Lecture 1:
Introduction and Overview

G51DBS Database Systems

Jason Atkin

jaa@cs.nott.ac.uk

This Lecture

- Module Material
- Credits to previous lecturers
- Why this module should be important for you
- Reference books
- Lectures and assessment
- What is a database?
- Relational databases introduction/overview
- Module overview

Module Material

- You can find all of the information on the moodle page
  - moodle.nottingham.ac.uk
  - G51DBS
- Coursework will need to be submitted through the Computer Science moodle page
  - The University moodle has some features missing which we need

Credits

This module has developed over quite a few years, getting many changes and updates each year

I would like to acknowledge and thank:

Michael Pound

for letting me use and modify (significantly) his material from 2010/2011

Peter Blanchfield and **Tim Brailsford**

for their slides from previous years (which Michael used)

Steve Mills and Michael Hartley

who wrote the slides which Peter and Tim used and changed

Books – check the library

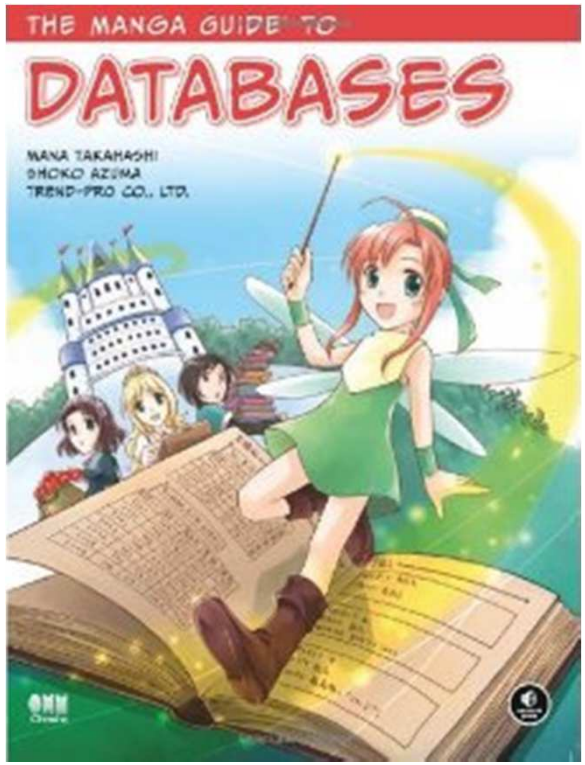
- **Database Systems - A Practical Approach to Design, Implementation, and Management, Connolly & Begg**
  - **Source of many of the diagrams**
  - **Provides much more information than you will need**
  - **A number of copies in the library**
- **Database Systems, Date**
  - **A classic book on databases**
- **Database Systems, Beynon-Davies**
  - **Some copies in the library**
  - **Very short chapters**
  - **I prefer Connolly and Begg, but this has a lot of similar information**
- **Database Principles and Design, Colin Ritchie**
  - **A reasonable overview, not in the same depth as Connolly and Begg, which may suit some people (less to read)**

Books : modern course companions

- **Database systems, Design, Implementation and Management, Rob, Coronel, Crockett**
  - **A modern book designed as a course text**
  - **Seems to cover the major topics**
- **Database Principles: Fundamentals of design, implementation and management, Coronel, Morris and Rob**
  - **Came out in 2012, giving a modern view with online resources**
  - **A nicely written book in my opinion, in depth with many examples**
  - **Written from the point of view of a course text**
  - **Possibly a little 'quick' if you are not confident in your abilities**
  - **Different order to this module (e.g. ER diagrams in chapter 7)**
- **Plus many others, e.g.**
  - **Fundamentals of Database Systems, Elmasri & Navathe**

Books : Manga Guide to Databases

- Simple introduction to some key principles:
  - The Manga Guide To Databases, Mana Takahashi and Shoko Azuma
  - Very accessible
  - Relatively inexpensive
  - Only covers the basic topics, not the later advanced topics
  - Source of some illustrations in this lecture



Lectures and Assessment

- Lectures
  - Tuesday 11am, LT3 Exchange building
  - Tuesday 12 noon, LT3 Exchange building
- Lab sessions (**from 2nd week, 6th February**)
  - Lab session: Wednesday 11am, A32 lab
  - **Important for the coursework**!
  - Please use these to get feedback and to ask questions
  - You need to put in other time yourself as well!
- Assessment
  - One Exam, worth 60%
  - One Coursework on database creation, worth 20%
  - One Coursework on SQL and PHP, worth 20%

An understanding between us

- I will (or have):
  - Prepare the lectures which will teach you key concepts
  - Put (preliminary) lecture slides on the web page in advance
  - Present the lectures
  - Provide lab exercises which will help you to understand
  - Give informal feedback and help in the labs
  - Arrange for lab helpers to help me give feedback and help
  - Write a coursework to test your understanding/practise things
  - Give feedback/marks for the coursework
  - Write the exam
- **You should:**
  - **Attend lectures and take in the information**
  - **Attend the labs and ask questions where appropriate**
  - **Do the lab exercises**
  - **Submit the coursework**
  - **Sit the exam**

Why Study Databases?

- Databases are important for computing
  - Many computing applications deal with large amounts of information
  - Database systems give a set of tools for storing, searching and managing this information
- Databases are a ‘core topic’ in both Computer Science and IT
- Basic concepts and skills with database systems are part of the skill set you will be ***assumed*** to have as a CS or IT graduate
- **Concepts will be of use in other modules**

Databases are (virtually) everywhere!

- Library catalogues
- Medical records
- Bank accounts
- Stock market data
- Personnel systems
- Product catalogues
- Telephone directories
- Train timetables
- Airline bookings
- Credit card details
- Student records
- Customer histories
- Stock market prices
- Discussion boards

and many more...

Do only I.T. people need to understand?

- Not just for I.T. and comp. sci. people!
  - Useful for many management and admin jobs
- "Understanding **relational** databases is fundamental"
  - Director of Student Recruitment and Admissions at a UK University, August 2010
- Employers will expect you to understand more than the managers and directors do

Why I wanted to teach this…

- **Relational databases** are very common
- The **principles** are REALLY important
  - Principles can even help with OO understanding
  - e.g. Shlaer-Mellor OOA, “modelling the world in data”, “modelling the world in states”, etc
  - Translational program development, see Wikipedia entry on Shlaer-Mellor
- Most languages/operating systems provide an SQL method for accessing relational databases
  - ODBC (Mac, Linux, Windows), JDBC (Java), …
  - SQLite is very common (De Re Basic, Android SDK)
- **Allows you to create data-driven applications**

Example: SQLite

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. **The source code for SQLite is in the public domain.**

- **Features Of SQLite**
- **Transactions** are atomic, consistent, isolated, and durable (**ACID**) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Implements most of SQL92. (Some features not supported)
- A complete database is stored in a single cross-platform disk file.
- Supports terabyte-sized databases and gigabyte-sized strings and blobs.
- …
- **Faster than popular client/server database engines for most common operations**.
- Simple, easy to use API.
- …
- Cross-platform: Unix (Linux and Mac OS X), OS/2, and Windows (Win32 and WinCE) are supported out of the box. Easy to port to other systems.
- …

(Source: http://www.sqlite.org)

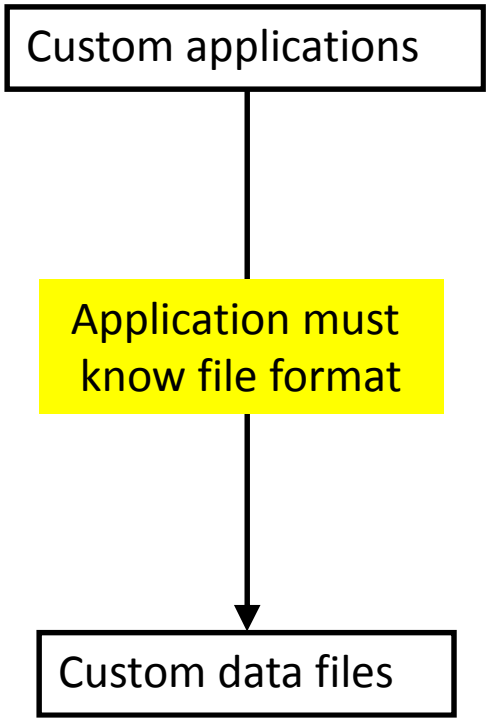
What is a database?

