Lecture 1: Relational Databases

CS1106/CS6503– Introduction to Relational Databases

Dr Kieran T. Herley

2019-2010

School of Computer Science & Information Technology
University College Cork

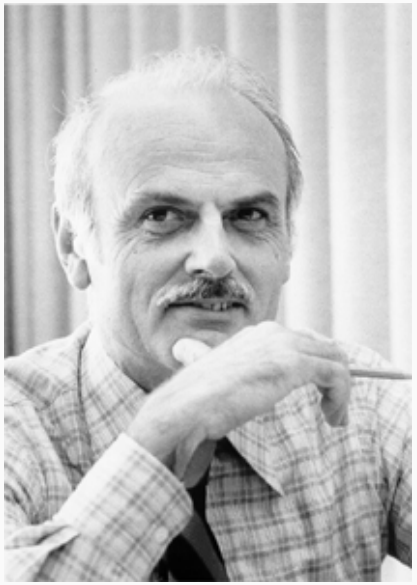
Summary

Relational databases: concepts and terminology. The SQL query language. Syntax of simple project-filter queries.

Relational Database Concept

Relational databases

- Predominant database concept since 1980s
- Conceptually organizes data into sets of two-dimensional tables
- Many relational DB software packages
  - Open Source: MySQL, Postgres
  - Proprietary: Oracle, Microsoft, IBM, Sybase and others



- Edgar "Ted" Codd (1923–2003), English computer scientist
- Introduced relational database concept c1970
- Turing Award 1981

Relations

Organization Conceptually relational databases are based on *tables*

modules

mod_code	*lecturer*	*mod_title*
cs1101	Morrison	Systems Organisation
cs1105	Brown	Foundations of Computer Science
cs1106	Herley	Introduction to Relational Databases
cs1107	Bowen	Intro. to Multimedia Authoring Systems
cs1108	Doherty	Computing: Profession and Society
cs1109	Bridge	Programming and Web Development

Benefits Conceptually intuitive, simple and elegant

Anatomy of a relational database

Database A database consists of a set of named *relations* (aka tables)

Relation

- A set of *tuples* (aka rows) with identical column structure
- Each column named with an *attribute* (must be distinct)
- Values in each column of uniform "type" drawn from some set (*domain*) such as integer, text, date and so on.

Anatomy of a relational database cont'd

students

id_number	*first_name*	*last_name*	*date_of_birth*	*hometown*	*course*	*points*
112345678	Aoife	Ahern	1993-01-25	Cork	ck401	500
112467389	Barry	Barry	1980-06-30	Tralee	ck402	450
112356489	Ciara	Callaghan	1993-03-14	Limerick	ck401	425
112986347	Declan	Duffy	1993-11-03	Cork	ck407	550
112561728	Eimear	Early	1993-07-18	Thurles	ck406	475
112836467	Fionn	Fitzgerald	1994-06-13	Bandon	ck405	485

Some technicalities

- According to relational model, all rows should be distinct; DB systems don't always enforce this however
- Values must be *atomic* ("elementary" and "unstructured")
  - Examples: integers, decimals, text, booleans, dates
  - Can't have lists, sets, records, arrays as single values
  - (Can represent lists etc. by other means–later)
- Every relation has a *key*: some subset of attributes such that no two tuples in the relation can ever have teh same key value.

Schema and instances

students

id_number	*first_name*	*last_name*	*date_of_birth*	*hometown*	*course*	*points*
112345678	Aoife	Ahern	1993-01-25	Cork	ck401	500
112467389	Barry	Barry	1980-06-30	Tralee	ck402	450
112356489	Ciara	Callaghan	1993-03-14	Limerick	ck401	425
112986347	Declan	Duffy	1993-11-03	Cork	ck407	550
112561728	Eimear	Early	1993-07-18	Thurles	ck406	475
112836467	Fionn	Fitzgerald	1994-06-13	Bandon	ck405	485

- Precise contents a database generally vary over time
- We distinguish between

  Database Schema the "structure" of the database in terms of constituent tables and the organization of each table in terms of attributes and domains and the "interpretation" of these in terms of the real-world situation the DB models; remains fixed over time

