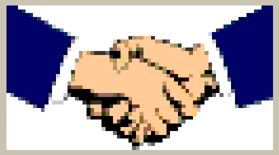
DATABASE FUNDAMENTALS

REVISION LECTURE



Learning outcomes . . . .

(Knowledge) Having succesfully completing this module the student will be able to:

- describe the architecture of a relational database
- define the terminology and concepts associated with relational databases
- explain various aspects of transaction processing
- define and describe **SQL**

(Skills) Having successfully completed this module, the student will be able to:

- Model database requirements using an ERD
- produce a **normalised** set of tables
- query and manipulate database objects (using **SQL**)

Topics

Topic 1: Features of a Relational Database

Topic 2: SQL

Topic 3: Database Design
(ERDs and Normalisation)

Topic 4: Transaction Processing, Security

Software: MySQL

Topics

Topic 1: Features of a Relational Database (lecture 1)

- What is Data
- What is a Database
- Relational database - tables, rows, cells etc.
- DBMS
- MySQL
- DB instance
- Database System
- Advantages of a database

What is a database?

- At its simplest, a tool to store **data** permanently, i.e. a persistent data store

What is data?

- Data is a collection of facts, such as values or measurements
- It can be numbers, words, measurements, observations or even just descriptions of things.

Databases

The focus of this course is

how to store data

Efficiently,

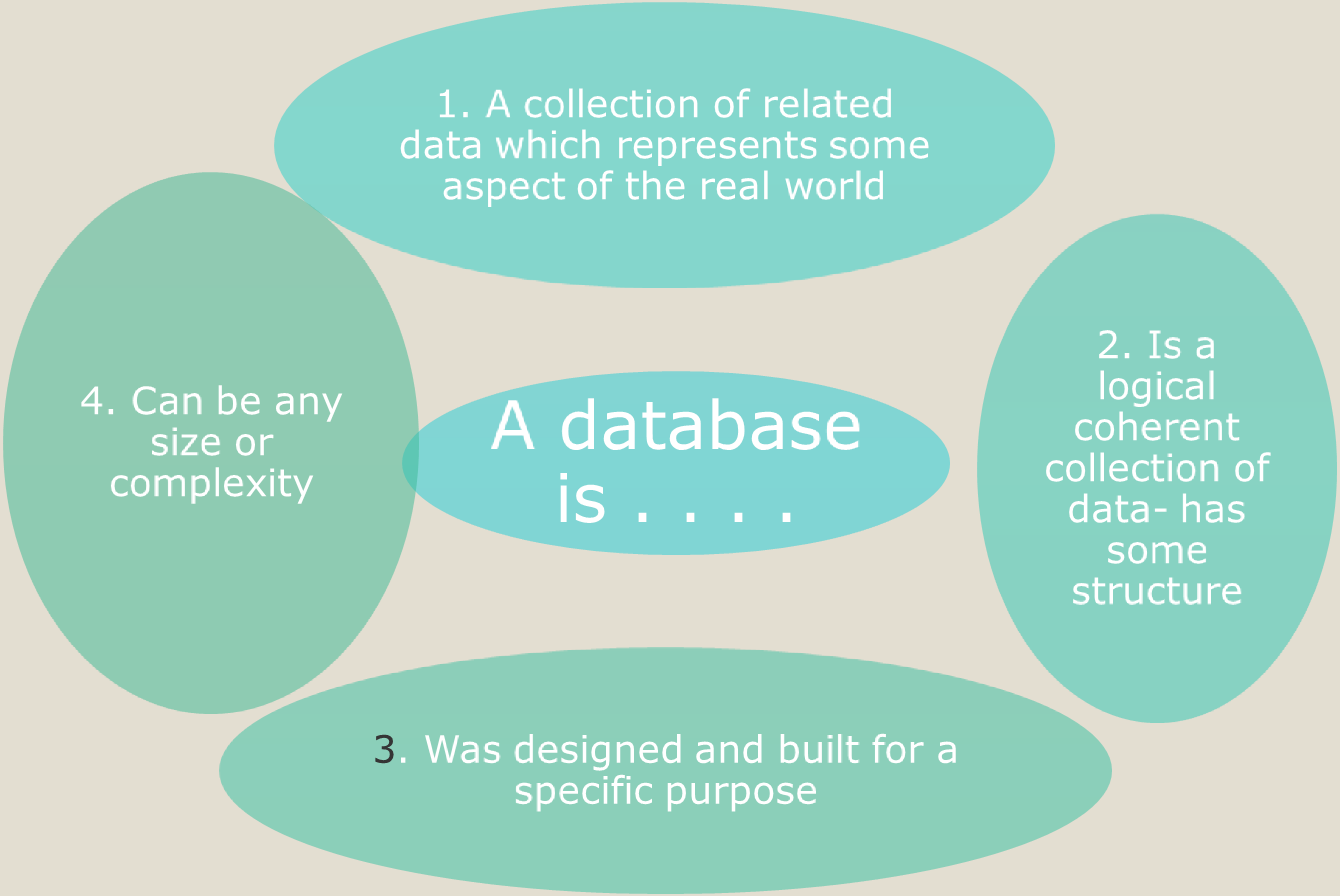
Accurately

and

Securely

so that it can be accessed easily from software programs

Some Terminology - database



A database is . . . .

1. A collection of related data which represents some aspect of the real world

2. Is a logical coherent collection of data- has some structure

3. Was designed and built for a specific purpose

4. Can be any size or complexity

Cells, Rows, Tables and Databases

- Database -- a collection of related tables describing various facets of a group of objects or events.

Database

Student Table

StudentID	StudentName	CourseCode
B00001234	Joe Bloggs	BN002
B00051413	Ann Ryan	BN001
B00012136	John Smith	BN005

Course Table

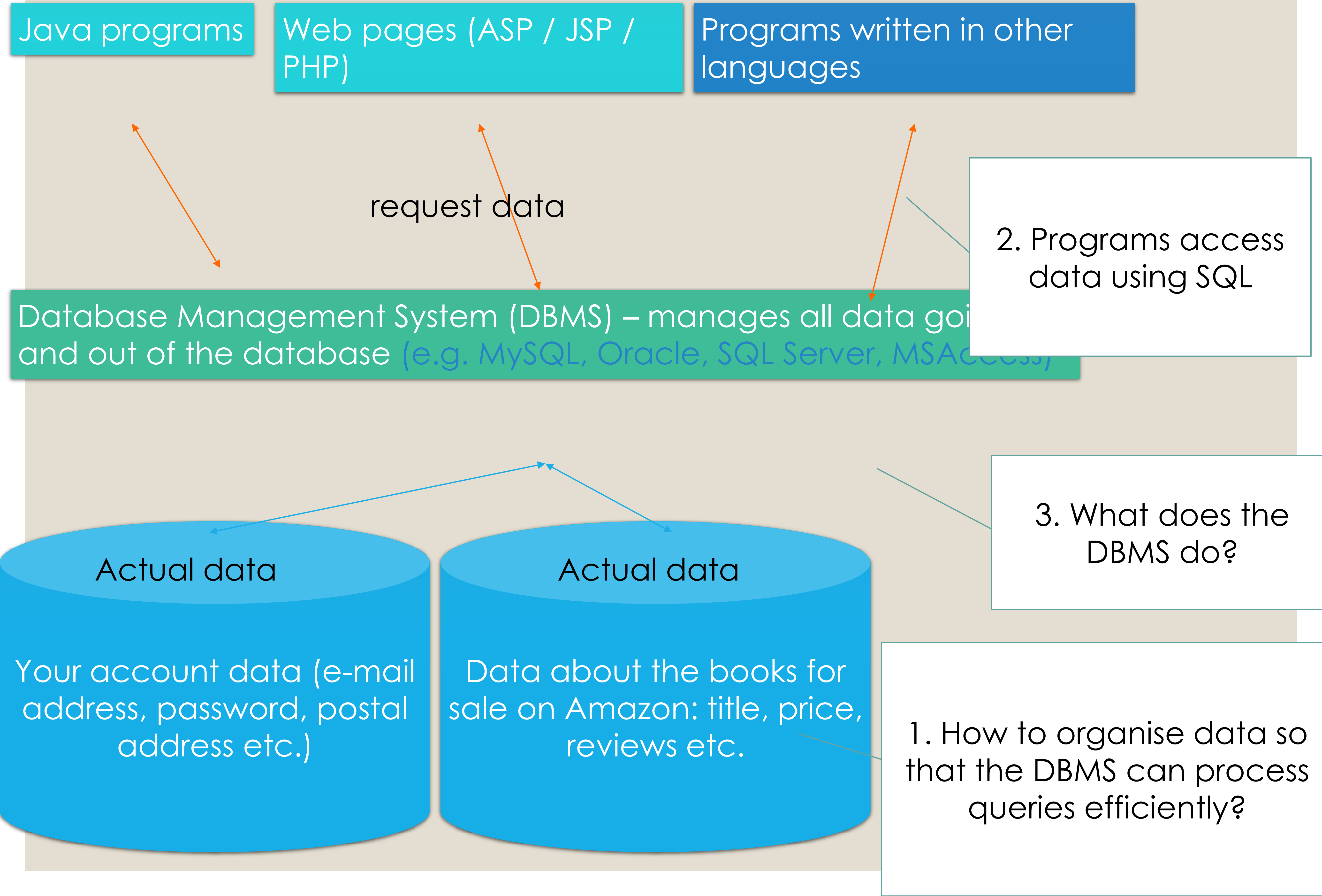
Course Code	Course Name
BN001	Certificate in Computer Engineering
BN002	Certificate in Computing
BN005	Certificate in Business Studies

