Introduction to Relational Databases

G51DBI – Databases and Interfaces

Yorgos Tzimiropoulos

yorgos.tzimiropoulos@nottingham.ac.uk

This Lecture

- Introduction to Databases
  - Definition, motivation, basic features
  - Database Management Systems
- The Relational Model
  - Relational data structures
  - Candidate, Primary and Foreign Keys
  - Entity and Referential Integrity

2

What is a Database?

- "A collection of data arranged for ease and speed of search and retrieval."
  - American Heritage Science Dictionary
- "A structured set of data held in computer storage"
  - 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

3

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
- and many more...

4

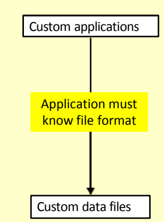
Why Study Databases?

- 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 Computer Science
- Basic concepts and skills with database systems are part of the skill set you will be ***assumed*** to have as a CS graduate

5

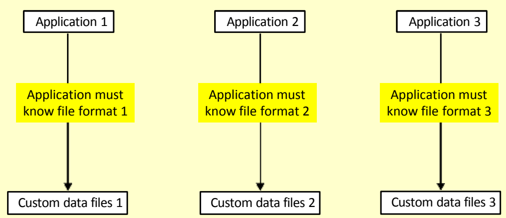
The early days...

- File-based system
- 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



6

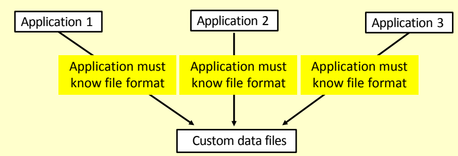
Multiple applications



7

One copy of the data

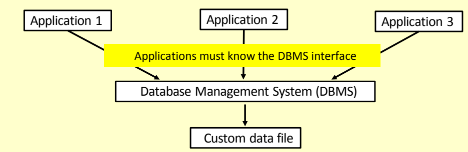
- So, keep one copy of data...
  - All applications must know the file format



8

Put something in the middle...

- A program in the middle can coordinate access
- Applications link with DBMS rather than data files



9

Definitions

- **Database**: a shared collection of logically related data and a description of the data designed to meet the needs of an organisation
- **Database Management System (DBMS)**: a software system that enables users to define, create, maintain and access the database
- **Applications program**: A program that interacts with a database through the DBMS by an appropriate request

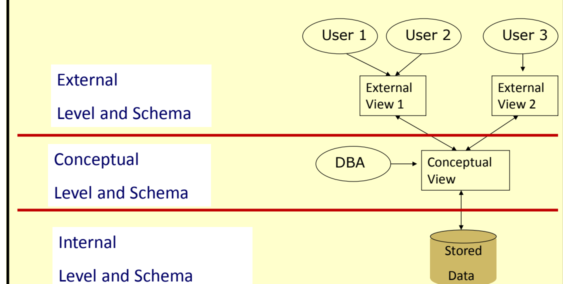
10

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

11

ANSI / SPARC Architecture



12

External Level

- Defines the user's view of the database (the part of the database that is relevant to each user)
  - 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
}
```


Conceptual Level (Middle)

- Defines what data is stored in the database and the relationships between data (e.g. table definitions, constraints on the data, security and integrity information)
- Deals with the organisation of the entire database content
- Conceptual schema example:

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

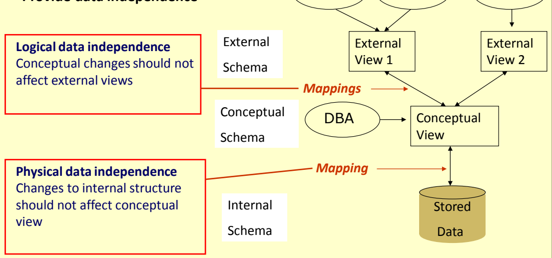

Internal Level

- Defines how data is stored in the database (storage space allocation, data structures, indexing, data compression, encryption etc.)
- Used by database system programmers
- Internal Schema example:

