Introductory Databases

R.J.Hendley@cs.bham.ac.uk
Office: 236

Classes

- Lectures:
  - Tuesday 3:00 Arts Main Lecture Theatre
  - Friday 2:00 G15 Muirhead

Assessment

- Assessment:
  - 80% Examination
  - 20% coursework
    - Class test
    - Java Exercise

Provisional Exercise Timetable

Exercise	Title	Date	Weight
1	SQL Class test	Week 3	10%
2	Java & DB	End week 4	10%

Exercise 1 is an unseen class test
Exercise 2 is assessed by viva

All work must be submitted through Canvas

Introductory Databases

Textbooks

- Databases: Ramakrishnan & Gehrke, Data Base Management Systems
- Java & DBs
  - Horstman & Cornell, Core Java, Vol 2
  - Horstman, Big Java

[Note: not *all* of these books!]

DataBase Systems

- In the School:
  - PostgreSql:
    - See: http://supportweb.cs.bham.ac.uk/documentation/postgres
    - A 'proper' DBMS which is freely available
- On your own machines:
  - Postgres can be installed or you can use your favourite system
    - … but be careful with non-standard features!
  - Or you can use ssh or vpn to connect remotely
- Coursework must be demonstrated in the lab.!!!!!
  - …. So it must work with Postgres
  - [Also: note that some implementations include non-standard features – so take care!]

Course Content

An **Introduction** to *design* and *use* of Database systems

- Background, alternatives and justification of DBMS
- Relational Databases:
  - Relational model
  - Introduction to SQL
    - creating & manipulating DBs
    - Introduction to Transactions and concurrency
  - Database Design – ER diagrams and mapping to DB
  - Java & SQL – using a DB through JDBC
- [Focus on *using* DBMS]

Database Management Systems - Background

How can we store (and share) data?

- Store?
- Share?

How can we store (and communicate) data?

Store	Share
In memory: ▪Persistently?	Shared memory
Java objects	Share files locally
Flat/Structured data files: ▪XML ▪Interchange formats	Share files remotely
….	Document Repository
	Data Server – fetch data as required
	……

All of these have their applications:

- Small, volatile datasets
- Documents, data files
- Diaries, address books ….
- Email
- Media libraries
- …..

- Typically:
  - Small
  - Limited shared access
  - Low→medium data security requirements
  - Low→medium consistency
  - Low→medium availability requirements
  - …
- Light weight solutions are fast, adequate, cheap, agile ….

… but what if we need:

- Integration with existing (DBMS-based) systems?
  - Add new functions
  - Add web clients
  - Add new functionality eg eCommerce
  - ….
- Very large amounts of data
- High availability
- High levels of data integrity
- Concurrent access from a large number of users
- High performance

Lecture 2

- What is a DataBase?
- What properties should they have?
- What models are there?
  - Relational Databases
- How does this fit into the bigger picture?
- A *bad* example

So – what is a DBMS?

- Core data base
  - Representation of data
  - Retrieval and maintenance of data
- Collection of tools
  - Design and build and maintenance
  - E.g. standard applications
- Provides:
  - An abstraction from implementation
  - Efficient implementation
  - Integrity, fault tolerance, security …
  - Standard (and bespoke) interfaces
    - For instance SQL for querying DB

DB models

- There are different models available:
  - Network/graph model
  - Hierarchical model
  - **Relational model**
  - … and many more
- **Relational databases** are the pre-eminent choice for *general purpose* data base systems:
  - Especially for commercial/ enterprise systems
    - Consistency, reliability etc. are critical

DB models

- We don't always need all the benefits of a relational database
  - Consistency, integrity, reliability, fault tolerance etc.
- We may not even write to or update our database
- We may need other things:
  - Performance:
    - Fast
    - Low power consumption
  - Scaleability
    - Very large number of users
  - Flexibility
- So other technologies may be better
- Data-warehousing and data-mining
- Geographical systems
- Network infrastructure
- Mobile devices/IoT
- Search engines
- Financial trading
- ……
- But:
  - These often look like proper RDBMS (on the surface)
    - SQL

Databases & Desiderata

- An *organised* collection of data
  - For a purpose
  - To facilitate some set of activities
  - Organised so that it can be accessed and maintained *efficiently*
- Desirable properties
  - Data independence
    - From internal or physical representation
  - Minimise redundancy
    - Store data once
  - Maximise consistency
    - One underlying representation
  - Enable integration and sharing
  - Facilitate change
  - Logical organisation

[of course these apply to most things!]