  Database Instance any specific collection of values populating the DB that conforms to the schema; changes as DB is modified
- When speaking of say "the academic records database" we generally mean the former

Brief digression on adjective "relational"

Binary relations in mathematics

- Let $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$ be the set of pairs of integers; Any subset of $\mathbb{Z}^2$ is a *binary relation*
- For example

$$\mathcal{L} = \{(x, y) : x, y \in \mathbb{Z} \text{ such that } x \leq y\}$$

  is the set of integer pairs where the first is no larger than the second

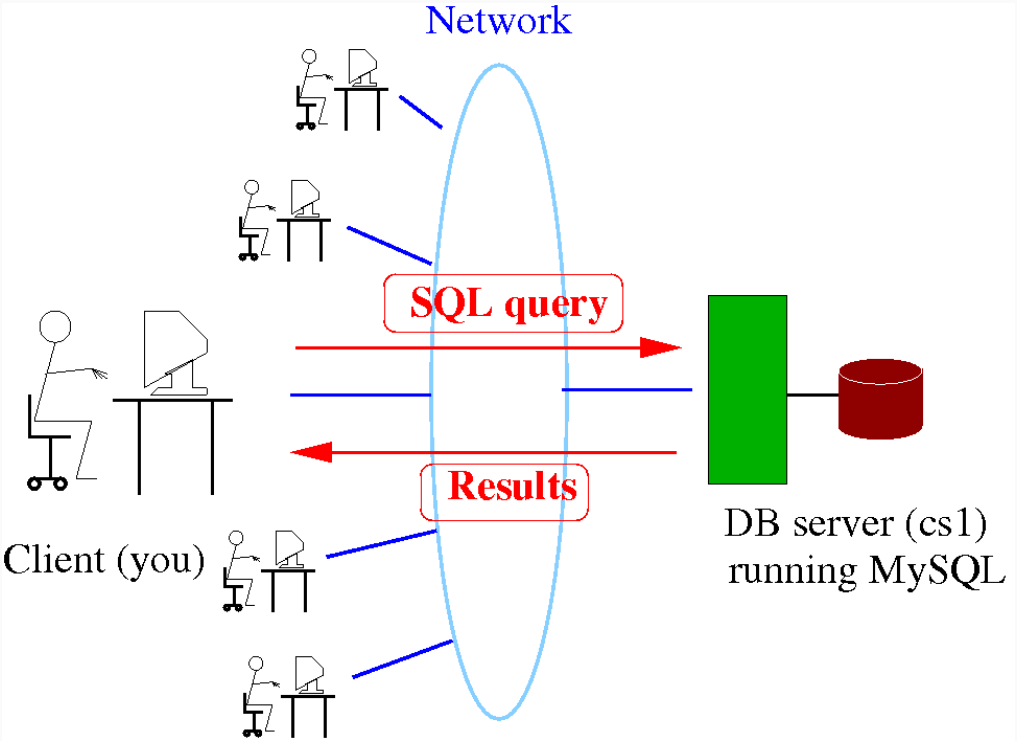
Other relations Can define relations with any number of components; Component values need not be restricted to integers

Relations and DBs

- Can interpret elements of k-ary (i.e. k components) relation as rows of a table
- Mathematical underpinning useful in handling DB manipulations

SQL

SQL



Our "instructions" to DBMS must be expressed in notation called SQL

SQL

What is SQL?

- Computer language for expressing management, manipulation and querying of relational databases
- Pronounced either as "ess-qu-ell" or "sequel"

Support

Good News "Standard" DB language; most common DBMSs support it; free open sources packages available

Bad News Most DBMSs adopt their own "dialects"

- DB software is generally not completely portable between DBMSs

Why computer languages?

Natural languages English can be imprecise and ambiguous, so very difficult to write software to "understand" and carry out our instructions/intentions reliably

Computer languages

- Designed to be clear, concise and unambiguous
- Amenable to automatic interpretation and execution
- Can seem cryptic and error-intolerant
- Examples: SQL, HTML, PHP, Java, . . .