What is a Database?

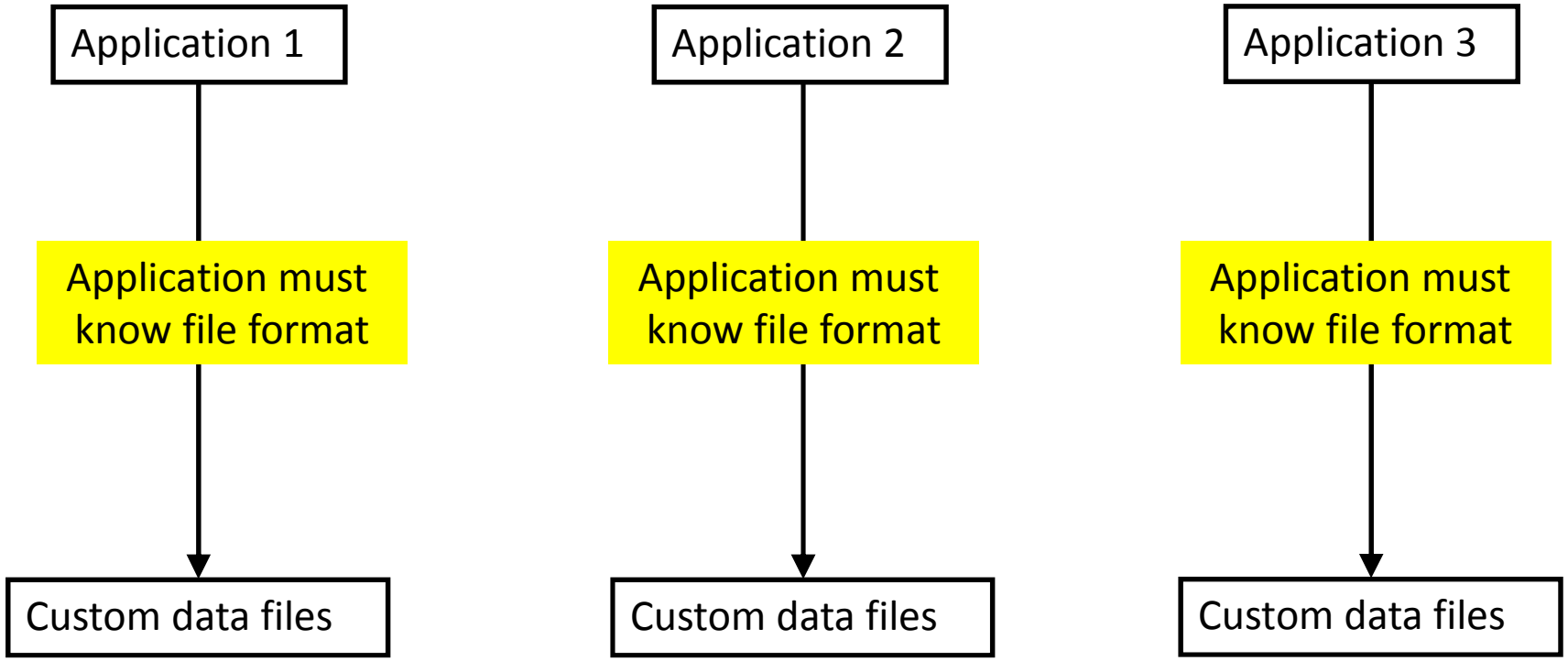
- "A set of information held in a computer"
  - Oxford English Dictionary
- "One or more large structured sets of persistent data, usually associated with software to update and query the data"
  - Free On-Line Dictionary of Computing
- "A collection of data arranged for ease and speed of search and retrieval by a computer. "
  - American Heritage Science Dictionary

The early days…

- Applications store (& persist) their data in files
- Each file has its own format
- Program has to know format
- Any other program using file has to know format