ACID

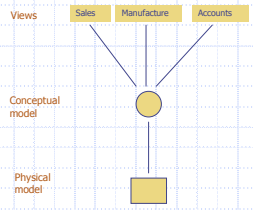
Necessary properties:

- Atomicity – a transaction happens as a whole (or not at all)
- Consistency – a DB is always in a consistent state (according to the rules defined)
- Isolation – the effect of two operations happening in parallel is the same as if they had happened sequentially
- Durability – once something is stored it won't disappear

All this, even if the plug is pulled!

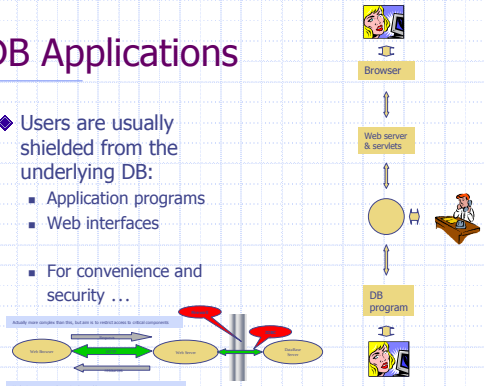
ANSI/SPARC DB Architecture

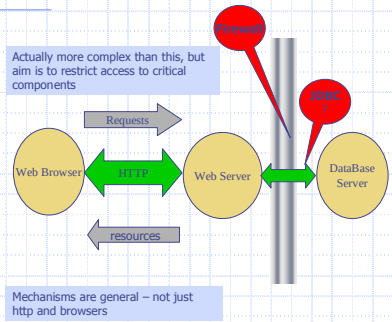
- View or external data model
  - A view onto (a subset of) conceptual model
- Conceptual data model
  - An abstract model independent of implementation
- Physical (internal) data model



DB Applications

- Users are usually shielded from the underlying DB:
  - Application programs
  - Web interfaces
  - For convenience and security …

