COM106: Introduction to Databases

Database Basics (continued) & Data Models

Database Basics (continued) & Data Models

Some Examples

Log into Socrative **(COM106)** at socrative.com (or via the app)

Given the data table below, identify any **data dependencies** by selecting all of the statements that are **TRUE**

- **A** – Knowing **EName**, uniquely identifies **Manager**
- **B** – Knowing **Manager**, uniquely identifies **Dept#**
- **C** – Knowing **Salary**, uniquely identifies **DName**
- ✓ **D** – Knowing **DName**, uniquely identifies **Manager**
- **E** – Knowing **EName**, uniquely identifies **Dept#**
- **F** – None of the above

Is the reverse True?

Does knowing **Manager** uniquely identify **DName** ?

No

Does the table contain any more data dependencies?

Yes:-

Dept# gives **DName** gives **Manager**

Notice the **data duplication** in the table

Poor database design

Emp#	EName	Salary	Dept#	DName	Manager
1	Smith	15000	A	Sales	Kelly
2	Jones	21000	B	Admin	Whyte
3	Kane	18000	D	Finance	Coyle
4	Dwyer	20000	B	Admin	Whyte
5	Cook	16000	C	Personnel	Woods
6	Healy	15000	A	Sales	Kelly
7	Conroy	19000	B	Admin	Whyte
8	Price	22000	D	Finance	Coyle
9	Murphy	23000	C	Personnel	Woods
10	Jones	16000	A	Sales	Kelly

Database Basics (continued) & Data Models

An example of some of the problems of the file-based approach:

Consider a University, where each department maintains and processes it's own data.

Student Accommodation Records

FName	SName	Acc Type	Address	Grades
Joe	Blogs	Halls	Room D1	CCD
Mary	Smith	Flat	10	AAB
Jane	Jones	Halls	Room A3	A*CB
David	Banda	Flat	23	210

Accommodation Staff Records

Name	Position
Jennifer	Inspector
Pat	Inspector

Student Registration Records

FName	SName	Studt No	Address	Grades
Joe	Blogs	B00123	Room D1	CCD
Mary	Smith	B00456	Room A1	AAB
Jane	Jones	B00789	Room A3	A*CB

Payroll Records

Name	Position
Jennifer	Inspector
Pat	Cleaner

Database Basics (continued) & Data Models

Student Accommodation Records

FName	SName	Acc Type	Address	Grades
Joe	Blogs	Halls	Room D1	CCD
Mary	Smith	Flat	10	AAB
Jane	Jones	Halls	Room A3	A*CB
David	Banda	Flat	23	210

Accommodation Staff Records

Name	Position
Jennifer	Inspector
Pat	Inspector

Security

Data Correctness

Data Consistency

Data Duplication

No Data Centralisation
Possible incompatible file formats

Data Consistency

Student Registration Records

FName	SName	Studt No	Address	Grades
Joe	Blogs	B00123	Room D1	CCD
Mary	Smith	B00456	Room A1	AAB
Jane	Jones	B00789	Room A3	A*CB

Payroll Records

Name	Position
Jennifer	Inspector
Pat	Cleaner

Database Basics (continued) & Data Models

The Evolution of Databases – Database Approaches

The database approach overcomes many of the problems with the file-based approach

(From earlier) a **database** is a shared **collection** of **logically related data** stored in an **organised structure** which provides the ability to **interact** with data and to extract salient **information and knowledge**.

A database provides a **data-centred** approach, designed to meet the needs of an organisation, where data can be **shared** among different applications.

A **database management system** (DBMS) is

a software system for managing a database(s) ….

which allows the creation, manipulation (queries) and maintenance of data ….

and provides controlled access to the data (security, integrity, recovery etc).

Examples of common DBMS include:

Microsoft Access - Single user DBMS, supports limited database size

Microsoft SQL Server - Multiuser, supports larger databases

MySQL – Free, supports larger databases, often used as database server in web-based applications

Oracle - Large scale DBMS in multiuser environments, expensive, many supporting facilities

Database Basics (continued) & Data Models

Database Advantages (over File Handling Systems)

1. **Data Independence** - application programs insulated from data
2. **Centralised Data** - security and integrity maintained
3. **Access Flexibility** - query language and storage mechanisms
4. **Data Model** - reflects real world (network, hierarchical, relational etc.)

We'll look at each of these in more detail

Other DBMS Advantages ….

