Data Retrieval

Grigorios Loukides
Email: grigorios.loukides@kcl.ac.uk

Response to interim feedback about the lecturer

- Audible 4.57/5 (if you did not hear something during the lecture ask, seat in the front rows, lecture capture)
- Pace 3/3 - slightly fast (ask me, office hours, lecture capture) or slightly slow (books)
- Explains material clearly 4.43/5 - (ask me, office hours, books)
- Relevance of the topic 4.36/5 - (ask me)
- Useful information on keats 4.18/5 - (in the form of announcements)
- Module objectives clear 4.32/5
- Assessment methods 4.21/5 - (more on the exam)
- Office hours (haven't used, 1 agree)
- Room functions 4.25/5

Response to interim feedback about the lecturer

- Well taught module. Keep it up!
- The professor is perfect !!
- Snippets of code, SQL commands - the next lectures have many. More on books. `https://www.w3schools.com/sql/`
- The contents of the lecture and course are clear and well-taught although a little basic. Given the level of this unit, I think the scope could expand somewhat. - books, material on whiteboard
- Writing on white board - write larger, not near the projector (ok)
- Past exams - `https://internal.kcl.ac.uk/NMS/depts/informatics/stu/pgt/exampapers.aspx`

Session Objectives

In this session, you will learn:

- Sructured Query Language (SQL)
- SELECT query
  - Select
  - Where
  - Sorting
  - Aggregation
  - Grouping
  - Having

Introduction

SQL is a database language that:

- Allows you to create database and table structures, to perform data management tasks and to perform complex queries designed to transform the raw data into useful information.
- It is portable, it is a de facto standard SQL

SQL

SQL functions fit into two broad categories:

- Data Definition Language (DDL): SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects
- It is a data manipulation language (DML): SQL includes commands to insert, update, delete, and retrieve data within the database tables

Data Manipulation Language (DML)

DML allows to retrieve and update data:

- `SELECT` statement retrieves data
- `INSERT`, `UPDATE`, `DELETE` statements update data

Database Example

We are going to illustrate SQL statements using the following database of a letting agency:

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Data Retrieval

- The purpose of the `SELECT` statement is to retrieve and display data from one or more database tables

Select Statement

Select Statement

```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE  condition]
[GROUP BY columnList]  [HAVING condition]
[ORDER BY columnList]
```

- Order of the clauses cannot be changed
- Only `SELECT` and `FROM` are mandatory

Select Statement Execution

`FROM`	Specifies table(s) to be used
`WHERE`	Filters rows
`GROUP BY`	Forms groups of rows with same column value
`HAVING`	Filters groups subject to some condition
`SELECT`	Specifies which columns are to appear in output
`ORDER BY`	Specifies the order of the output

Simple SELECT


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
```

- Can use * as an abbreviation for all columns
- Use DISTINCT to eliminate duplicates (ALL is the default option)
- To name column, use AS clause

Simple SELECT: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

List full details of all staff

```
SELECT *
FROM Staff;
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Simple SELECT: Exercise

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Produce a list of salaries for all staff, showing only the staff number, the first and last names, and the salary details.

Simple SELECT: Exercise

Produce a list of salaries for all staff, showing only the staff number, the first and last names, and the salary details.

```
SELECT staffNo, fName, lName, salary
FROM Staff;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Row selection (WHERE clause)


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
```

[WHERE condition]

Comparison	Compare the value of one expression to the value of another expression
BETWEEN/NOT BETWEEN	Test whether the value of an expression falls within a specified range of values
IN/NOT IN	Test whether the value of an expression equals one of a set of values
LIKE/NOT LIKE	Test whether a string matches a specified pattern
IS NULL/ NOT IS NULL	Test whether a column has a null value

Comparison


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
```

- The following simple comparison operators are available: $=, <>, <, \leq, >, \geq$
- More complex predicates can be generated using the logical operators `AND`, `OR` and `NOT`

Comparison: Example

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

```
SELECT branchNo,street,city,postcode
FROM branch
WHERE city='London' OR city='Glasgow'
```

- What does this query do?

BETWEEN / NOT BETWEEN


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
```

- The BETWEEN / NOT BETWEEN is a simpler way to express a search condition when considering a range of values
- They do not add much expressibility since that can be expressed using two comparison tests

Range: Exercise

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000
```

- Express this select without using a range operator

IN / NOT IN


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
```

- The `IN / NOT IN` tests whether a data value matches one of a list of values
- This list of values can be calculated dynamically with a select (subquery)

Membership: Example

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor')
```

- What does this query do?

LIKE / NOT LIKE


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
```

- LIKE / NOT LIKE test whether a string matches a specified pattern:
  1. The % percent character represents any sequence of zero or more characters
  2. The _ underscore character represents any single character
  3. If you want to search for strings with the symbols % or _ you need to escape them (i.e., using \%or _)

Pattern: Example

OWNERNO	FNAME	LNAME	ADDRESS	TELNO
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

```
SELECT ownerNo, fName, lName, address, telNo
FROM PrivateOwner
WHERE address LIKE '%Glasgow%'
```

- What does this query do?