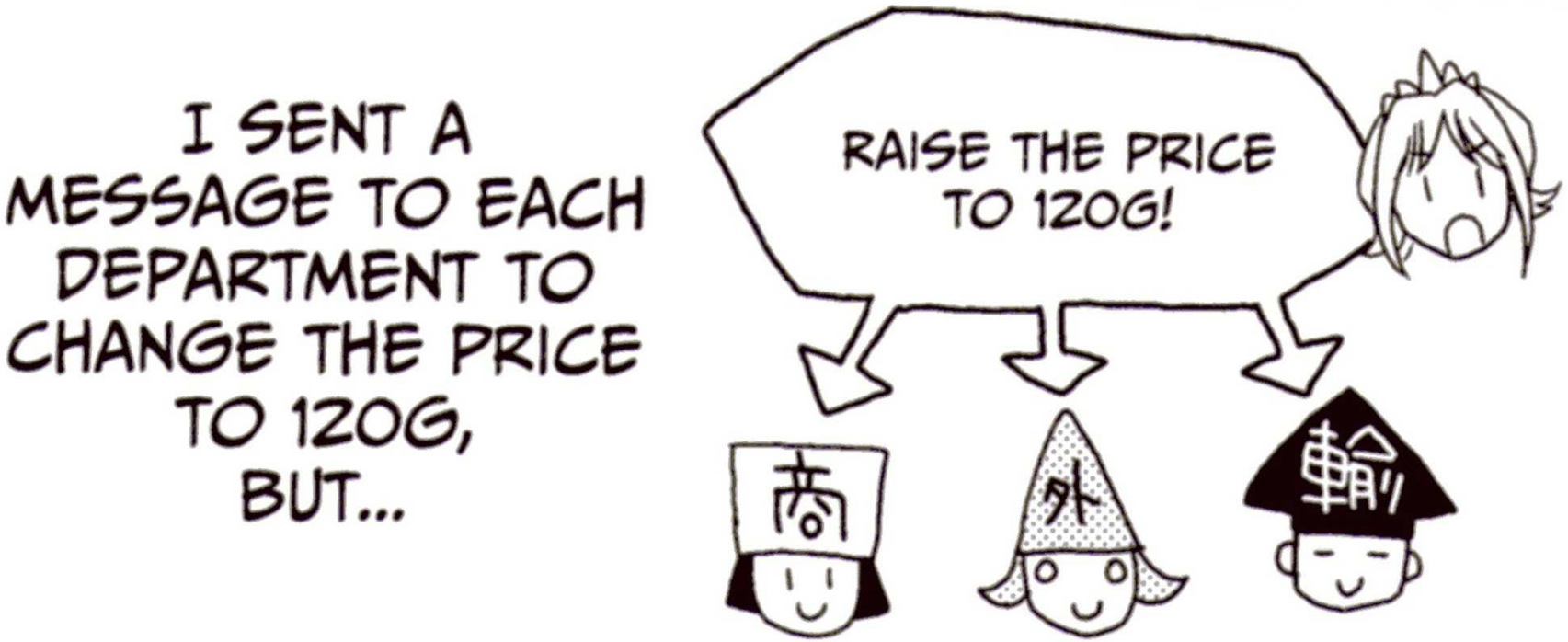
Multiple applications



An Example

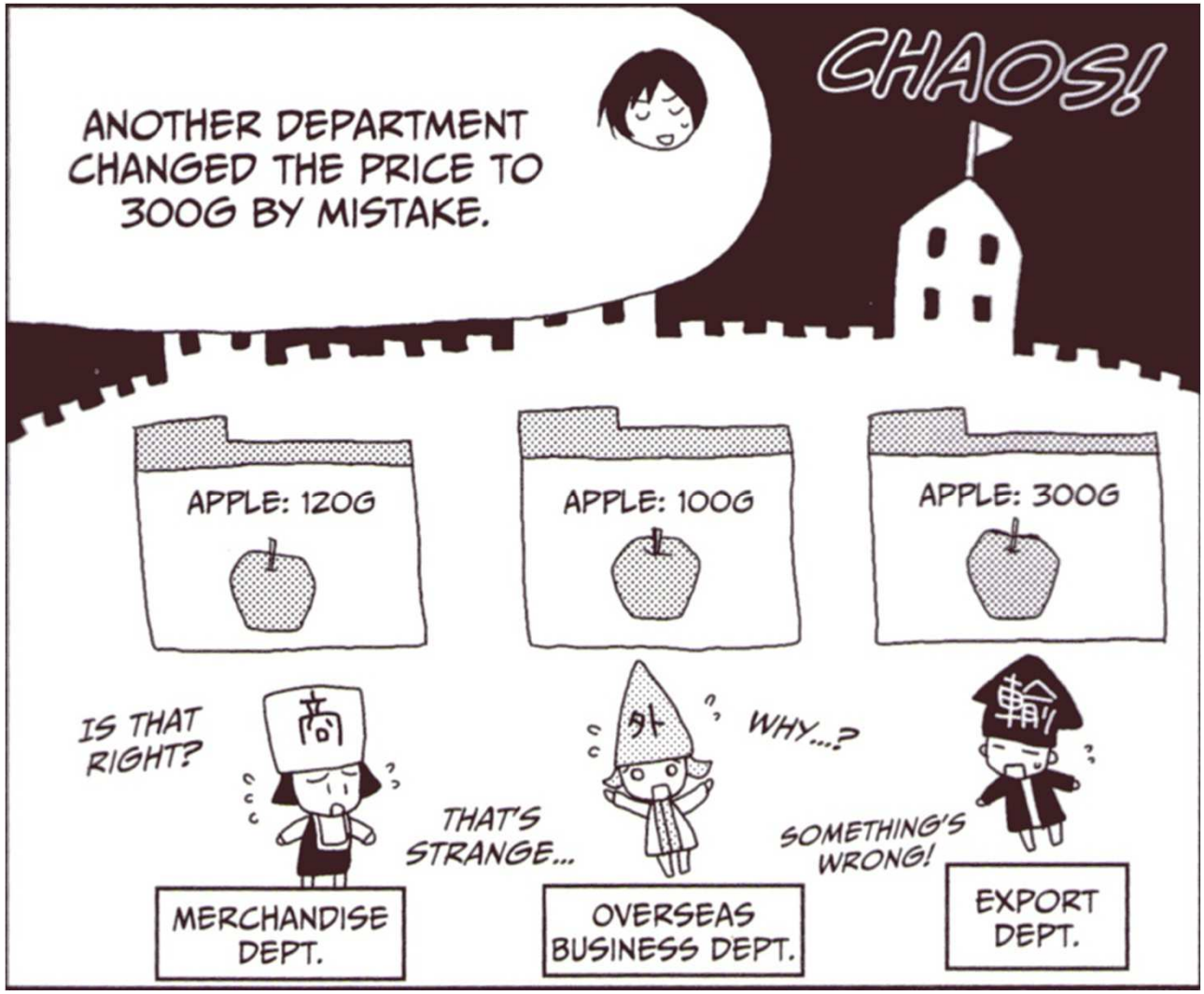
- From the Manga Guide to Databases:
- The Kingdom of Kod exports apples
- Price is 100G per container of apples
- Departments:
  - Merchandise
  - Overseas Business
  - Export

Can lead to errors…



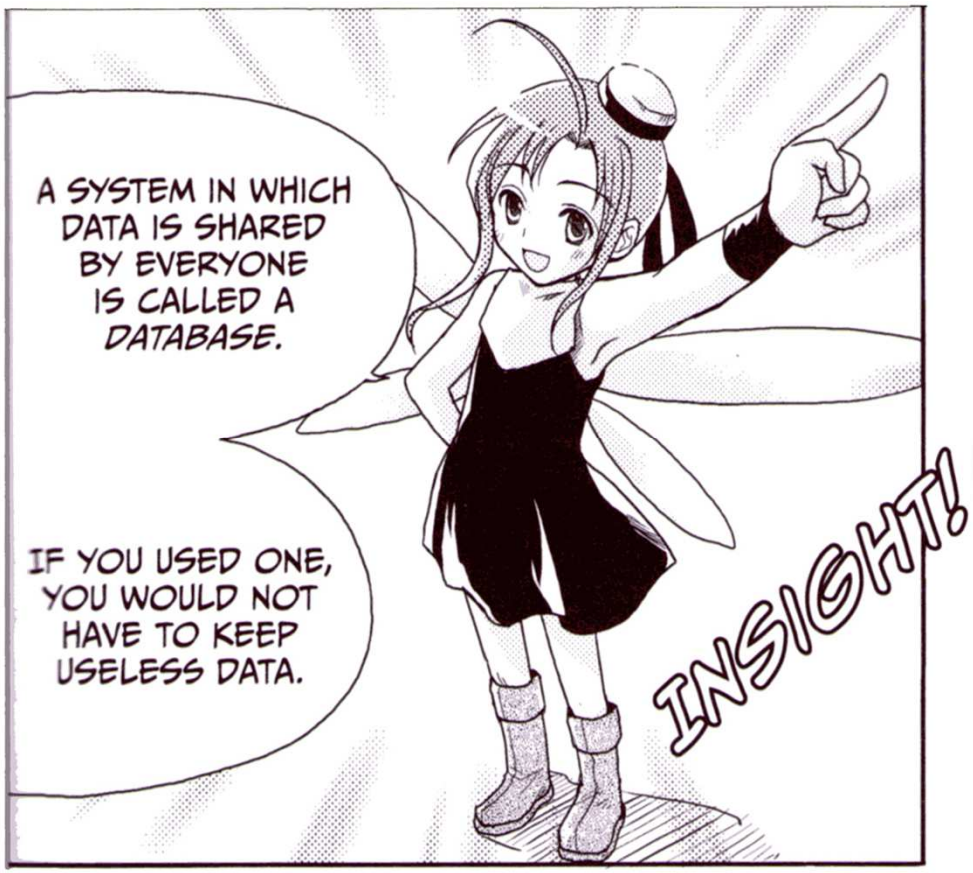
Mistakes happen…

… overseas business department didn't get the message and ...

