Week 08

SQL

Johnny Chan

Previously ...

- Basics of SQL
  - CREATE, ALTER and DROP TABLE
  - INSERT, SELECT, UPDATE and DELETE
  - Operator
  - Sorting
  - Single-row function
- SQLite

Agenda

- Multi-row function (aggregate function)
- Join
- Subquery

Multi-row function

- Work on multiple rows to give one result per group
- Also known as group or aggregate function

Multi-row function

Function	Description
COUNT(*) or COUNT(X)	The COUNT(X) function returns a count of the number of times that X is not NULL in a group. The COUNT(*) function (with no arguments) returns the total number of rows in the group
MAX(X)	It returns the maximum value of all values in the group
MIN(X)	It returns the minimum value of all values in the group
AVG(X)	It returns the average value of all values in the group
SUM(X) or TOTAL(X)	It returns sum of all non-NULL values in the group
GROUP_CONCAT(X,Y)	It returns a string which is the concatenation of all non-NULL values of X. If parameter Y is present then it is used as the separator between instances of X

GROUP BY clause

- Often the intention of using multi-row function is to apply them to selected group, where the group is defined by the data
  - e.g. Write a SELECT statement to list all the cities from the Staff table and the number of staff living in each city
- The GROUP BY clause allows a multi-row function to be used on each group specified
- All columns in the SELECT statement that are not associated with the multi-row function MUST be placed in the GROUP BY clause

Example

```
SELECT staffCity city, COUNT(staffCity) '# of staff'
FROM Staff
GROUP BY staffCity;

city         # of staff
-----------  -----------
Auckland     7
Dunedin      1
Hamilton     1
Manukau      1
Nelson       1
Westland     1
```

Another example

```
SELECT roleID, AVG(salary)
FROM StaffAssignment
GROUP BY roleID;

roleID      AVG(salary)
----------  -----------
1           58750.0
2           40750.0
3           47500.0
```

HAVING clause

- The HAVING clause is used to filter the result from a SELECT statement with a multi-row function and a GROUP BY clause

```
SELECT branchNo, AVG(salary)
FROM StaffAssignment
GROUP BY branchNo
HAVING AVG(salary) > 54000;

branchNo    AVG(salary)
----------  -----------
1           55000.0
```

- 🤔 How is HAVING different from WHERE in SQL?

Quiz 01

- Write a SELECT statement to show the number of books transacted per branch

Join

- If the core rationale behind the relational data model and normalisation is to break down a complex data structure into a set of smaller relations, then there must be a mechanism to join them back together to support queries
- What is a join?
  - It is a mean to combine columns from more than one table
  - Most of the time a join condition is specified
  - It is required to prefix the column name with the table name or table alias when the same column name exists in more than one table

Example

```
SELECT bookCode, w.authorNo, a.authorNo, authorFirstName
FROM Writing w, Author a
WHERE w.authorNo = a.authorNo
AND a.authorNo IN (1,2);

bookCode    authorNo    authorNo    authorFirstName
----------  ----------  ----------  ---------------
110         1           1           De Silva
111         1           1           De Silva
112         2           2           Stevens
```

- Instead of using the full table name, it is easier to use a table alias to differentiate among the same named columns from two or more tables

Join type

- Inner join
  - It requires each row in the two joined tables to have matching rows
  - Could use either implicit join notation or explicit join notation
  - Equi-join vs non equi-join
- Outer join
  - The joined table retains each row even if no other matching row exists
  - Left outer join always contains all rows of the left table even if the join condition does not find any matching row in the right table
  - No implicit join notation is allowed
  - Right outer join and full outer join are not supported in SQLite
- Cross join / Cartesian join
  - It joins each row of one table to each row of the other table
- Self join
  - It joins a table to itself

Inner join: equi-join

```
SELECT bookTitle, authorLastName, pubName, pubDate
FROM Author a, Writing w, Book b, Publisher p
WHERE a.authorNo = w.authorNo
AND b.bookCode = w.bookCode
AND w.pubCode = p.pubCode;

bookTitle           authorLastName  pubName         pubDate
------------------  --------------  --------------  ----------
Far from the Crowd  Clarice         Barclay Books   2006-08-31
A Loud Game         Clarice         Bridgeman Pub   2004-07-13
The Artist          Rob             Chuck Sawyer    2000-01-02
Passage to Freedom  Louis           Chuck Sawyer    2003-03-01
Tornado             Clive           McMillan Publ   2007-06-15
Knockdown           Clive           Metcalf Publi   1972-01-23
Judo                Lisa            Hatfield and    1985-02-24
```

- 🤔 Rewrite the SELECT statement with explicit join notation

Inner join: non equi-join

```
SELECT bookCode, price, bookGrade
FROM BookPrice p, BookGrade g
WHERE price BETWEEN minValue AND maxValue;

bookCode    price       bookGrade
----------  ----------  ----------
110         32.5        Low
111         132.5       High
110         82.5        Medium
112         300.0       Very High
113         47.1        Low
114         98.1        Medium
115         23.45       Very Low
114         56.24       Medium
116         84.5        Medium
```

- 🤔 Rewrite the SELECT statement by replacing bookCode with bookTitle

Outer join: left outer join

```
SELECT authorLastName, bookCode
FROM Author a LEFT OUTER JOIN Writing w
ON a.authorNo = w.authorNo;

authorLastName  bookCode
--------------  ----------
Clarice         110
Clarice         111
Rob             112
Louis           113
Clive           114
Clive           115
Theodora
Lisa            116
Gabriella
Beatrice
```

- When there is no matching bookCode for an author, a NULL is given

Cross join

```
SELECT bookCode, a.authorNo
FROM Writing w, Author a;

bookCode    authorNo
----------  ----------
110         1
111         1
112         1
113         1
114         1
115         1
116         1
110         2
111         2
112         2
113         2
```

- 📣 Avoid this in most circumstances!