IS NULL / NOT IS NULL


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
```

- IS NULL / NOT IS NULL test whether a column has a null value

Pattern: Example

Viewing Table

CLIENTNO	PROP	VIEWDATE	COMMNT
CR56	PA14	24-MAY-04	too small
CR76	PG4	20-APR-04	too remote
CR56	PG4	26-MAY-04	
CR62	PA14	14-MAY-04	no dining room
CR56	PG36	28-APR-04	

```
SELECT clientNo, viewDate
FROM Viewing
WHERE propertyNo = 'PG4' AND commnt IS NULL
```

- What does this query do?

Row sorting (ORDER BY clause)


```
SELECT [DISTINCT | ALL]
      {* | [columnExpression [AS newName]] [,...] }
FROM tableName [alias] [, ...]
[WHERE condition]
[ORDER BY columnList]
```

The ORDER BY clause:

- Consists of a list of column identifiers that the result is to be sorted on, separated by commas
- Allows the retrieved rows to be ordered in ascending (ASC) or descending (DESC) order
- ASC is the default ordering option:

  ORDER BY type is the same as ORDER BY type ASC

Sorting: Exercise

The following query

```
SELECT propertyNo, type, rooms, rent
FROM PropertyForRent
ORDER BY type
```

produces:

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

- Write an SQL sentence that arranges these four flats in this list in order of rent

SQL Aggregate Functions

To perform some form of summation or aggregation of data

- There are eight main aggregate functions:
  1. `COUNT` returns number of values in specified column
  2. `COUNT (*)` counts all rows of a table, regardless of nulls or duplicate values
  3. `SUM` returns sum of values in specified column
  4. `AVG` returns average of values in specified column
  5. `STDDEV` returns standard deviation of values in specified column
  6. `VARIANCE` returns variance of values in specified column
  7. `MIN` returns smallest value in specified column
  8. `MAX` returns largest value in specified column
- `COUNT`, `MIN`, and `MAX` apply to numeric and non-numeric fields, but `SUM`, `STDDEV`, `VARIANCE` and `AVG` may be used on numeric fields
- `DISTINCT` can be used to eliminate duplicates

Aggregate: Example

The following query

```
SELECT COUNT(DISTINCT staffNo) AS myCount
FROM Staff
WHERE DOB BETWEEN '01/01/1960'
        AND '31/12/1960'
```

- What does this query do?

Aggregate: Example

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT COUNT (staffNo), SUM(salary)
FROM staff
WHERE position='Manager'
```

- What does this query do?

When Aggregate Functions Can Be Used

- Aggregation functions can only be used in the `SELECT` list and in the `HAVING` clause:
  - If the `SELECT` contains an aggregate function and no `GROUP BY` clause is being used, then no item in the `SELECT` list can include any reference to a column unless that column is in the aggregate function

Aggregation: Exercise I

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

How many properties cost more than 350 per month to rent?

Aggregation: Exercise II

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Find the minimum, maximum, and average staff salary

Grouping

`GROUP BY columnList`

- The `GROUP BY` clause forms groups of rows with same column name (get sub-totals)
- `SELECT` and `GROUP BY` closely integrated: each item in `SELECT` list must be single-valued per group, and `SELECT` clause may only contain:
  - All column names in `SELECT` list must appear in `GROUP BY` clause unless name is used only in an aggregate function
- If `WHERE` is used with `GROUP BY`, `WHERE` is applied first, then groups are formed from remaining rows satisfying predicate

GROUP BY: Example

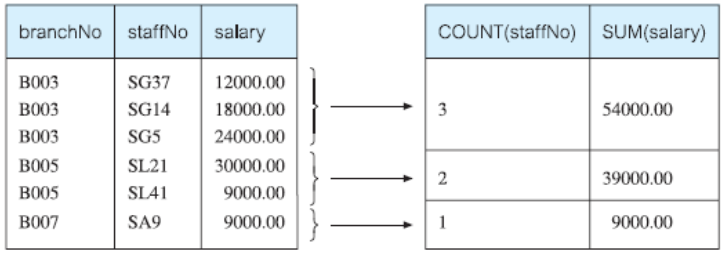
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT branchNo, COUNT(staffNo) AS myCount,
   SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo;
```

- What does this query do?

GROUP BY: Example