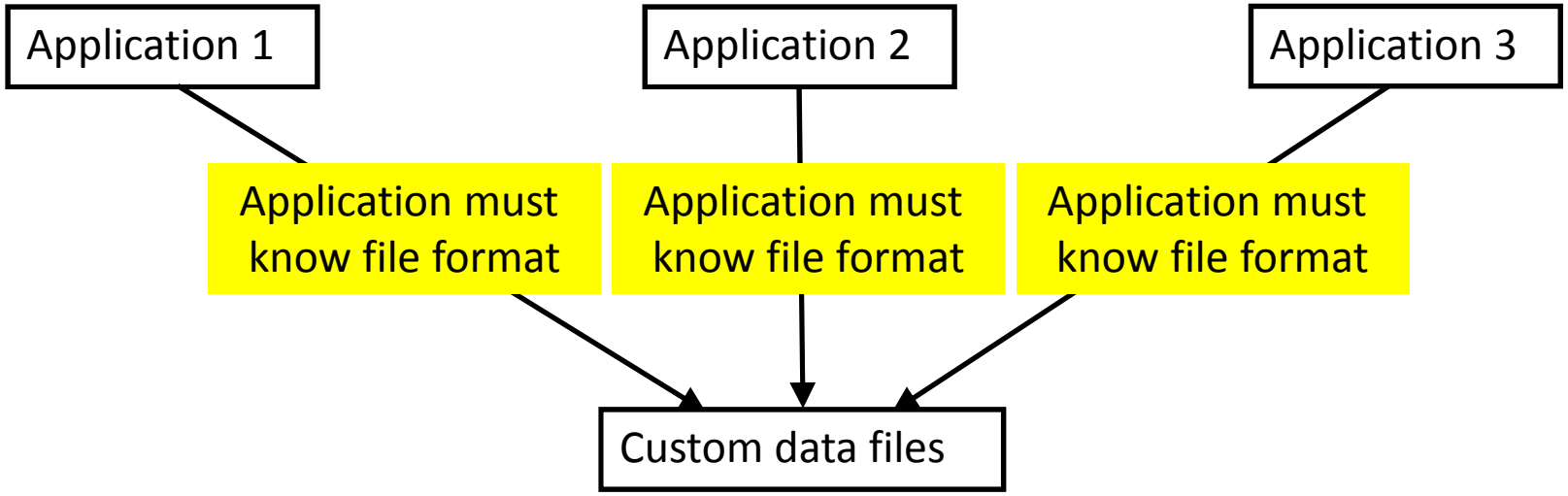


Redundant Data

Storing the same data several times in different places (redundancy) is error-prone!

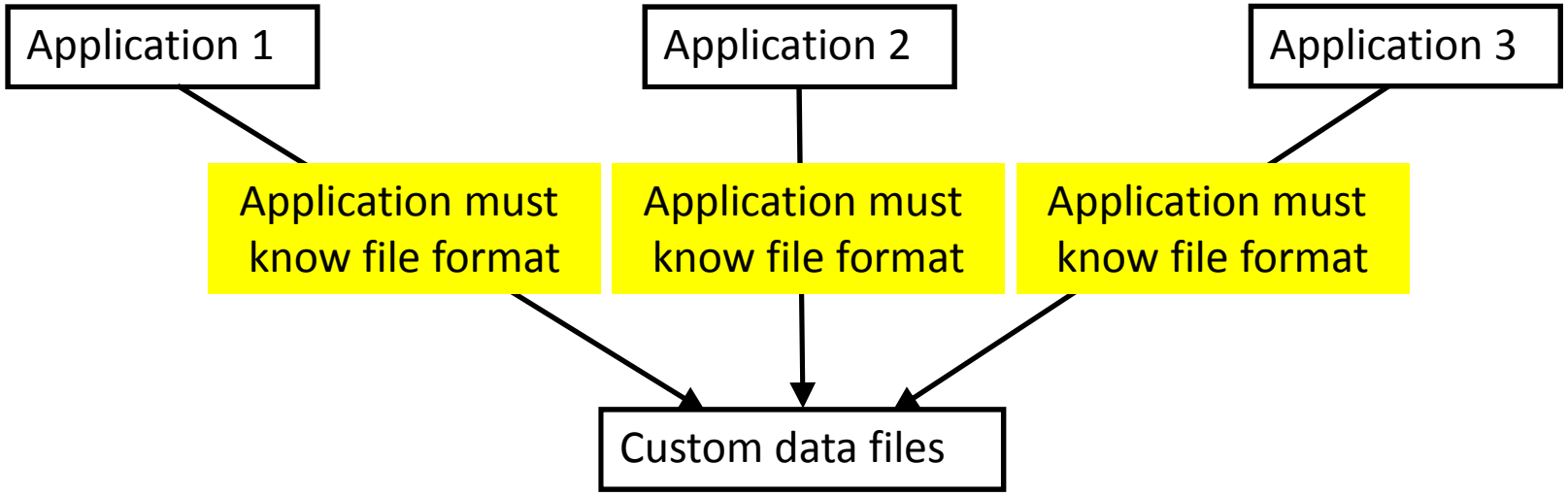


Multiple applications



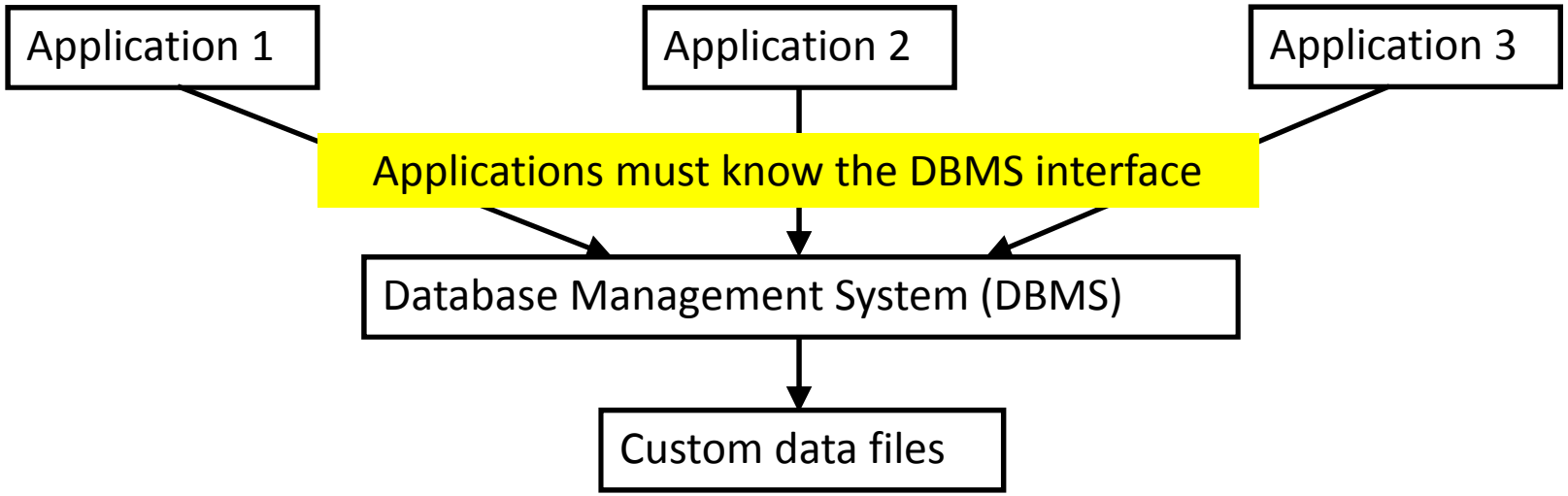
- So, keep one copy of data…
  - All applications must know the file format
  - Data for all applications must be present
    - Is any duplicated?
  - If one changes something, do other things change?
    - Must know the data for the other applications?
  - What if file or data format changes?