Self join

```
SELECT c.pubName publisher, p.pubName parent
FROM Publisher c, Publisher p
WHERE c.parentPubCode = p.pubCode;

publisher           parent
------------------  --------------------
Chuck Sawyer Books  Bridgeman Publishers
Lake House Books    Bridgeman Publishers
Barclay Books       Bridgeman Publishers
Metcalf Publishers  Chuck Sawyer Books
Hatfield and Sons   Chuck Sawyer Books
McMillan Publishin  Chuck Sawyer Books
```

- 🤔 Rewrite the SELECT statement so that all publishers are listed on the publisher column

Subquery

- Who has a higher salary than Jones?
  - Main query: staff with a higher salary than Jones
  - Subquery: Jone’s salary
- Similar to nested function, using subquery in SQL is a technique to combine multiple queries into one. The subquery executes before the main query, and the result of the subquery is used to solve the main query
- Enclose subquery in parentheses
- Use single-row operator with single-row subquery; and multi-row operator with multi-row subquery
- For a SELECT statement, subquery can be used within the SELECT, FROM, WHERE and/or HAVING clauses

Subquery with WHERE

```
SELECT staffLastName, salary
FROM Staff s, StaffAssignment sa
WHERE s.staffCode = sa.staffCode
AND salary > (SELECT salary
    FROM Staff s, StaffAssignment sa
    WHERE s.staffCode = sa.staffCode
    AND LOWER(staffLastName) = 'jones');

staffLastName  salary
-------------  ----------
Gupta          72000.0
Marks          64000.0
Spencer        45000.0
McDonald       54000.0
Todd           48000.0
Henderson      40000.0
```

Single-row subquery

- Who gets the highest paid?

```
SELECT staffLastName
FROM Staff s, StaffAssignment sa
WHERE s.staffCode = sa.staffCode
AND salary = (SELECT MAX(salary)
 FROM StaffAssignment);

staffLastName
-------------
Gupta
```

Multi-row subquery

```
SELECT staffLastName FROM Staff
WHERE staffCode IN
 (SELECT staffCode FROM StaffAssignment
    WHERE salary >
     (SELECT MIN(salary) FROM StaffAssignment
      WHERE roleID = 1));

staffLastName
-------------
Gupta
Marks
McDonald
Todd
Pikes
Schindler
```

- 🤔 What is the question of this query?

Subquery with HAVING

```
SELECT branchNo, MIN(salary)
FROM StaffAssignment
GROUP BY branchNo
HAVING MIN(salary) >
 (SELECT MIN(salary) FROM StaffAssignment);

branchNo    MIN(salary)
----------  -----------
1           45000.0
3           40000.0
4           40000.0
```

Quiz 02

- Write a single SQL statement to list all the staff members who have either the same role or same salary as Sean Henderson (staffCode = 7). The query should have 4 columns: staffCode, branchNo, roleID and salary, and it should exclude Sean Henderson from the result

Quiz 03

- Write a single SQL statement to list all the staff members who have been assigned/hired with the three earliest start dates

Subquery with FROM

- Subquery could also be used as a table in the FROM clause, which could participate in join just like any table

```
SELECT MIN(avgSalary)
FROM
 (SELECT branchNo, AVG(salary) avgSalary
  FROM StaffAssignment
    GROUP BY branchNo) t;

MIN(avgSalary)
----------------
43333.3333333333
```

Quiz 04

- Write a single SQL statement to list all the books in stock. There should be two columns: book title and the total in stock (Hint: The total in stock can be calculated by the total received minus the total sold)

Subquery with UPDATE and DELETE

```
UPDATE StaffAssignment
SET salary = (SELECT salary FROM StaffAssignment
              WHERE staffCode = 7)
WHERE salary = (SELECT MIN(salary) FROM StaffAssignment);
```

```
DELETE FROM BookPrice
WHERE bookCode = (SELECT bookCode FROM Book
                  WHERE bookTitle = 'Secrets');
```

Summary

- By now you have learnt:
  - how to use multi-row function
  - how to use join
  - how to use subquery

Reading

- Essential
  - Chapter 3: SQL for SQLite
- Further
  - Aggregate function in SQLite

Schedule

Week	Lecture
01	Introduction
02	Relational model
03	ER modelling
04	Data modelling
05	Data modelling
06	Normalisation
07	SQL
08	SQL
09	SQL
10	DBMS fundamentals
11	Data warehouse
12	Review

THE END

Database is awesome in everywhere!