Enforcement of **Standards**

Data Sharing (different applications built on same data)

Economy of Scale (all applications built on common source data)

Increased Productivity (DBMS provides many standard functions)

…. and Disadvantages

Complexity (complex system requires level of understanding)

Cost (in large organisation with large number of records, cost of DBMS plus conversion and maintenance costs are high)

Higher **impact of failure** (*downside of centralisation*)

Database Basics (continued) & Data Models

DBMS Facilities

Application Programming Language – e.g. Visual Basic in MS Access

Report Generator (data analysis for management)

Data Dictionary (stores data definitions etc.)
independent of the stored data (supporting data independence)

Query Language - e.g. **SQL** in relational database DBMSs

Screen/Form Editor - allows creation of data entry screens (**views**)

Functions provided by DBMS

Data storage, retrieval & update

System Catalog/Data Dictionary - definition of all data objects

Transaction support - ensures correct update completion
- transaction **concurrency control** for multiuser access
- transaction **recovery** from failure of database

User authorisation

Provide access across a network

Provides **data integrity** and promote **data independence**

And a range of utilities (e.g. **Indexing**, Performance monitoring, file import/export)

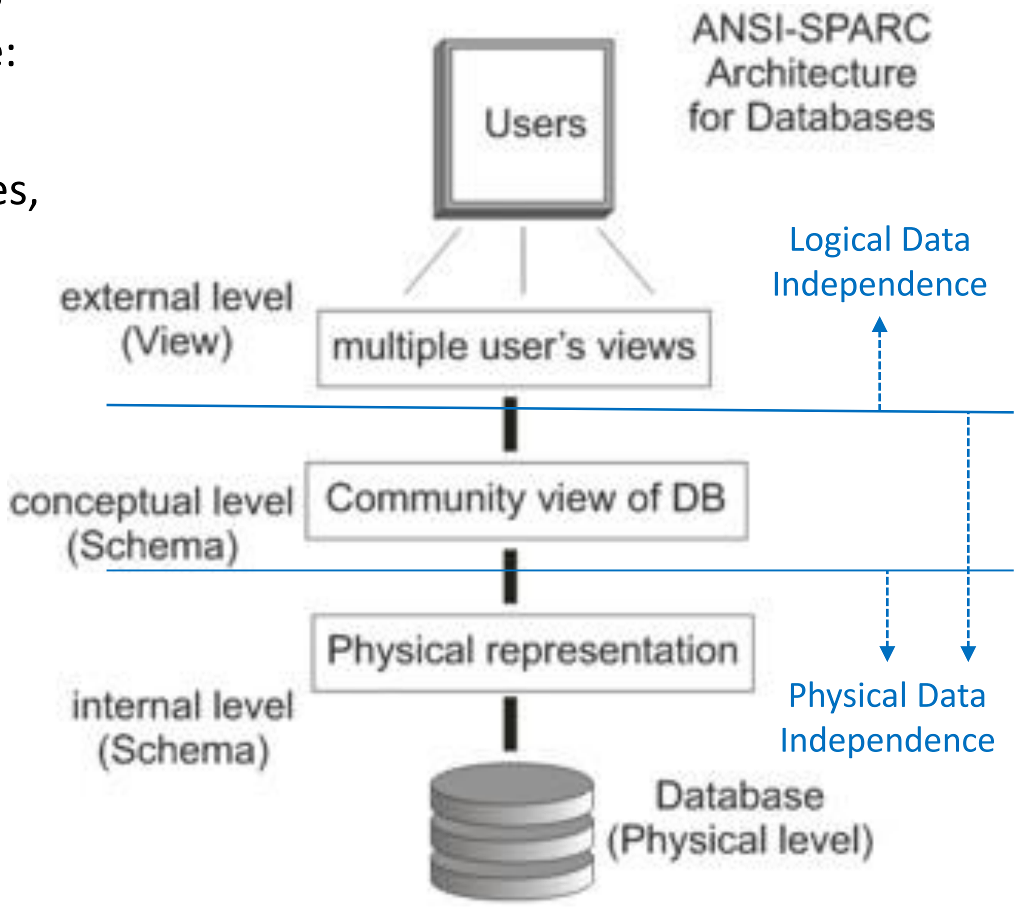
Database Basics (continued) & Data Models

Database Advantages - 1. Data Independence

Applications separated from data using a **3-layered model (ANSI SPARC)** of a database:

