Lesson 11 : SQL

2018-10-22

SEQUEL: Structured English QUEry Language
↳ Based on some algebra, but mostly tuple calculus

General SQL Query syntax:

SELECT column$_1$, column$_2$, ... column$_N$
FROM table$_1$, table$_2$, ... table$_M$
WHERE condition;
↳ Column$_i$ is the name of a column, like BirthYear
↳ Table$_j$ is the name of a table, like RegularUser
↳ Condition may compare values of columns to constants or to each other like:
- BirthYear > 1985
↳ Conditions can be combined

Distinct: Since tables may have duplicate rows, use DISTINCT when you have duplicate rows in a set.

SQL Dot notation: Example would be RegularUser.BirthYear
This is useful if ambiguity is present

Natural inner join => Aliases: Place an alias after the table name to shorten the length of the query:
Example: SELECT Email, R.BirthYear ... FROM RegularUser AS R
↳ Must be used when joining a table to itself

String matching: Useful for finding text matches
Any length: SELECT ... FROM ... WHERE ... LIKE '%value'
Exact length: " " " " 'V____';

Summary: SQL is the lingua franca of database languages. It has a general form, with tables, columns, and conditions. SQL queries can have duplicates (use DISTINCT). Dot notation is useful, and sometimes required, when joining tables

Lesson 11: SQL

2018-10-22

Sorting: Sorting is absolutely needed for output sets
Example: SELECT ... FROM ... ORDER BY <attr> ASC/DESC

Set operation: UNION ⇒ Used for getting single result from multiple tables
=> UNION will return a set; no duplicates!
=> UNION ALL will return duplicates where they occur

Set operation = INTERSECTION ⇒ Used for returning multiple conditions
Example: SELECT ... FROM ... INTERSECT ... SELECT ... FROM ...
↳ No duplicates! To include duplicates, use INTERSECT ALL

Set Operation = EXCEPT ⇒ Opposite of intersection.
Example: SELECT ... FROM ... EXCEPT ... SELECT ... FROM ...
↳ No duplicates! To include, use EXCEPT ALL

Built-in functions: count, sum, avg, min, max

Group By: grouping a return set based on some condition. This is useful when using COUNT so that those columns can be grouped.
Ex: SELECT ... FROM ... GROUP BY <attr> ...

Condition on the GROUP BY: When you want to include conditions on the groups created by GROUP BY.
Ex: SELECT ... FROM ... GROUP BY <attr> ... HAVING <condition>

Summary: SQL can replicate set operations (UNION, INTERSECT, EXCEPT) both with/without duplicates ('ALL'). You can group results by GROUP BY, and place conditions on those groups using HAVING.

Lesson 11: SQL

2018-10-22

Nested queries

Type 1: SELECT... FROM... WHERE [attr] IN (SELECT....)

↳ This is the IN / NOT IN type

Type 2: SELECT... FROM... WHERE ... AND [attr] > ALL (SELECT...)

↳ This is the comparison (=, ≠, ≤, ≥, <, >), SOME/ALL

↳ Basically, all nested results have a condition that they must meet in order to be included.

Nested Queries: =, ≠, ≤, ≥, <, > SOME/ALL

```
SELECT CurrentCity
FROM RegularUser R, YearSalary Y
WHERE R.BirthYear = Y.BirthYear AND  Salary > ALL
(SELECT Salary
FROM RegularUser R, YearSalary Y
WHERE R.BirthYear = Y.BirthYear AND HomeTown = 'Austin');
```

Find CurrentCitys with at least one RegularUser with a Salary that's higher than all Salaries of RegularUsers with HomeTown Austin

YearSalary

Birth Year	Salary
1985	27,000
1969	43,000
1967	45,000
1968	44,000
1988	24,000
1986	26,000
1974	38,000

RESULT

Current City
San Diego
College Park

RegularUser

Email	Birth Year	Sex	Current City	HomeTown
user1@gt.edu	1985	M	Seattle	Atlanta
user2@gt.edu	1969	M	Austin	Austin
user3@gt.edu	1967	M	San Diego	Portland
user6@gt.edu	1988	F	San Diego	Atlanta
user8@gt.edu	1968	M	College Park	Atlanta
user12@gt.edu	1974	F	College Park	Austin

Type 3: Correlated nested queries

Ex: SELECT ... FROM... WHERE NOT EXIST (SELECT...)

↳ Example below cannot have inner query evaluated first

↳ Think of it as a sub-query evaluated once for each row of the outer query

Nested Queries – correlated

Find Email and BirthYear of RegularUsers who have no Interests

```
SELECT R.Email, BirthYear
FROM RegularUser R
WHERE NOT EXIST
    (SELECT *
    FROM  UserInterests  U
    WHERE U.Email = R.Email);
```

think of it as a sub-query evaluated once for each row of the outer query

RegularUser

Email	Birth Year	Sex
user1@gt.edu	1985	M
user2@gt.edu	1969	M
user3@gt.edu	1967	M
user4@gt.edu	1988	M
user6@gt.edu	1988	F
user8@gt.edu	1968	M
user9@gt.edu	1988	F

UserInterests

Email	Interest	SinceAge
user1@gt.edu	Music	10
user2@gt.edu	Blogging	13
user2@gt.edu	Meditation	21
user3@gt.edu	Music	11
user3@gt.edu	Reading	6
user4@gt.edu	DIY	18

RESULT

Email	Birth Year
user6@gt.edu	1988
user8@gt.edu	1968
user9@gt.edu	1988