Still problems



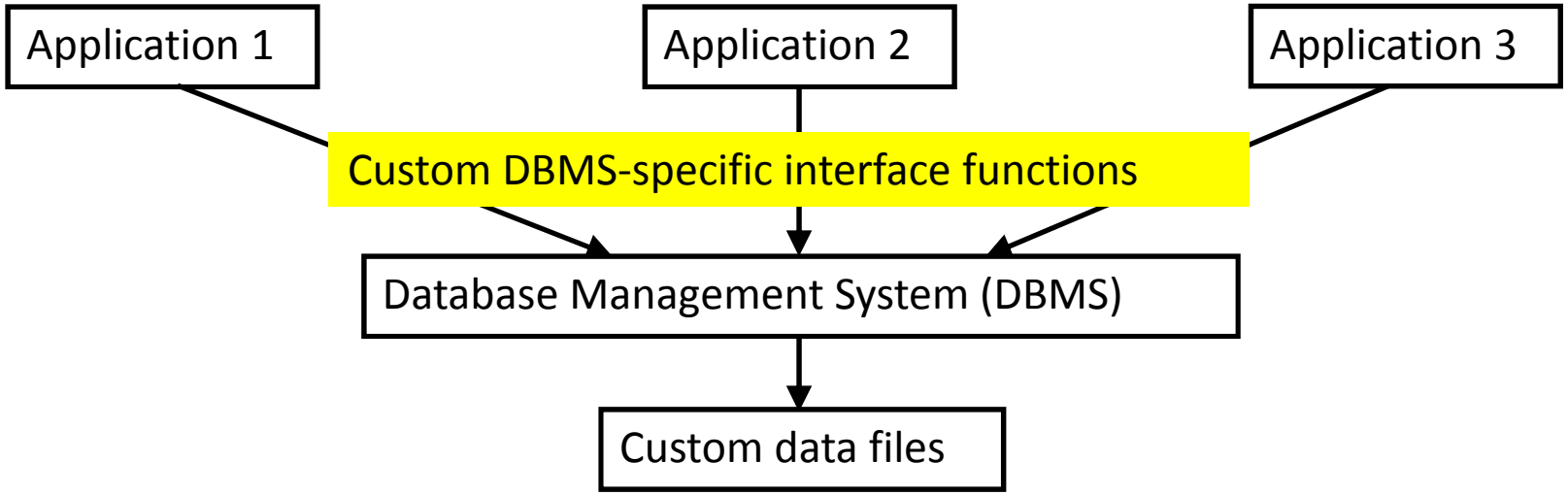
- Still problems, for example:
  - Concurrency (multiple simultaneous changes?)
  - Security (everyone can see everything?)
  - Integrity (validation/correctness)

Put something in the middle…



- A program in the middle can coordinate access
  - Preventing simultaneous access problems
  - Could provide extra integrity and security checks
- Applications link with DBMS rather than data files

Early databases



- Early databases were organised by the developer
  - New functions specifically created, not reusable
  - Adding new queries was complicated
  - No standards – database specific
  - Data duplication and data dependencies
  - Did not aid security, recovery, concurrency etc

Relational databases

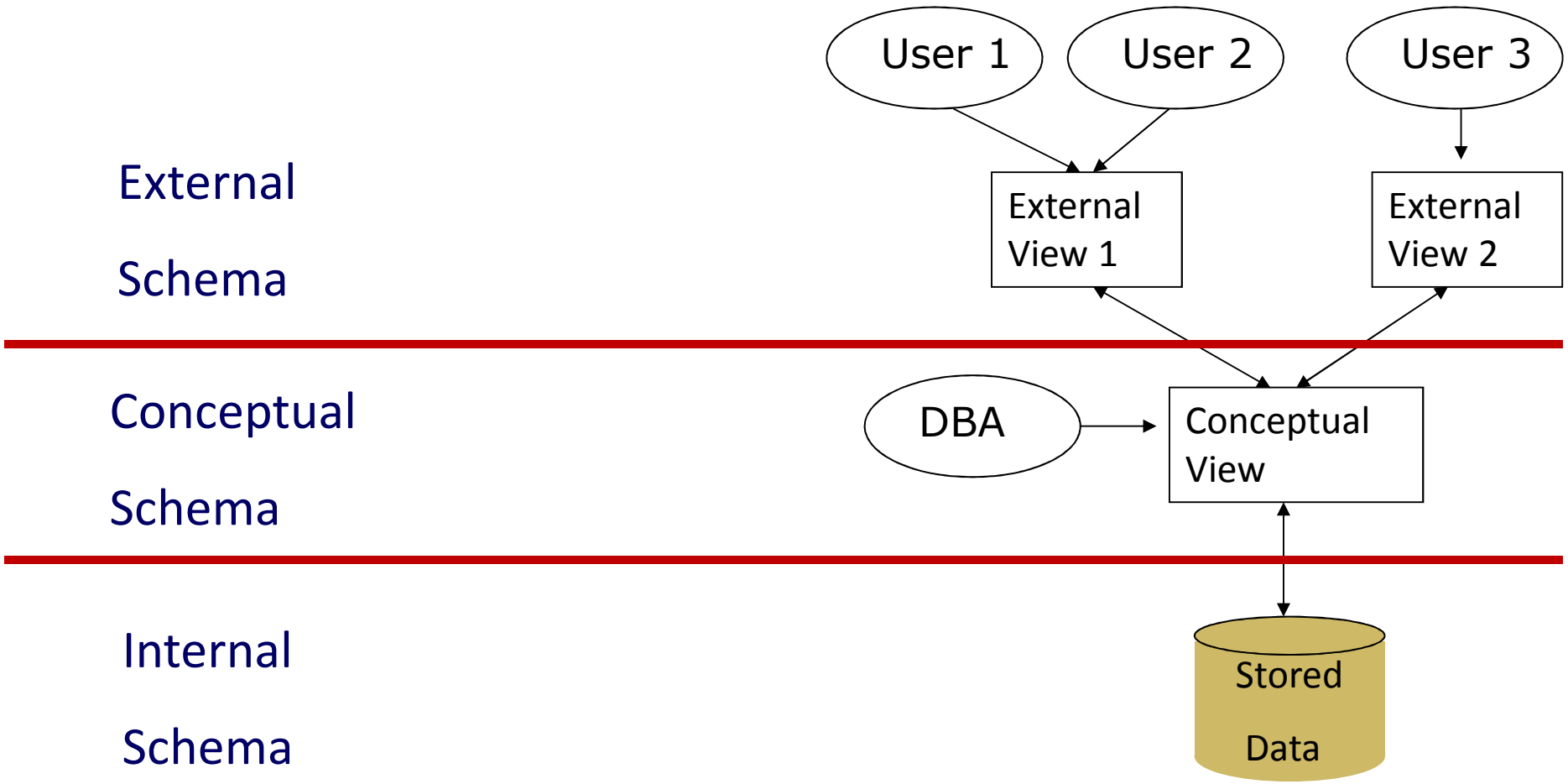
Relational Databases

- 1970: E. F. Codd introduced the relational model
  - “A relational Model for Large Shared Databanks”
- Information stored as records in relations (tables)
  - Sound mathematical basis
- Model covers data:
  - Structure
  - Integrity
  - Manipulation
- Other types of database exist, many are old (obsolete?), others are special purpose