What SQL allows us to xxpress

- SQL incorporates several aspects

  Data Manipulation Language (DML)

  - notation for expressing
  - management of DB
  - manipulations: adding/removing/updating tuples
  - queries to interrogate DB

  Data Definition Language (DDL)

  - notation for expressing DB structure (databases, tables, domains *etc.*) and so on

- We will focus on SQL queries for now

Some SQL Examples

```
CREATE TABLE students
(
    id_number CHAR(9),
    first_name  VARCHAR(20),
    last_name VARCHAR(20),
    date_of_birth  DATE,
    hometown VARCHAR(20),
    course  CHAR(5),
    points  INT,

    PRIMARY KEY (id_number)
);
```

Defines structure of students table

```
SELECT first_name, last_name
FROM students
WHERE points >= 500;
```

Seeks names of 500-point students

SQL Sample 2

```
SELECT first_name, last_name
FROM students
WHERE points >= 500;
```

Meaning "Extract names (first and last) of all students with at least 500 points"

Notes

- SELECT, FROM, WHERE are *keywords*
- Other words are names of table/columns
- >= means $\geq$
- Terminate query with semicolon

SQL Sample 2 cont'd

Table

students

id_number	*first_name*	*last_name*	*date_of_birth*	*hometown*	*course*	*points*
112345678	Aoife	Ahern	1993-01-25	Cork	ck401	500 •
112467389	Barry	Barry	1980-06-30	Tralee	ck402	450
112356489	Ciara	Callaghan	1993-03-14	Limerick	ck401	425
112986347	Declan	Duffy	1993-11-03	Cork	ck407	550•
112561728	Eimear	Early	1993-07-18	Thurles	ck406	475
112836467	Fionn	Fitzgerald	1994-06-13	Bandon	ck405	485

SQL Query

```
SELECT first_name, last_name
FROM students
WHERE points >= 500;
```

Result

first_name	*last_name*
Aoife	Ahern
Declan	Duffy

- Result is a table (anonymous)
- Includes specified columns and rows only
- Original table left completely untouched

A Closer Look At That Query

```
SELECT first_name, last_name      /* First bit */
FROM students                     /* Second bit */
WHERE points >= 500;              /* Third bit */
```

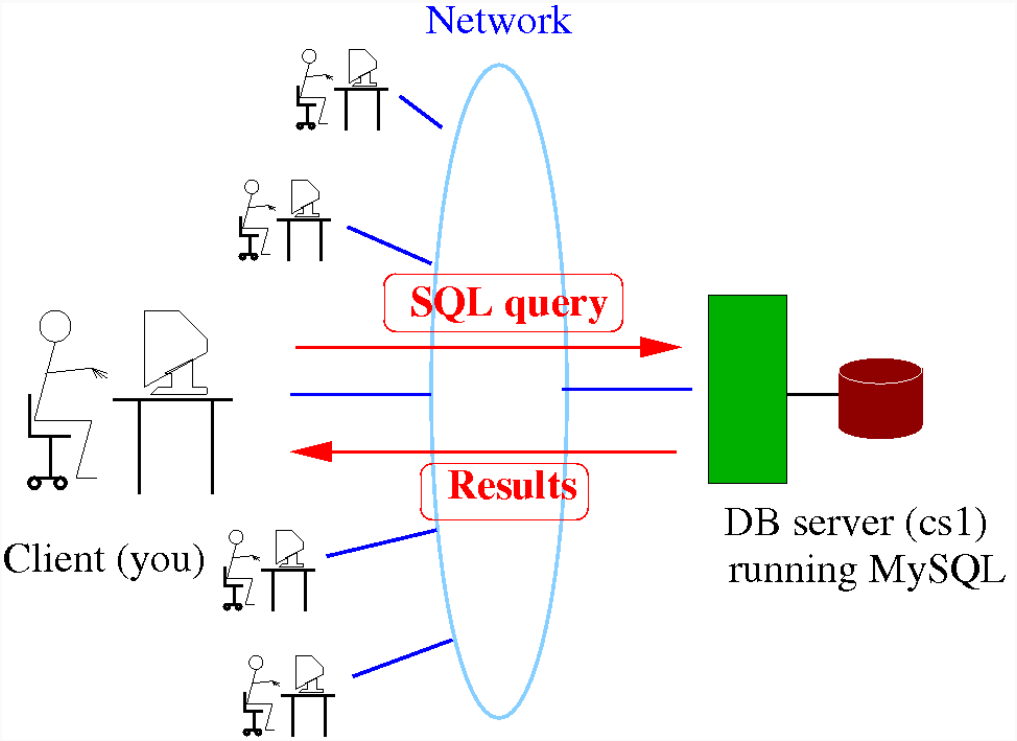
- First bit specifies attributes of interest (note comma separators)
- Second bit specifies table the query applies to
- Third bit specifies *criterion* for rows of interest
  - For each row, condition "points >= 500" is true or false based on the points value in that row
  - Only rows where condition is satisfied appear in result

