (**UPPER**()) to lower (**LOWER**()) case (and vice versa)

There is regular expression string matching by using a SQL function **regexp_like**(). And there are some metacharacters you can use to construct the expression.



( )	grouping
|	alternative
[ ]	character list
.	matches any character
*	repeat preceeding pattern zero, one, or more times
+	repeat preceeding pattern one or more times
^	start of a line
$	end of line

The date and time in SQL, we can just use the constants provided by the system, **CURRENT_DATE** and **CURRENT_TIME**. And the time and date ops including **EXTRACT**(component **from** date), **DATE** string, +/- interval (e.g. '2012-04-01' + **INTERVAL** '36 HOUR')



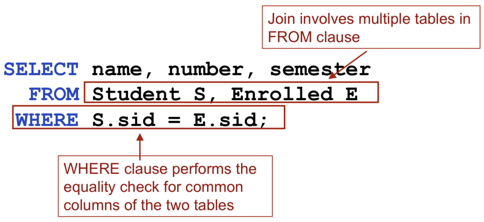
SQL Type	Example	Accuracy	Description
DATE	'2012-03-26'	1 day	a date (some systems incl. time)
TIME	'16:12:05'	ca. 1 ms	a time, often down to nanoseconds
TIMESTAMP	'2012-03-26 16:12:05'	ca. 1 sec	Time at a certain date: SQL Server: DATETIME
INTERVAL	'5 DAY'	years - ms	a time duration

The **FROM** statement lists the tables (relations) where the data comes from. Which is really basic that appear in the previous examples given above.

The aliases are for the case that some relation will be referred twice, then you can just name each relation a preferred name that you can distinguish them in the later use. (e.g. **SELECT** sid **AS** student_ID)

2.1 Join

The **JOIN** keyword will concern several relations and combine them into one, then all the conditions and selected results are from the new-generated relation table. And the common column in those tables is usually the primary key



Join involves multiple tables in FROM clause

```
SELECT name, number, semester
  FROM Student S, Enrolled E
WHERE S.sid = E.sid;
```

WHERE clause performs the equality check for common columns of the two tables

Join type	description	example
inner join	Join based on the common column	the joins in postreSQL are inner join by default R **JOIN** S = R **INNER JOIN** S
outer join	The rows that have nothing in common can be joined and appear in the result table	-
natural join	Tell the system directly use the common column to join those tables (like ... inner join ... on(using) ...)	R **NATURAL JOIN** S
union join	includes all columns from each table in the join, and an instance for each row of each table	-

the more common used operations are:

- R **natural join** S
- R **inner join** S **on** <join condition>
- R **inner join** S **using** (<list of attributes>)