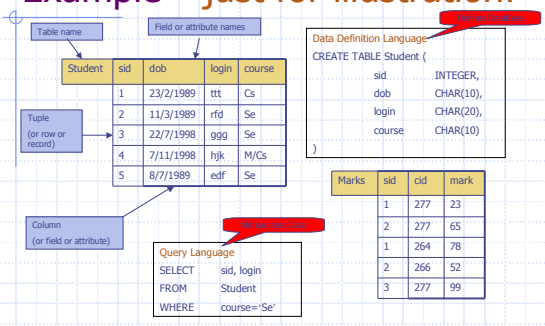


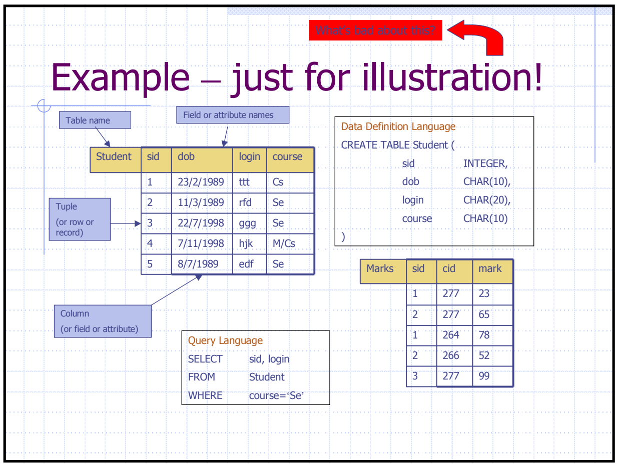


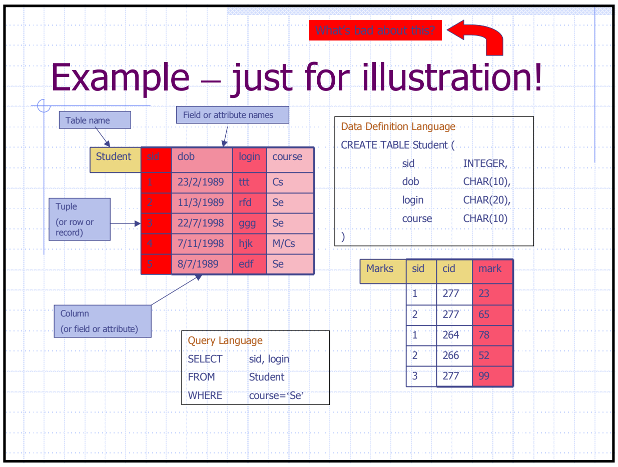
Relational Databases

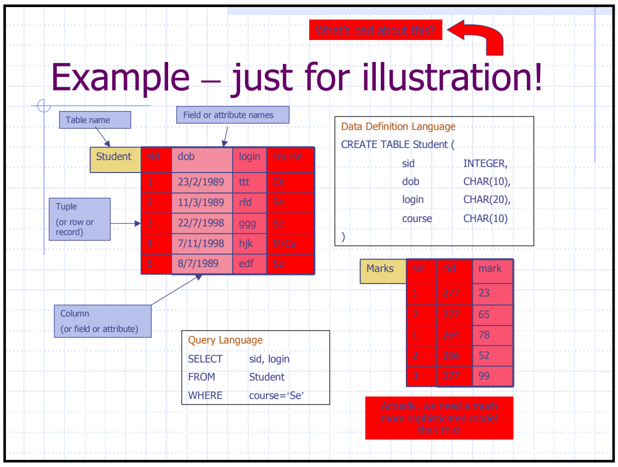
- Main model for data base systems
- Data in form of sets and relations
- Data connected using basic set theory
  - Selecting, combining etc.
- Normally *viewed* as **tables**
- Queries *specify* result **not** how it is computed
  - *declarative*

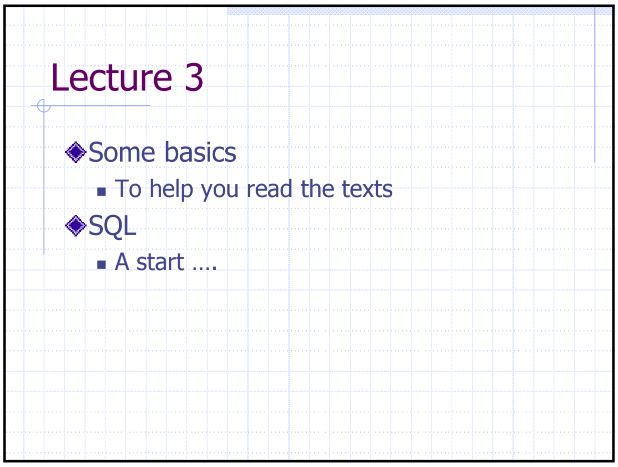
Example – just for illustration!




What's bad about this?
Example – just for illustration!
Table name
Field or attribute names
Student | sid | dob | login | course
1 | 23/2/1989 | ttt | Cs
2 | 11/3/1989 | rfd | Se
3 | 22/7/1998 | ggg | Se
4 | 7/11/1998 | hjk | M/Cs
5 | 8/7/1989 | edf | Se
Tuple (or row or record)
Column (or field or attribute)
Data Definition Language
CREATE TABLE Student (
sid INTEGER,
dob CHAR(10),
login CHAR(20),
course CHAR(10)
)
Marks | sid | cid | mark
1 | 277 | 23
2 | 277 | 65
1 | 264 | 78
2 | 266 | 52
3 | 277 | 99
Query Language
SELECT sid, login
FROM Student
WHERE course='Se'


What's bad about this?
Example – just for illustration!
Table name
Field or attribute names
Student | sid | dob | login | course
1 | 23/2/1989 | ttt | Cs
2 | 11/3/1989 | rfd | Se
3 | 22/7/1998 | ggg | Se
4 | 7/11/1998 | hjk | M/Cs
5 | 8/7/1989 | edf | Se
Tuple (or row or record)
Column (or field or attribute)
Data Definition Language
CREATE TABLE Student (
sid INTEGER,
dob CHAR(10),
login CHAR(20),
course CHAR(10)
)
Marks | sid | cid | mark
1 | 277 | 23
2 | 277 | 65
1 | 264 | 78
2 | 266 | 52
3 | 277 | 99
Query Language
SELECT sid, login
FROM Student
WHERE course='Se'


What's bad about this?
Example – just for illustration!
Table name
Field or attribute names
Student | sid | dob | login | course
1 | 23/2/1989 | ttt | Cs
2 | 11/3/1989 | rfd | Se
3 | 22/7/1998 | ggg | Se
4 | 7/11/1998 | hjk | M/Cs
5 | 8/7/1989 | edf | Se
Tuple (or row or record)
Column (or field or attribute)
Data Definition Language
CREATE TABLE Student (
sid INTEGER,
dob CHAR(10),
login CHAR(20),
course CHAR(10)
)
Marks | sid | cid | mark
1 | 277 | 23
2 | 277 | 65
1 | 264 | 78
2 | 266 | 52
3 | 277 | 99
Query Language
SELECT sid, login
FROM Student
WHERE course='Se'
Actually, we need a much more sophisticated model than this!


Lecture 3
Some basics
To help you read the texts
SQL
A start ….