ANSI / SPARC Architecture

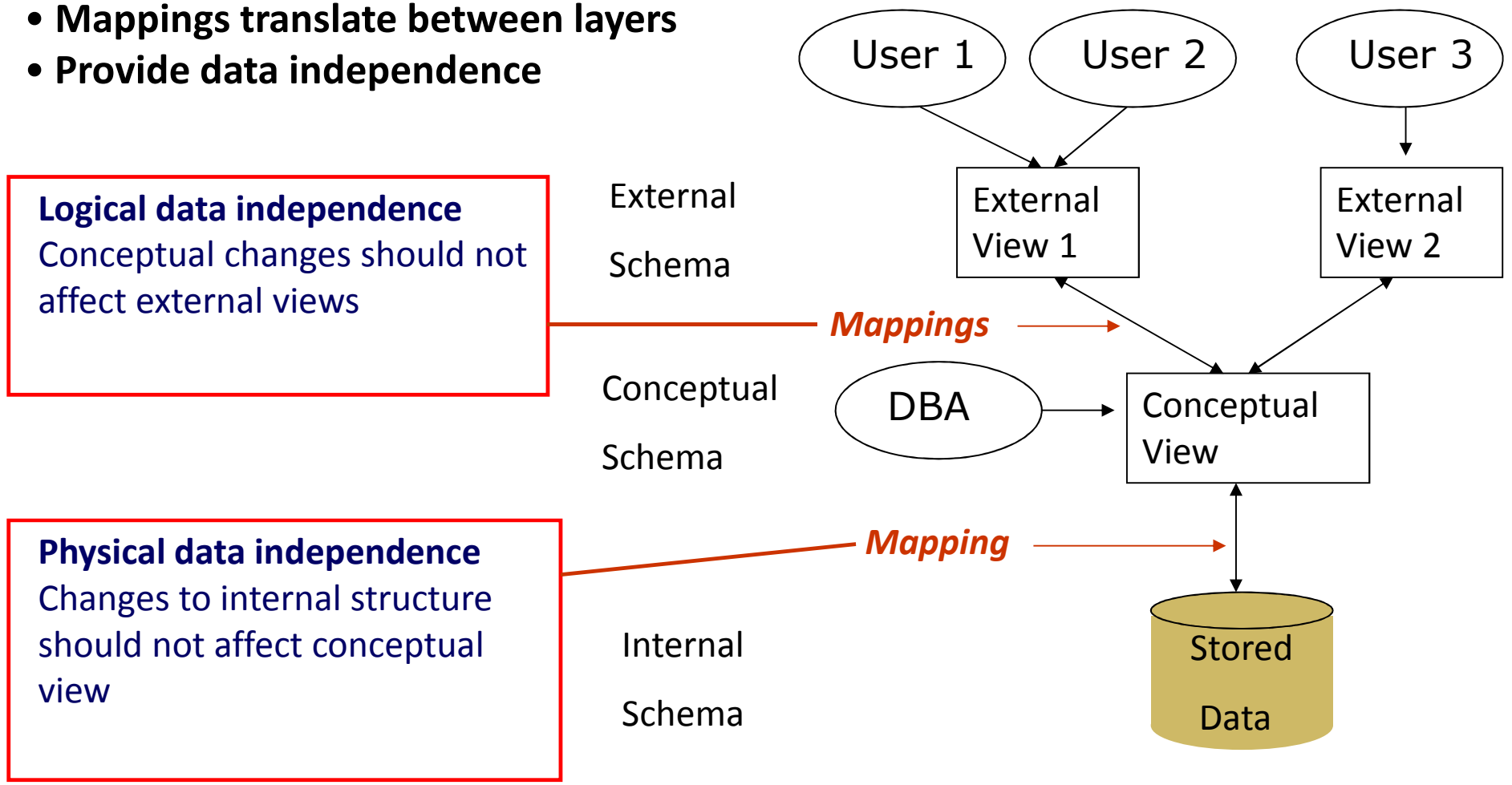
- Proposed a framework for DBMS in 1975
  - American National Standards Institute
  - Standards Planning Requirements Committee
- Three tier/level architecture
  - External level - for database users
  - Conceptual level - for database designers
  - Internal level - for systems designers

Mappings and users



Mappings and users

- **Mappings translate between layers**
- **Provide data independence**



Conceptual Level (Middle)

- Deals with the organisation of the entire database content
  - Used by database administrators and application programmers
  - Abstractions are used to remove unnecessary details of the internal level (i.e. file formats)
- Database holds metadata (data about data)
  - E.g. format of tables
- Conceptual schema example:

```
CREATE TABLE Employee (
  Name VARCHAR(25), Salary REAL,
  Department VARCHAR(10) )
```

External Level

- Provides the view determined by the user
  - Data may be hidden
  - Data may be presented in a suitable form
  - Used by users and applications programmers

- External Schema example:

```
Create View myView as {
  SELECT Name FROM Employee
}
```

Internal Level

- Deals with physical storage of data
  - Structure of records on disk - files, pages, blocks
  - Indexes and ordering of records
- Used by database system programmers
- Internal Schema example:

```
RECORD EMP
LENGTH=44
HEADER: BYTE(5) OFFSET=0
NAME: BYTE(25) OFFSET=5
SALARY: FULLWORD OFFSET=30
DEPT: BYTE(10) OFFSET=34
```

Example Modern DBMSs

- Database Management System (DBMS): the software that implements a database
- Examples:
  - Oracle
  - DB2
  - MySQL
  - Ingres
  - PostgreSQL
  - Microsoft SQL Server
  - MS Access ? (cut-down DBMS)

DBMS User Facilities

- Allow users to:
  - Store data
  - Manage change (updates)
  - Organise data
  - Retrieve data
  - Retain privacy (security)
- And they expect it to always work!
  - i.e. recover from errors, avoid multi-user problems, inform if something goes wrong,…

DBMS Functions

- Data storage, retrieval and update
- User accessible catalog
- Transaction support (all or nothing)
- Concurrency control (correct updates)
- Recovery services (if something goes wrong)
- Authorisation services (security)
- Support communication software (remote applications)
- Integrity services (allow rules to be enforced)
- Promote data independence (from structure)
- Utility services (import/export, monitoring and logs, statistical analysis, consolidate files/indexes, reporting tools,…)

Provided Languages

- Data Definition Language (DDL)
  - Specify database format
- Data Manipulation Language (DML)
  - Specify and retrieve database contents
- Data Control Language (DCL)
  - Specify access controls

- Which are often all one piece of software

Module Overview

- The relational model – VERY IMPORTANT
- Entity relationship modelling
- SQL data definition and data manipulation language
- PHP – for using SQL in a web page
- Handling NULL values
- Normalisation
- Transactions
- Concurrency
- Security
- Efficiency and storage
- Alternative modern databases