More on terminology – a database management system (DBMS)

- A collection of programs that enables users to create and maintain a database.
  - Records the structure of the data in the databases (meta data)
  - Handles request from users and programs to:
    - Add data the the database
    - Delete data from the database
    - Update data in the database
    - Query the database (makes requests such as list all books sold by amozon in the last 30 minutes)

CRUD application – Acrostic for an application using a database. The letters stand for Create, Read, Update and Delete

The big picture . . .



Java programs

Web pages (ASP / JSP / PHP)

Programs written in other languages

request data

Database Management System (DBMS) – manages all data goi
and out of the database (e.g. MySQL, Oracle, SQL Server, MSAccess)

Actual data

Your account data (e-mail address, password, postal address etc.)

Actual data

Data about the books for sale on Amazon: title, price, reviews etc.

2. Programs access data using SQL

3. What does the DBMS do?

1. How to organise data so that the DBMS can process queries efficiently?

MySQL – the DBMS we used in the lab

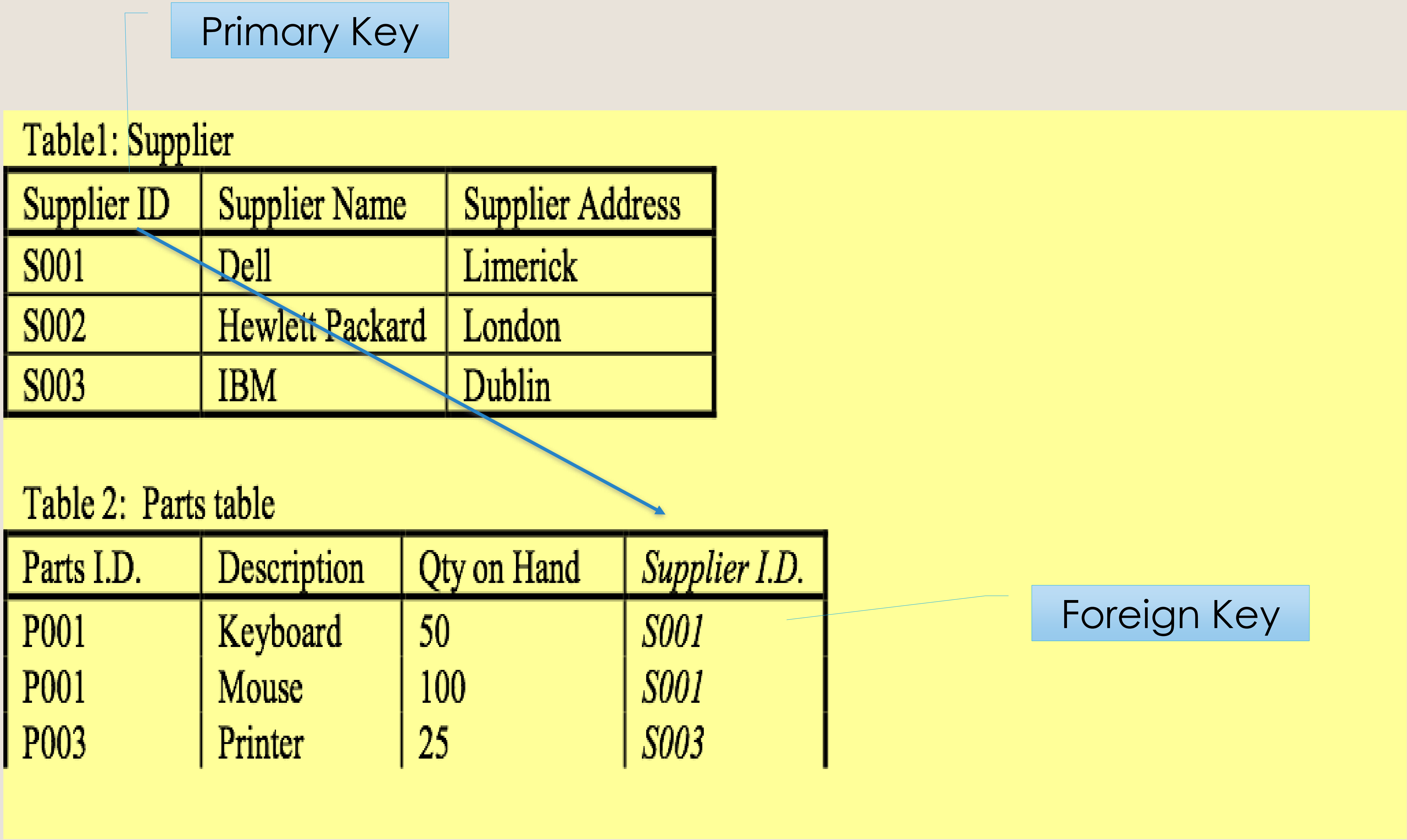
- MySQL is a Relational Database Management System (RDBMS).
- MySQL is the most popular Open Source database implementation
- MySQL Database Server is very fast, reliable, and easy to use.
- The MySQL Database Software is a client/server system.