A Closer Look cont'd

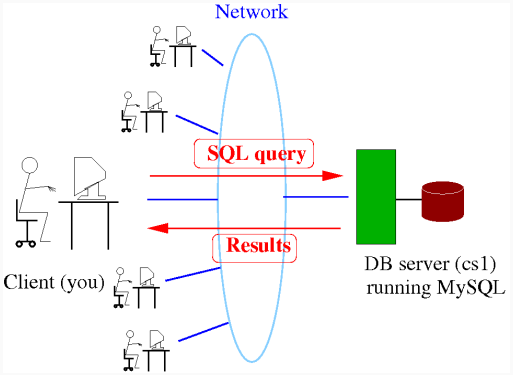
```
SELECT first_name, last_name      /* First bit */
FROM students                     /* Second bit */
WHERE points >= 500;              /* Third bit */
```

- SQL is very picky about dotting of *i*s and crossing of *t*s, so
  - Spell keywords, names correctly ("SELECT" not "SELCT")
  - Take care of ordering of elements within query (don't alter order of bits)
  - Use symbols (commas, semicolons etc.) correctly (don't omit commas etc.)
- (Stuff inside `/* . . . */` are *comments*– annotations intended for human not computer consumption– and have no bearing on meaning or execution of query)

Using SQL


Network
SQL query
Results
Client (you)
DB server (cs1)
running MySQL

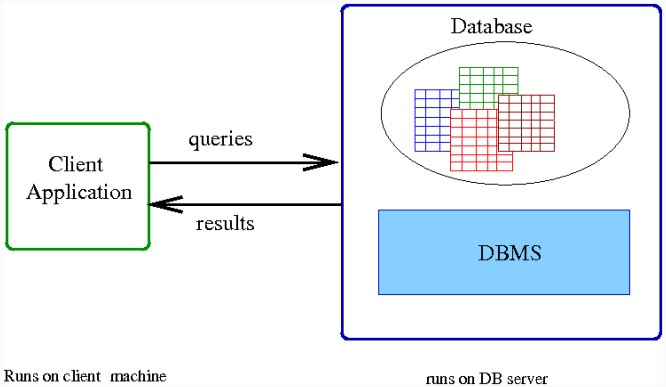
Using SQL cont'd



- Database server: machine that runs DBMS (e.g. MySQL)– provides database "service" to *clients*
- Clients may interact directly with DBMS
  1. Client connects to DBMS (logs in)
  2. Client types SQL query and submits
  3. DBMS executes query and returns results to client
  4. Repeat (2,3) at will; disconnect when done
- (Some computer programs can interact directly to DMBS without human intervention)

Using SQL With Other Programs

- Database application: program that "talks" to DB (via DBMS)



- Example: *student transcript generator*
  - Program generates SQL queries, sends to DBMS
  - DBMS services query, returns results to program
  - Program manipulates results as needed
- Will see more on this in cs1109 later; uses PHP programs to interact with DB

Notes and Acknowledgements