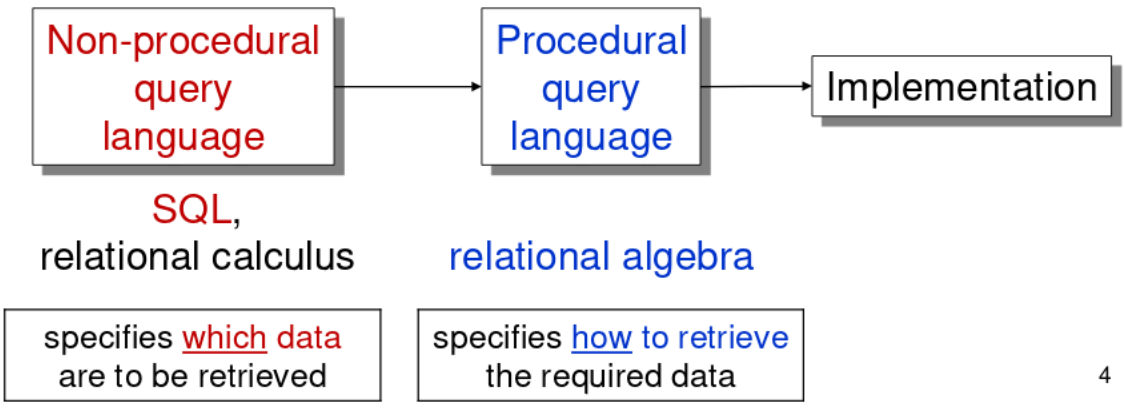
SQL I

1 Database languages

- A good database language should allow users to:
  - Create the database, define relation structures
  - Perform basic data management
  - Perform simple and complex queries
- All these tasks with minimal user effort!
  - Syntax/command structure should be easy to learn
  - Users should concentrate on which queries to make (not how they are implemented)
- The language should be portable:
  - Conform to a recognized standard
  - We can use the same language with many DBMS's
- SQL: (structured query language)
  - The most common database language
  - simple syntax/ easy to learn and use
  - It has two components: DDL & DML
- Data Definition Language (DDL)
  - Allows users to define the database
  - Define the schema for each relation (attributes/types)
  - Define the domain of each attribute
  - Specify integrity constraints
- Data Manipulation Language (DML)
  - Allows users to insert/update/delete/retrieve data from the DB
  - Query Language: the part of the DML that involves data retrieval

2 Two types of query languages

- SQL: formal definition of a new relation from existing relations in the DB
- Relational algebra: specifies how to build a new relation from existing relations in the DB
- Their place in the big picture



3 Writing SQL statements

- SQL statements consist of:
  - Reserved words: a fixed part of SQL
  - User defined words: made up by the user
- SQL Statements
  - Case insensitive (both upper/lower case)
  - Except for literal character data (i.e. data entries)
- SQL is free-format
  - Parts of statements do not have to be written in specific locations on the screen
- However:
  - More readable with systematic indentation and lineation
  - Each clause should begin on a new line
  - Start of a clause should line up with the start of other clauses
  - If a clause has several parts, they should each appear on separate lines and be indented under the start of clause

4 Data Manipulation Language (DML)

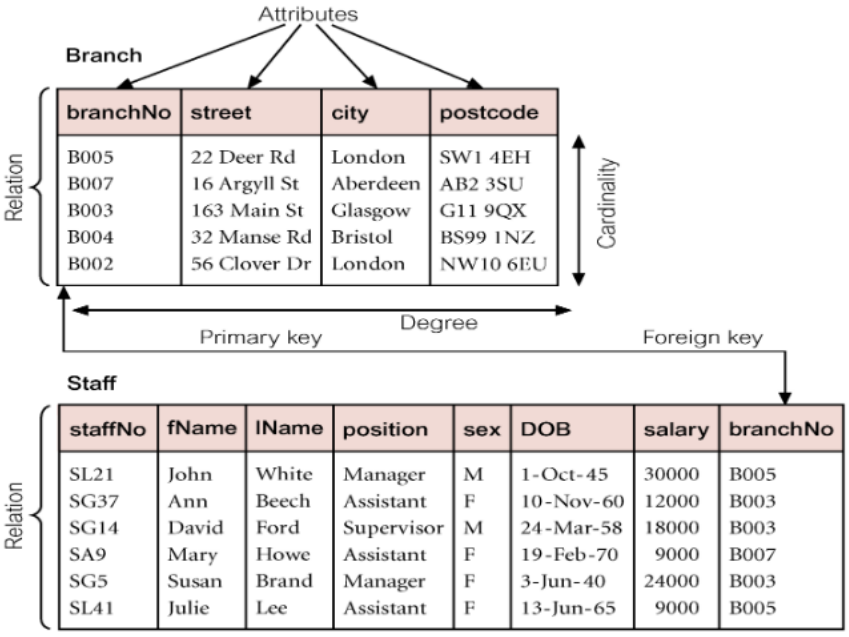
- We mainly look at DML aspects of SQL
- To create a database in MySQL
  - either use DDL command
  - or just use the interactive tools of phpMyAdmin
- Main statements of interest in DML:
  - SELECT - to query data in the database
  - INSERT - To insert new data into an existing table
  - UPDATE - To update data in an existing table
  - DELETE - To delete data from an existing table