- **External Layer (Views)**: programs, queries, reports
- **Conceptual Layer (Database Schema)**: single representation of logically stored data (e.g. set of linked relational tables)
- **Internal Layer (Physical)**: definition of how data is stored and access paths to that data

Each layer is defined by a schema (set of definitions)


ANSI-SPARC Architecture for Databases
Users
external level (View)
multiple user's views
Logical Data Independence
conceptual level (Schema)
Community view of DB
Physical representation
internal level (Schema)
Physical Data Independence
Database (Physical level)

Three levels of **abstraction**

Database Basics (continued) & Data Models

The description of the database structure is the **database schema** – like a blueprint of how the database is constructed

Changes to the applications (queries/programs etc) do not necessarily require changes to the tables or stored records (logical independence)

Changes to the record storage or table design do not necessarily require changes to the applications (physical independence)

Database Advantages - 2. Centralised Data

DBMS is managed by a Database Administrator

Data is stored centrally, thereby allowing/requiring:

Security to be applied to users and database objects as appropriate (passwords, privileges, views)

Integrity of the data to be maintained through:

Integrity Constraints (e.g., input validation, referential integrity, etc)

transaction management

- concurrency control
- recovery (backup + automated recovery)

We'll study these topics later in the module

Database Basics (continued) & Data Models

Brief Aside – Transaction Processing

A **transaction** (key concept) is one or more atomic sequential operations that make up a single task

Operations - one of four categories (**CRUD**)

- **C**reate
- **R**ead (Query)
- **U**pdate
- **D**elete

An Example of a Transaction – ATM Withdrawal

Checks the Account Balance (**R**ead)

Account Table

Account#	Balance
123456789	£300

Reduces balance by withdrawal (**U**pdate) *Update Account – Set Balance - £20*

Logs the transaction (**C**reate)

Log Table

Transaction#	Account#	Action	Date
TR_89	123456789	-£20	28/11/15

Database Basics (continued) & Data Models

Database Advantages - 3. Access Flexibility

Powerful query language to allow ad hoc (immediate) querying of the database

- Query By Example - Easy to create visual queries
- **SQL** - powerful query language (starting in Week 3)

Storage structures and access paths

- e.g. Indexed Sequential, Hashing, B-Trees

We'll study these topics later in the module

Physical storage reorganisation (for large databases)

- in response to query performance (query optimisation)
- might involve a change of storage structure

We won't be studying these topics

Database Advantages - 4. Data Model

Data Model – How the database is organised to reflect the real world

Important real world objects (**entities**), and the associations and **relationships** between them, are modelled.

e.g. Employee works_in Department

Student enrolls_in Module

We'll study ER Modelling later in the module

Database Basics (continued) & Data Models

Different data models used:

Look at these data models in your own independent study

- Hierarchical
- Network
- **Relational** – uses the Relational Model to give flexible navigation through data relationships between tables
- Object oriented

The vast majority of databases in use today are Relational Databases

NoSQL are a new class of database system that is growing in popularity **for particular tasks**

For NoSQL --- read --- **'Not only SQL'**

Lots of different variations, and lots of different products (e.g., MongoDB, CouchDB, Neo4J, etc)

- Document Store
- Key Value Store
- Graph Databases

We focus on the Relational Model in this module. We won't look at NoSQL in much detail.

Designed to address problems which relational database are not good at tackling

- Need for a **flexible schema**
- Dealing with **enormous** amounts of **unstructured/semi-structured data** (Big data)
- Where the ability to **scale** is more important than maintaining **data consistency**.

Database Basics (continued) & Data Models

The Relational Model

The most widely used database type in the world today

Proposed by Codd (1970) - see paper in Additional Module Resources (seminal work)

Based on mathematical set theory which:

- Presents the user with a simple view of data as two-dimensional tables (aka - **relations**)
- Defines a set of **relational algebra operations** to **manipulate** relations (tables)

In relational databases, data is **manipulated** using Structured Query Language (**SQL**)

SQL was formalised by ANSI (American National Standards Institute) in 1986 and has a direct relationship to **relational algebra**.

We won’t study relational algebra in detail, but later we will look at its relationship to SQL.