```
Struct  employee{
  char fNAME [15];
  char lNAME [15];
  float salary;
  char dept;};
```


Mappings and users

- Mappings translate between layers
- Provide data independence



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
- Which are often all one piece of software (i.e. Structured Query Language or SQL)

Example Modern DBMSs

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

19

Relational databases

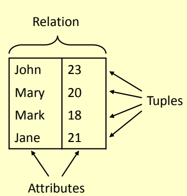
The Relational Model

- 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
- Information stored as records in relations (tables)
  - Sound mathematical basis
- Model covers data:
  - Structure
  - Integrity
  - Manipulation

21

Relational Data Structure

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



22

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

Andrew	aaa@cs.nott.ac.uk
Bill	bbb@cs.nott.ac.uk
Christine	ccc@cs.nott.ac.uk

23

Terminology

- **Degree of a relation**: how long each tuple is, or how many columns the table has
  - In the previous example (name, email), the degree of the relation is 2
- **Cardinality of a relation**: how many different tuples there are, or how many rows a table has

24

Schemas and Attributes

- It is often helpful to reference columns using names, which we will have to provide
- *Attributes* are named columns in a relation
- A *schema* defines the attributes for a relation

Relational Data Structure

- Each relation has a *schema* (sometimes called a scheme or heading)
- The schema defines the relation's *attributes* (columns).

Relation

Name	Age
John	23
Mary	20
Mark	18
Jane	21

Schema

Tuples

Attributes

Named and Unnamed Tuples

- Tuples specify values for each attribute in a relation
- When writing tuples down, they can be named as sets of pairs, e.g.
  - { (Name, John), (Age, 23) }
- Or unnamed, for convenience, e.g.
  - (John, 23) (equivalent to the above)
- There is no real difference between named and unnamed tuples, but be careful with the ordering of unnamed tuples.

Relational Data Structure

- More formally:
  - A schema is a set of attributes
  - A tuple assigns a value to each attribute in the schema
  - A relation is a set of tuples with the same schema

Name	Age
John	23
Mary	20
Mark	18
Jane	21

{ { (Name, John), (Age, 23) },
{ (Name, Mary), (Age, 20) },
{ (Name, Mark), (Age, 18) },
{ (Name, Jane), (Age, 21) } }

Example Relation

ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

Example Relation

ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

Schema is { ID, Name, Salary, Department }

Example Relation

Attributes are ID, Name, Salary and Department

The degree of the relation is 4

ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

Schema is { ID, Name, Salary, Department }

Example Relation

Attributes are ID, Name, Salary and Department

The degree of the relation is 4

ID	Name	Salary	Department
M139	John Smith	18,000	Marketing
M140	Mary Jones	22,000	Marketing
A368	Jane Brown	22,000	Accounts
P222	Mark Brown	24,000	Personnel
A367	David Jones	20,000	Accounts

Schema is { ID, Name, Salary, Department }

Tuples, e.g.
{ (ID, A368),
(Name, Jane Brown),
(Salary, 22,000),
(Department, Accounts)}

The cardinality of the relation is 5

Candidate Keys

- A set of attributes in a relation is a candidate **key** if, and only if:
  - Every tuple has a unique value for that set of attributes: ***uniqueness***
  - No proper subset of the set has the uniqueness property: ***minimality***

ID	First	Last
S139	Alan	Carr
S140	Jo	Brand
S141	Alan	Davies
S142	Jimmy	Carr

Candidate key is {ID}; {First, Last} looks plausible, but people might have the same name

{ID, First}, {ID, Last} and {ID, First, Last} satisfy uniqueness, but are not minimal

{First} and {Last} do not give a unique identifier for each row

Choosing Candidate Keys

- You can't necessarily infer the candidate keys based solely on the data in your table
  - More often than not, an instance of a relation will only hold a small subset of all the possible values
- You must use knowledge of the real-world to help

Choosing Candidate Keys

What are the candidate keys of the following relation?

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Choosing Candidate Keys