Relations and Relational Algebra

G51DBS Database Systems

Jason Atkin

Information is on the Moodle page

Email: jaa@cs.nott.ac.uk

This Lecture

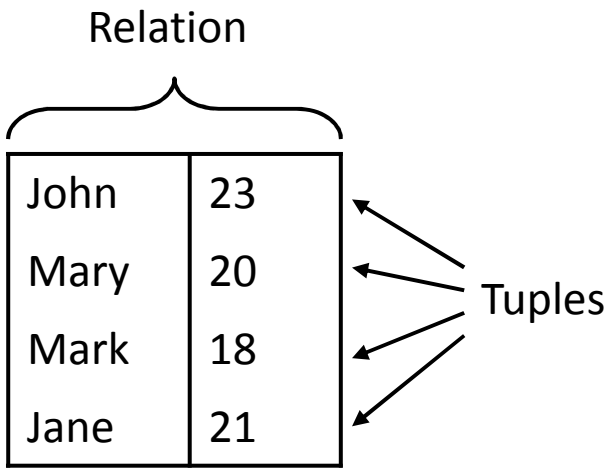
- The Relational Model
  - Relational data structures
- Relational algebra
  - Union, Intersection and Difference
  - Product of Relations
  - Projection, Selection
- **I will be asking questions later**
- Further reading
  - Database Systems, Connolly & Begg, 4.2 and 5.1
  - The Manga Guide to Databases, Chapter 2

The Relational Model

- Introduced by E.F. Codd in his paper "A Relational Model of Data for Large Shared Databanks", 1970
- The foundation for most (but not all) modern database systems

Relational Data Structure

- Data is stored in *relations* (tables)
- Data takes the form of *tuples* (rows)
  - The order of tuples is not important
  - There must not be duplicate tuples



Relations

- We will use tables to represent relations
- This is an example relation between people and email addresses:

Anne	aaa@cs.nott.ac.uk
Bob	bbb@cs.nott.ac.uk
Chris	ccc@cs.nott.ac.uk

Relations

- In general, each column has a *domain*, a set from which all possible values for that column can come
- For example, each value in the first column below comes from the set of first names

Anne	aaa@cs.nott.ac.uk
Bob	bbb@cs.nott.ac.uk
Chris	ccc@cs.nott.ac.uk

Relations

- A mathematical relation is a set of tuples: sequences of values. Each tuple represents a row in the table:

Anne	aaa@cs.nott.ac.uk	0115 911 1111
Bob	bbb@cs.nott.ac.uk	0115 922 2222
Chris	ccc@cs.nott.ac.uk	0115 933 3333

- {<Anne, aaa@cs.nott.ac.uk, 01159111111>,
  <Bob, bbb@cs.nott.ac.uk, 01159222222>,
  <Chris, ccc@cs.nott.ac.uk, 01159333333>}

Terminology

- **Degree of a relation**: how long each tuple is, or how many columns the table has
  - In the first example (name, email), the degree of the relation is 2
  - In the second example (name, email, phone) the degree of the relation is 3
  - Degrees of 2, 3, ... are often called Binary, Ternary, etc.
- **Cardinality of a relation**: how many different tuples there are, or how many rows a table has

Degree and Cardinality

- What is the degree of the following relation?
- What is its cardinality?

Anne	aaa@cs.nott.ac.uk	0115 911 1111
Bob	bbb@cs.nott.ac.uk	0115 922 2222

- {<Anne, aaa@cs.nott.ac.uk, 01159111111>,
  <Bob, bbb@cs.nott.ac.uk, 01159222222>}

Mathematical Definition

- Here is the mathematical definition of a relation R of degree n, where values come from domains A_1, ..., A_n:

  $R \subseteq A_1 \times A_2 \times \ldots \times A_n$

  (i.e. a relation is a subset of the Cartesian product of domains)

Cartesian product: $A_1 \times A_2 \times \ldots \times A_n =$

$\{ \langle a_1, a_2, \ldots, a_n \rangle : a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n \}$

Data Manipulation

- Data is represented as relations
- Manipulation of this data (through updates and queries) corresponds to operations on relations
- Relational algebra describes these operations. **These operations take relations as arguments, and produce new relations**
- Relational algebra contains two types of operators:
  - Common, set-theoretic operators (from ‘set theory’)
  - Operators specific to relations (new operators)

Union

- Standard set-theoretic definition of union:

  $A \cup B = \{x: x \in A \text{ or } x \in B\}$

- For example, $\{a,b,c\} \cup \{a,d,e\} = \{a,b,c,d,e\}$
- For relations, we require the results to be in the form of another relation.
- In order to take a union of relations R and S, R and S must have the same number of columns and corresponding columns must have the same domains

Union-compatible Relations

- Two relations R and S are **union-compatible** if:
  - They have the same number of columns
  - Corresponding columns have the same domains

Union-compatible Example

Same number of columns and matching domains

Anne	1970
Bob	1971
Chris	1972

Tom	1980
Sam	1985
Steve	1986

Not union-compatible Example

Different numbers of columns

Anne	1970	NG7
Bob	1971	NG16
Chris	1972	NG21

Tom	1980
Sam	1985
Steve	1986

Not union-compatible Example

Corresponding columns have different domains

Anne	NG7
Bob	NG16
Chris	NG21

Tom	1980
Sam	1985
Steve	1986

Unions of Relations

- Let R and S be two union-compatible relations. The Union $R \cup S$ is a relation containing all tuples from both relations:

  $R \cup S = \{x: x \in R \textbf{ or } x \in S\}$

- Note that union is a **partial operation** on relations. That is, it is only defined for some (i.e. compatible) relations
- This is similar to division of numbers: division by zero is undefined

Union Example

R

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00

S

Cream	2.00
Soap	1.00

R ∪ S

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00
Cream	2.00

Difference of Relations

- Let R and S be two union-compatible relations. The **difference** R - S is a relation containing all tuples from R that are not in S:

  $R - S = \{x: x \in R \text{ and } x \notin S\}$

- This is also a partial operation on relations

Difference Example

R

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00

S

Cream	2.00
Soap	1.00

R - S

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20

Intersection of Relations

- Let R and S be two union-compatible relations. The **intersection** $R \cap S$ is a relation containing all tuples that are in both R and S:

  $R \cap S = \{x: x \in R \text{ and } x \in S\}$

- This is also a partial operation on relations

Intersection Example

R

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cap S$

Soap	1.00

Intersection Example

R

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cap S$

Soap	1.00

Cartesian Product

- Cartesian product is a total operation on relations
  - Can be applied to any relations
  - Can be applied to relations of any relative size
- Set-theoretic definition of product:

  $R \times S = \{\langle x, y\rangle : x \in R, y \in S\}$
- For example:

  if <Cheese, 1.34> $\in$ R

  and <Soap, 1.00> $\in$ S then

  <<Cheese,1.34>,<Soap,1.00>> $\in$ R x S

Extended Cartesian Product

- **Extended** Cartesian product flattens the result into a single tuple. For example:

  <Cheese, 1.34, Soap, 1.00>

- This is more useful for relational databases
- **For the rest of this course, "product" will mean extended Cartesian product**
- **We will use these a LOT**
  - You will need to understand them well