Relational Model Terminology

Relation -	a named table with **attributes** (columns) and **tuples** (rows)
Attribute-	a named column of a relation (defined on a **domain**)
Tuple-	a row of a relation (with certain characteristics) - contains one **value** per attribute

FName	SName	Studt No	Grades
Joe	Blogs	B00123	CCD
Mary	Smith	B00456	AAB
Jane	Jones	B00789	A*CB

Database Basics (continued) & Data Models

Domain - the set of allowable **values** for one or more **attributes**

Represents all the values that can ever be used (not just the values present at any time) and specifies whether duplicate values are allowed

Domains cannot be precisely defined in database systems - **data types** are used instead when creating the tables (**generalisation**)

Integrity Constraints can be used to further refine legitimate values of a data type. E.g., an attribute holding **age** might be of data type **tinyint**, but additional constraints can be used to ensure that **values lie only between 0 and 99**.

Domains need to be known in order to relate data from different relational tables

- i.e. to relate data across two relations (**join**) requires that both attributes are
  - defined on the same domain (or have the same **data type**)

Some database terminology is used interchangeably. These terms are **approximately** equivalent:

Relational	Common	File-Based
Relation	Table	File
Attribute	Column	Field
Tuple	Row	Record
Domain		Data-Type

Database Basics (continued) & Data Models

Properties of Relations

Each relation has a distinct **name**

Each **attribute** has a distinct name

There are no duplicate **tuples**

Tuple order has no significance

Each **cell** has a single (atomic) value

Attribute values are all from the same **domain**

Attribute order is insignificant

Relation Schema (Table Plan)

A **relation schema** is the relation name followed by its attribute names in brackets

e.g. **PATIENT (patient_no, pat_name, condition, doctor)**

An instance of relation PATIENT (i.e. sample data):

patient_no	pat_name	condition	doctor
32947	Adams	Halitosis	Jekyll
59421	Brown	Influenza	Johnson
37983	Clark	Lurgi	Jekyll
10442	Doyle	Haemorrhoid	Johnson
72511	Evans	Amnesia	Who
26743	Brown	Lurgi	Johnson

patient_no	is defined on positive integers (>= 10000) (no duplicates)
pat_name	is defined on all valid surnames (with duplicates)
condition	is defined on all valid medical conditions (with duplicates)
doctor	is defined on all valid surnames (with duplicates)

Database Basics (continued) & Data Models

Keys

Keys are fundamental to the structure of a relation (table)

A **super key** is any combination of fields within a table that **uniquely** identifies each record within that table.

A **candidate key** of a relation is a subset of its attributes which have the **time-independent** rules:

1. **unique identification**: the key value uniquely identifies each tuple in the relation
2. **minimality** (non-redundancy): no attribute in the key can be discarded without destroying rule 1.

    i.e., a **candidate key (CK)** is the **minimum number of attributes** which together **uniquely identify** each tuple

    A candidate key of a relation is a subset of a super key.

Since each tuple in a relation is distinct **a key always exists** (even if it is a combination of all attributes)

A **primary key (PK)** is the candidate key chosen as the unique identifier in a relational table

i.e., ***the smallest possible combination of attributes that uniquely identifies every tuple (record) in a relational table*** (i.e. **unique & minimal**)

the definition of a primary key is the same as that of a candidate key

every primary key is a candidate key until it is chosen (there may be several CKs)

Database Basics (continued) & Data Models

A **Foreign Key** (FK) is an attribute in one relation matching the primary key in some other relation (used to express a **relationship**)

Consider the following relational schema:

EMP (emp_no, nat_ins_no, ename)

Employee Number (**emp_no**) and national Insurance Number (**nat_ins_no**) are normally unique - so they are **CKs**

One is selected as the **primary key** (e.g. **emp_no**)

Any others are **alternate keys** (e.g. **nat_ins_no**)

A candidate/primary key which contains two or more attributes is known as a **composite key**

To decide on a candidate/primary key you need to decide:

which attribute(s) have **unique** values

OR, if no single attribute has unique values, which combination of attributes has **unique** values

This requires knowledge of (or assumptions about) the ‘real world’ meaning (**domains**) of attributes (whether attributes have unique values)

Decisions on PKs may be made from **relational schema** alone or from tables with **sample data**

In practice ‘**artificial**’ attributes may be created as primary keys to replace composite keys in some circumstances – e.g. the student BCode

Database Basics (continued) & Data Models

What is the Primary Key (**PK**) of the relation:

EMPLOYEE (emp_no, ename, salary)

First, identify the candidate keys - do any attributes contain unique values?

emp_no - unless there is any evidence to the contrary we may assume that emp_no is created to identify employees uniquely

ename - There could be more than one employee with the same name (non unique values)

salary - More than one employee could have the same salary (non unique values)