The candidate keys are {OfficeID}, {Phone} {Name, Postcode/Zip} {Name, Country}

CompanyOffices

officeID	Name	Country	Postcode/Zip	Phone
O1001	Headquarters	England	W1 1AA	0044 20 1545 3241
O1002	R&D Labs	England	W1 1AA	0044 20 1545 4984
O1003	US West	USA	94130	001 415 665981
O1004	US East	USA	10201	001 212 448731
O1005	Telemarketing	England	NE5 2GE	0044 1909 559862
O1006	Telemarketing	USA	84754	001 385 994763

Note: Keys like {Name, Postcode/Zip, Phone} satisfy uniqueness, but not minimality

Primary Keys

- One candidate key is usually chosen to identify tuples in a relation
- This is called the *Primary Key or just Key*
- Often a special ID is used as the Primary Key

ID	First	Last
S139	Alan	Carr
S140	Jo	Brand
S141	Alan	Davies
S142	Jimmy	Carr

We might use either {ID} or {First,Last} as the primary key. ID is more convenient as we know it will always be unique. People could have the same name

NULLs and Primary Keys

- Missing information can be represented using NULLs
- A NULL indicates a missing or unknown value
- NULL is not the same as 0 or blank space character
- *Entity integrity:* Primary Keys cannot contain NULL values (why? Because it contradicts the notion of the key)

Foreign Keys

- Foreign Keys are used to link data in two relations. A set of attributes in the first (referencing) relation is a Foreign Key if its value:
  - Matches a Primary/Candidate Key value in a second (referenced) relation
  - Is NULL
- This is called *Referential Integrity*

Foreign Keys Example

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

{DID} is a Candidate Key for Department – Each entry has a unique value for DID

{DID} is a Foreign Key in Employee – each employee's DID value is either NULL, or matches an entry in the Department relation. This links each Employee to at most one Department

Referential Integrity

- When relations are updated, referential integrity might be violated
- This usually occurs when a referenced tuple is updated or deleted
- There are a number of options when this occurs:
  - RESTRICT – stop the user from doing it
  - CASCADE – let the changes flow on
  - SET NULL – make referencing values null
  - SET DEFAULT – make referencing values the default for their column

Referential Integrity Example

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department
- Using RESTRICT, CASCADE and SET NULL

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

RESTRICT

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

RESTRICT

- RESTRICT stops any action that violates integrity
  - You cannot update or delete Marketing or Accounts
  - You *can* change Personnel as it is not referenced

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

CASCADE

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

CASCADE

- CASCADE allows the changes made to flow through
  - If Marketing's DID is changed to 16 in Department, then the DIDs for John Smith and Mark Jones also change
  - If Accounts is deleted then so is Mary Brown

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

Employee

EID	EName	DID
15	John Smith	~~13~~ 16
~~16~~	~~Mary Brown~~	~~14~~
17	Mark Jones	~~13~~ 16
18	Jane Smith	NULL

SET NULL

- What happens if
  - Marketing's DID is changed to 16 in Department?
  - The entry for Accounts is deleted from Department

Department

DID	DName
13	Marketing
14	Accounts
15	Personnel

Employee

EID	EName	DID
15	John Smith	13
16	Mary Brown	14
17	Mark Jones	13
18	Jane Smith	NULL

SET NULL

- SET NULL allows the changes to happen but
  - If Marketing's DID is changed to 16 in Department, then the DIDs for John Smith and Mark Jones is set to NULL
  - If Accounts is deleted then Mary Brown's DID is set to NULL

Department

DID	DName
~~13~~ 16	Marketing
~~14~~	~~Accounts~~
15	Personnel

Employee

EID	EName	DID
15	John Smith	~~13~~ NULL
16	Mary Brown	~~14~~ NULL
17	Mark Jones	~~13~~ NULL
18	Jane Smith	NULL

Take home messages

1. Databases are everywhere
2. Databases are important
   - Especially in the Big Data era
3. DBMS as software that implements a DB
4. We use Relational Databases
5. Basic data structure in RD is a table
6. Primary key is a way to identify a tuple in a table
7. Primary key satisfies Entity Integrity
8. Foreign key links different tables
9. Foreign key must satisfy Referential Integrity