```
SELECT branchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo;
```



branchNo	staffNo	salary
B003	SG37	12000.00
B003	SG14	18000.00
B003	SG5	24000.00
B005	SL21	30000.00
B005	SL41	9000.00
B007	SA9	9000.00

COUNT(staffNo)	SUM(salary)
3	54000.00
2	39000.00
1	9000.00

1. SQL divides the staff into groups according to their respective branch numbers
2. For each group, SQL computes the number of staff members and calculates the sum of the values in the salary column to get the total of their salaries
3. Finally, the result is sorted in ascending order of branch number, branchNo

GROUP BY: Example


```
SELECT branchNo, COUNT(staffNo) AS myCount,
    SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Group By: Exercise I

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

List the number of properties managed by each branch

Having


```
GROUP BY          columnList
HAVING condition
```

- HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table
- Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups
- Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function

HAVING: Example

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT branchNo,
COUNT(staffNo) AS count,
SUM(salary) AS sum FROM Staff
GROUP BY branchNo HAVING COUNT(staffNo) > 1
```

- What does this query do?

Data Updates

SQL can be used for modifying the data in the database:

- `INSERT` adds new rows of data to a table
- `UPDATE` modifies existing data in a table
- `DELETE` removes rows of data from a table

Insert Rows

`INSERT INTO` *table_name* [(columnList)] `VALUES` (dataValueList)

- columnList is optional; if omitted, SQL assumes a list of all columns in their original `CREATE TABLE` order
- Any columns omitted must have been declared as `NULL` when table was created, unless `DEFAULT` was specified when creating column
- dataValueList must match columnList

Insert Rows: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName,position, salary,
 branchNo)
VALUES ('SG44', 'Anne', 'Jones','Assistant', 8100, 'B003')
```

or

```
INSERT INTO Staff
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL, NULL,
 8100, 'B003');
```

Copying Rows

`INSERT INTO` *table_name* [(columnList)] selectStatement

- Allows multiple rows to be copied from one or more tables to another

Delete Rows

`DELETE FROM` tableName [`WHERE` searchCondition]

- tableName can be name of a base table or an updatable view.
- searchCondition is optional; if omitted, all rows are deleted from table. This does not delete table. If searchCondition is specified, only those rows that satisfy condition are deleted.

Update Rows: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Delete all properties that relate to branch B003.

```
DELETE FROM PropertyForRent
 WHERE branchNo = 'B003';
```

Update Rows

`UPDATE` tableName
`SET` columnName1 = dataValue1 [, columnName2 = dataValue2...] [`WHERE` searchCondition]

- tableName can be name of a base table or an updatable view.
- `SET` clause specifies names of one or more columns that are to be updated.

Update Rows: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Give all staff a 3% pay increase.

```
UPDATE Staff SET salary = salary*1.03;
```

Update: Exercise I

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Give all Managers a 5% pay increase

Update: Exercise II

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Assume there is a table StaffPropCount that contains names of staff and number of properties they manage:

StaffPropCount(staffNo, fName, lName, propCnt)

Populate StaffPropCount using Staff and PropertyForRent tables

Exercise 1

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Produce a list of monthly salaries for all staff, showing only the staff number, the first and last names, and the salary details.

Exercise 2

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

List the names of all cities where there is a branch

Exercise 3

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

List all properties that are located in 'London' or 'Liverpool' in alphabetical order of street.

Exercise 4

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

How many different members of the staff are managing properties that cost more than 10000 and aren't in London

Exercise 5

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

What is the average salary of managers in each branch?

Exercise 6

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Display the id of all members of the staff that manage more than 10 properties

Exercise 7

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Display the total cost of all flats managed for each member of the staff and each branch

Exercise 8

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Display the id of all owners that have flats managed by different branches

Conclusion

In this session we have covered:

- Introduction to SQL
- Select Statement:
  - Select
  - Where
  - Order by
  - Aggregation
  - Group by
  - Having
- Data Manipulation

Lab Session

Next week's session is about performing simple SELECT queries in MySQL

Request

Complete labs and tutorial feedback on KEATS please!!

Suggested Readings

- Chapter 4 of Fundamentals of Database Systems. Elmasri & Navathe.
- Chapter 5 of Database systems: a practical approach to design, implementation, and management. Connolly, Thomas M; Begg, Carolyn