So, one candidate key (**emp_no**), which is also the primary key (**PK**)

EMPLOYEE Relation (Table)

emp_no	ename	salary
1	J Smith	20000
2	B Foster	18000
3	P Allen	23000
4	D Kelly	20000
5	M Corry	18000
6	J Smith	27000

By inspection **emp_no** can be seen to have unique values

(and we assume that **all future values** will be unique – based on an assumption that the **domain** of emp_no contains unique values only)

Database Basics (continued) & Data Models

Another Example:

What is the primary key of the relation:

TAKES (student_no, module_code, exam_result)

Identify the candidate keys - do any attributes contain unique values?

NO - **student_no** - If a student takes several modules then the same student# will occur in more than one record

module_code - If a module has many students then the same module code will appear in many records

exam_result - Several students could have the same exam result

Do any combination of two attributes contain unique values?

YES – **student_no** & **module_code** – combined contain unique values (**composite key**) and are the **Primary Key**

student_no & **module_code** are the **Composite Primary Key** (assuming that this represents modules taken by students in a **single academic year**)

TAKES

student_no	module_code	exam_result
90923204	COM140J4	43
90923204	COM147J4	56
90923204	COM158J1	38
90924104	COM140J4	67
90924104	COM147J4	70
90924104	COM158J1	70
90927504	COM140J4	56
90927504	COM147J4	67
90927504	COM158J1	49

Database Basics (continued) & Data Models

Foreign Keys

Relationships between **relations** are represented by **foreign keys (FK)**, normally copies of primary keys

Consider two relations, **EMP** and **DEPT**

EMP (<u>emp_no</u>, ename, dept_no*)

DEPT (<u>dept_no</u>, dname)

Notice **dept_no** is in both tables - dept_no in **DEPT** is a primary key
dept_no in **EMP** is a **foreign key**

Primary keys are **<u>underlined</u>** in a relation

Foreign keys usually, but not necessarily, have the same attribute name as the corresponding primary key and are **not** underlined.

In this module we will adopt the convention of indicating a **FK** with an *****

To **link relations** foreign keys must contain only valid values of the corresponding primary key

Keys are essential in enforcing and maintaining **relational integrity** – c.f. **data integrity**

There are two important relational integrity rules

1. Entity Integrity: *No attributes participating in the primary key of a relation are allowed to accept* ***null*** *values.*

Null – represents a value for an attribute that is currently unknown or is not applicable for this tuple. It is not the same as 0 or string spaces - it is the **absence of a value**

Database Basics (continued) & Data Models

2. Referential Integrity: *If a relational table includes a foreign key (FK) matching the primary key (PK) of another relational table, then every value of the FK must:*

- *either be equal to a value of the PK in some tuple (row)*
- *or be wholly null*

An example - Consider two relations, **EMP** and **DEPT**

EMP (<u>emp_no</u>, ename, dept_no*)

DEPT (<u>dept_no</u>, dname)

Tables are linked by a FK

EMP

emp_no	ename	dept_no
1	Smith	A
2	Jones	A
3	White	B
4	Brown	B

DEPT

dept_no	dname
A	Sales
B	Admin

A foreign key value (e.g. C) cannot be entered in **dept_no** from the **EMP** table unless it already exists in **dept_no** in the **DEPT** table.

i.e. an employee cannot be assigned to a department unless that department exists

The FK can be **NULL** – employee not yet assigned to a department

Database Basics (continued) & Data Models

What happens if a row referenced by a FK in another table is deleted?

Consider the **EMP** and **DEPT** tables above, linked by an **FK/PK match on dept_no**.

What happens if the row **B, Admin** is deleted from **DEPT**?

By default, in SQL Server, this is **Not Permitted** – an error is raised.

Referential integrity would be breached (FK values in **EMP** can't reference PK values in **DEPT** that don't exist!)

What happens ON DELETE can be specified when the FK constraint is created (or altered)

NO ACTION –	Default. An error is raised and no deletion occurs.
CASCADE -	Corresponding rows are deleted from the referencing table (**EMP**) if that row is deleted from the parent table (**DEPT**).
SET NULL -	All the values that make up the FK are set to **NULL** when the corresponding row in the parent table is deleted.
SET DEFAULT -	All the values that comprise the FK are set to their **default values** when the corresponding row in the parent table is deleted.