5 Writing SQL commands

- Literals (character data/ numericals) are constants used in SQL statements
- All non numeric literals must be enclosed in single quotes
- All numeric literals must not be enclosed in quotes
- Notation:
  - UPPER-case letters represent **reserved** words
  - lower-case letters represent **user-defined** words
  - a vertical bar (—) indicates a choice among **alternatives**
  - curly braces {*a*} indicate a required element
  - square braces [a] indicate an optional element
  - ellipsis (...) indicates optional repetition (0 or more)

6 Examples of syntax

All examples are based on the following tables:



6.1 Simple queries

The sequence of processing in a SELECT-FROM-WHERE statement is:

- SELECT: specifies which columns are to appear in the output
- FROM: specifies the table or tables to be used
- WHERE: filters the rows subject to some condition
  - GROUP BY: forms groups of rows with the same column value
  - HAVING: filters the groups subject to some condition
  - ORDER BY: specifies the order of the output

SELECT and FROM are mandatory

6.2 Syntax

```
SELECT [ALL|DISTINCT] column1[,column2,column3,...]
FROM table1[,table2,table3,...]
[WHERE 'conditions']
[GROUP BY 'column-list']
[HAVING 'conditions']
[ORDER BY 'column-list' [ASC|DESC]]
```

Example:

```
SELECT staffNo, fName, lName, position, sex, DOB, salary, branchNo FROM staff;
```

- The above statement will select the (whole) specified columns from the staff table

Note that if you want to place more queries at once, remember to put a semicolon at the end of each SQL statement. The ; indicates that your SQL statement has finished and the next one can start

7 SELECT

7.1 Example 1

- *List full details of all staff (all columns, all rows)*

```
SELECT staffNo, fName, IName, position, sex, DOB, salary, branchNo FROM Staff
```

- Alternative

```
SELECT * FROM Staff
```

7.2 Example 2

Produce a list of salaries for all staff, showing only: staff number, first name, last name and salary

```
SELECT staffNo, fName, IName, salary FROM Staff
```

This command creates a new table from the table Staff containing the designated columns in the specified order. The rows are NOT ordered

7.3 Example 3

Produce a list of monthly salaries for all staff, showing only staff number, first name, last name and ***monthly salary***

```
SELECT staffNo, fName, IName, salary/12 FROM Staff
```

We can leave the column name blank or use an ”AS” clause

```
SELECT staffNo, fName, IName, salary/12 AS 'Month Salary' FROM Staff
```

8 SELECT & FROM clause review

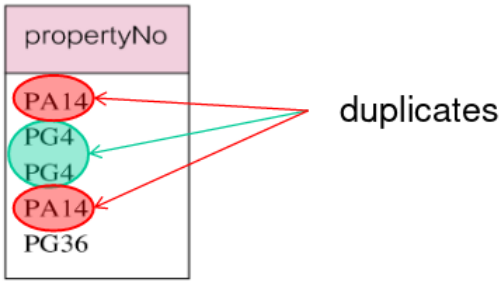
```
SELECT first_column_name, second_column_name
FROM table_name
WHERE first_column_name>12000
```