Extended Cartesian Product of Relations

- Let R be a relation with column domains $\{A_1,...,A_n\}$ and S a relation with column domains $\{B_1,...,B_m\}$
- Their extended Cartesian product R x S is a relation:

$$R \times S = \{<c_1, \ldots, c_n, c_{n+1}, \ldots, c_{n+m}> : <c_1, \ldots, c_n> \in R, \quad <c_{n+1}, \ldots, c_{n+m}> \in S \}$$

Product Example

R

Cheese	1.34
Milk	0.80
Bread	0.60
Eggs	1.20
Soap	1.00

S

Cream	2.00
Soap	1.00

R x S

Cheese	1.34	Cream	2.00
Milk	0.80	Cream	2.00
Bread	0.60	Cream	2.00
Eggs	1.20	Cream	2.00
Soap	1.00	Cream	2.00
Cheese	1.34	Soap	1.00
Milk	0.80	Soap	1.00
Bread	0.60	Soap	1.00
Eggs	1.20	Soap	1.00
Soap	1.00	Soap	1.00

Projection

- Sometimes using all columns in a relation is unnecessary
- Let R be a relation with n columns, and X be a set of column identifiers. The projection of R on X is a new relation $\pi_X(R)$ that only has the columns in X
- For example, $\pi_{1,2}(R)$ is a relation that contains only the 1st and 2nd columns of R
- We can use numbers or names to index columns (naming columns will be discussed in the next lecture)

Projection Example

R

1	2	3
Anne	aaa@cs.nott.ac.uk	0115 911 1111
Bob	bbb@cs.nott.ac.uk	0115 922 2222
Chris	ccc@cs.nott.ac.uk	0115 933 3333

$\pi_{1,3}(R)$

Anne	0115 911 1111
Bob	0115 922 2222
Chris	0115 933 3333

Selection

- Sometimes we want to select tuples based on one or more criteria
- Let R be a relation with n columns, and α is some (any) property of tuples
- **Selection from R subject to condition α** is defined as:

$$\sigma_{\alpha}(R) = \{ <a_1,\ldots,a_n> \in R : \alpha(a_1,\ldots,a_n) \}$$

Comparison Properties

- We assume that properties are written using expressions of the form:

  col(i) Θ col(j) e.g. col(3) < col(1)

  or col(i) Θ v e.g. col(2) = 'Nolan'

  and {and, or, not}

  - Where Θ is a comparator which makes sense when applied to values from columns i and j. Often these will be: $=, \neq, \leq, \geq, <, >$
  - i, j are column numbers
  - v is a value from domain A_i

Meaningful Comparisons

- Comparisons between values can only take place where it makes sense to compare them
  - We can always perform an equivalence test between two values in the same domain
  - In some cases you can compare values from different domains, e.g. if both are strings
- For example:

  1975 < 1987 is a meaningful comparison
  “Anne” = 1981 is not meaningful
- We can only use a comparison in a selection if its result is true or false, never undefined

Selection Example

- $\sigma_{col(3) < 2002 \text{ and } col(2) = Nolan}(R)$

R

Insomnia	Nolan	2002
Magnolia	Anderson	1999
Insomnia	Skjoldbjaerg	1997
Memento	Nolan	2000
Gattaca	Niccol	1997

Selection Example

- $\sigma_{\text{col(3) < 2002 and col(2) = Nolan}}(R)$

R

Insomnia	Nolan	2002
Magnolia	Anderson	1999
Insomnia	Skjoldbjaerg	1997
→ Memento	Nolan	2000
Gattaca	Niccol	1997

Selection Example

- $\sigma_{col(3) < 2002 \text{ and } col(2) = Nolan}(R)$

$\sigma_{col(3) < 2002 \text{ and } col(2) = Nolan}(R)$

Memento	Nolan	2000

Other Operations

- Not all SQL queries can be translated into the relational algebra operations defined in this lecture

- Extended relational algebra includes counting, joins and other additional operations

Examples

Union Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cup S$?

Union Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cup S$

Cheese	1.34
Soap	1.00
Cream	2.00

Difference Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

R – S ?

Difference Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

R - S

Cheese	1.34

Intersection Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cap S$?

Intersection Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

$R \cap S$

Soap	1.00

Product Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

R x S ?

Product Example

R

Cheese	1.34
Soap	1.00

S

Cream	2.00
Soap	1.00

R x S

Cheese	1.34	Cream	2.00
Soap	1.00	Cream	2.00
Cheese	1.34	Soap	1.00
Soap	1.00	Soap	1.00

Projection Example

R

1	2	3
Anne	aaa@cs.nott.ac.uk	0115 911 1111
Bob	bbb@cs.nott.ac.uk	0115 922 2222
Chris	ccc@cs.nott.ac.uk	0115 933 3333

$\pi_{2,3}(R)$?

Projection Example

R

1	2	3
Anne	aaa@cs.nott.ac.uk	0115 911 1111
Bob	bbb@cs.nott.ac.uk	0115 922 2222
Chris	ccc@cs.nott.ac.uk	0115 933 3333

$\pi_{2,3}(R)$

aaa@cs.nott.ac.uk	0115 911 1111
bbb@cs.nott.ac.uk	0115 922 2222
ccc@cs.nott.ac.uk	0115 933 3333

Selection Example

- $\sigma_{col(1) = Insomnia \text{ and } col(3) > 1997}(R)$?

R

Insomnia	Nolan	2002
Magnolia	Anderson	1999
Insomnia	Skjoldbjaerg	1997
Memento	Nolan	2000
Gattaca	Niccol	1997

Selection Example

- $\sigma_{col(1) = Insomnia \text{ and } col(3) > 1997}(R)$

R

Insomnia	Nolan	2002
~~Magnolia~~	~~Anderson~~	~~1999~~
Insomnia	Skjoldbjaerg	1997
~~Memento~~	~~Nolan~~	~~2000~~
~~Gattaca~~	~~Niccol~~	~~1997~~

Selection Example

- $\sigma_{col(1) = Insomnia \text{ and } col(3) > 1997}(R)$

R

Insomnia	Nolan	2002
~~Magnolia~~	~~Anderson~~	~~1999~~
~~Insomnia~~	~~Skjoldbjaerg~~	~~1997~~
~~Memento~~	~~Nolan~~	~~2000~~
~~Gattaca~~	~~Niccol~~	~~1997~~

This Lecture in Exams

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

R

Anne	111111
Bob	222222

S

Chris	111111
Dan	222222

Note: Alternative questions could involve translating between Relational Algebra and SQL (later lectures)

Answer to the question

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

R

Anne	111111
Bob	222222

S

Chris	111111
Dan	222222

Answer to the question

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

R x S

Anne	111111	Chris	111111
Anne	111111	Dan	222222
Bob	222222	Chris	111111
Bob	222222	Dan	222222

Answer to the question

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

$\sigma_{col(2) = col(4)}(R \times S)$

Anne	111111	Chris	111111
Bob	222222	Dan	222222

Answer to the question

What is the result of the following operation

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$, where R and S are:

$\pi_{1,3}(\sigma_{col(2) = col(4)}(R \times S))$

Anne	Chris
Bob	Dan

Next Lecture

- Further reading (Normally 2 hours per week)
  - Database Systems, Connolly & Begg, Chapter 4.2
  - The Manga Guide to Databases, Chapter 2
- The Relational Model
  - More on Relations
  - Relational data integrity
- Entity/Relationship models
  - Entities and Attributes
  - Relationships
  - Attributes
  - E/R Diagrams