- Next to the SELECT keyword:
  - the column name(s) specify which will be returned
  - as many columns as we like
  - or * to return all columns
- Next to the FROM keyword:
  - The table name(s) specifies the table that will be required to retrieve the results
- Next to the (optional) WHERE keyword:
  - the condition(s) specifies which rows will be returned (filtering the rows)

9 DISTINCT

- In normalized relational tables there are no repeated rows
- But the use of SELECT may have duplicate rows

```
SELECT propertyNo FROM Viewing
```



- Use DISTINCT to eliminate duplicates

```
SELECT DISTINCT propertyNo FROM Viewing
```

9.1 SELECT Statement, Example 4 (DISTINCT)

List all branch numbers for all branches

```
SELECT branchNo FROM Staff
```

With use of DISTINCT:

```
SELECT DISTINCT branchNo FROM Staff
```

10 SELECT statement (WHERE)

- We often need to restrict the rows that are retrieved
- The WHERE clause is followed by a search conditions (predicates)
  - Comparisons: compare values of two expressions
  - Range
    - BETWEEN/NOT BETWEEN
    - tests whether the values falls within a specified range
  - Set membership
    - IN/NOT IN
  - Pattern matching
    - LIKE/NOT LIKE

11 Range conditions

```
SELECT staffNo, salary
FROM staff
WHERE salary BETWEEN 20000 AND 30000
```

Is there same as

```
SELECT staffNo, salary
FROM staff
WHERE salary >= 20000 AND Salary <= 30000
```

Note that this is inclusive

12 Set membership conditions

List all managers and supervisors

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

This is just to make syntax nicer, it is equivalent to:

```
SELECT staffNo, fName, IName, position
FROM Staff
WHERE position='Manager' OR position='Supervisor'
```

13 Pattern matching (LIKE)

- Sometimes we want to search within a string
- SQL has two special pattern matching symbols
  - % represents an arbitrary sequence of zero or more characters (called wildcard)
  - _ represents an arbitrary single character
- LIKE 'H%' means:
  - first character must be H, but the rest can be anything
- LIKE 'H___' means:
  - exactly 4 characters, first character must he H
- LIKE '%e' means:
  - any sequence of characters, ending at 'e'
- NOT LIKE 'H%' means:
  - The first character can not be 'H'

Find all owners with the string 'Glasgow' in their address

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

14 Combining conditions and Boolean Operations

- The logical AND operator:
  - both sides of the condition must be true
- The logical OR operator:
  - at least one of the two sides must be true
- They can be used in two (or more) conditions in the WHERE clause

```
SELECT fName, IName, position, salary
FROM staff
WHERE position = 'Manager' OR position='Supervisor
```

```
SELECT fName, IName, position, salary
FROM staff
WHERE salary>=24000 AND title='Manager'
```

These two operators can also be used combined

15 ORDER BY clause

- In the resulting table of a SELECT query the rows are NOT ordered
- ORDER BY can be used to sort the rows
  - according to the values of a particular set of columns
  - can be ascending/descending
  - ordering appears regardless of whether that column appears in the result

General format:

```
SELECT column1
FROM 'list-of-tables'
ORDER BY 'column-list' [ASC|DESC]
```

We can also sort according to multiple columns:

- first sort according to the first column
- among rows with the same value in the first column, sort according to the second column etc

16 Aggregate functions

Aggregate functions:

- Operate on a single column
- return a single (numeric) value

numeric data	SUM	returns the sum of the numeric values in a given column
	AVG	returns the average value of a given column
any data	MIN	returns the smallest value in a given column
	MAX	returns the largest value in a given column
	COUNT	returns the total number of values in a given column
	COUNT(*)	returns the number of rows in a table

16.1 Examples

How many properties cost more than £350 to rent

```
SELECT COUNT (DISTINCT propertyNo) AS myCount
FROM PropertyForRent
WHERE rent>350
```

17 GROUP BY clause

- Aggregate functions are similar to the totals at the bottom of a report
- Often we need also ”subtotals” in reports at the bottom of some part of the report
- GROUP BY can be used to:
  - partition the data into groups
  - produce a single summary row (e.g. ”subtotal”) for each group

Find the number of staff working in each branch and sum of their salaries

```
SELECT branchNo,
        COUNT(staffNo) AS myCount
        SUM(Salary) AS mySum
FROM staff
GROUP BY branchNo
ORDER BY branchNo
```