Topics

Topic 2: SQL (Lectures 2 - 5)

Relational Database

Tables are linked by having common fields - **Primary Key / Foreign Key** links



Primary Key

Table1: Supplier

Supplier ID	Supplier Name	Supplier Address
S001	Dell	Limerick
S002	Hewlett Packard	London
S003	IBM	Dublin

Table 2: Parts table

Parts I.D.	Description	Qty on Hand	*Supplier I.D.*
P001	Keyboard	50	*S001*
P001	Mouse	100	*S001*
P003	Printer	25	*S003*

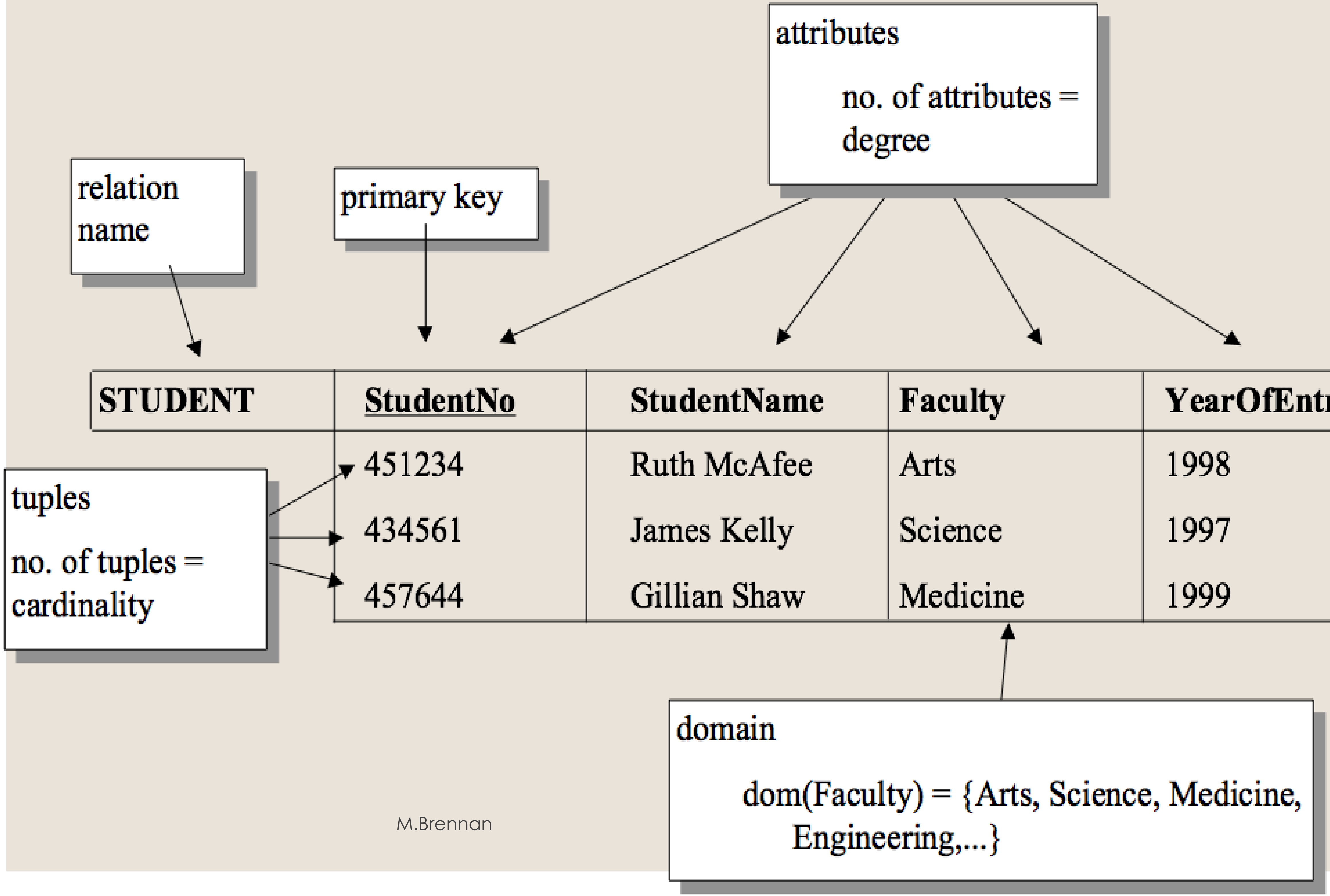
Foreign Key

The Relational DB Model

- **Degree of a Table**: number of columns

- **Cardinality of a Table:** number of rows

STUDENT table



attributes

no. of attributes = degree

relation name

primary key

STUDENT	**StudentNo**	**StudentName**	**Faculty**	**YearOfEntry**
	451234	Ruth McAfee	Arts	1998
	434561	James Kelly	Science	1997
	457644	Gillian Shaw	Medicine	1999

tuples

no. of tuples = cardinality

domain

dom(Faculty) = {Arts, Science, Medicine, Engineering,...}

Introduction to SQL

- Universal Language for
  - Creating Tables to hold the data (Data Definition Language – DDL: 6 commands)
  - Data Manipulation & Retrieval (Data Manipulation Language - DML: 8 commands)
  - Data Control – gives users permissions for the database (Data Control Language – DML: 3 commands)

- Note: While SQL is a standard language, Database vendors support slight variations of SQL. Variations occur in the data types supported, and the functions support (to be covered in a later lecture)

DDL (Data Definition Language) used to define the table structure and attributes of the database table

SQL commands:-
- CREATE TABLE specifies attributes and constraints for a table.
- DROP TABLE
- ALTER TABLE
- TRUNCATE etc.

DML (Data Manipulation Language) used to retrieve, insert, modify or delete information within the database.

SQL commands:- SELECT, UPDATE, INSERT, DELETE

DCL (Data Control Language) - used to manage DB security, i.e. assign access rights to users

SQL commands:– GRANT, DENY, REVOKE

Note the order of the clauses!!

```
SELECT columnlist
FROM tablename
WHERE condition
GROUP BY
HAVING group condition
ORDER BY            ;
```

SQL Statement Processing Order

- SELECT – identifies the **columns** to be displayed
- FROM – identifies the **table**(s) involved
- WHERE – Finds **rows** meeting a stated condition
- GROUP BY –Identifies groups to which a **group functionis** to be applied (max, min, avg, sum etc.)
- HAVING – Finds all **groups** meeting a stated conditions
- ORDER BY – **order** in which results are to be displayed

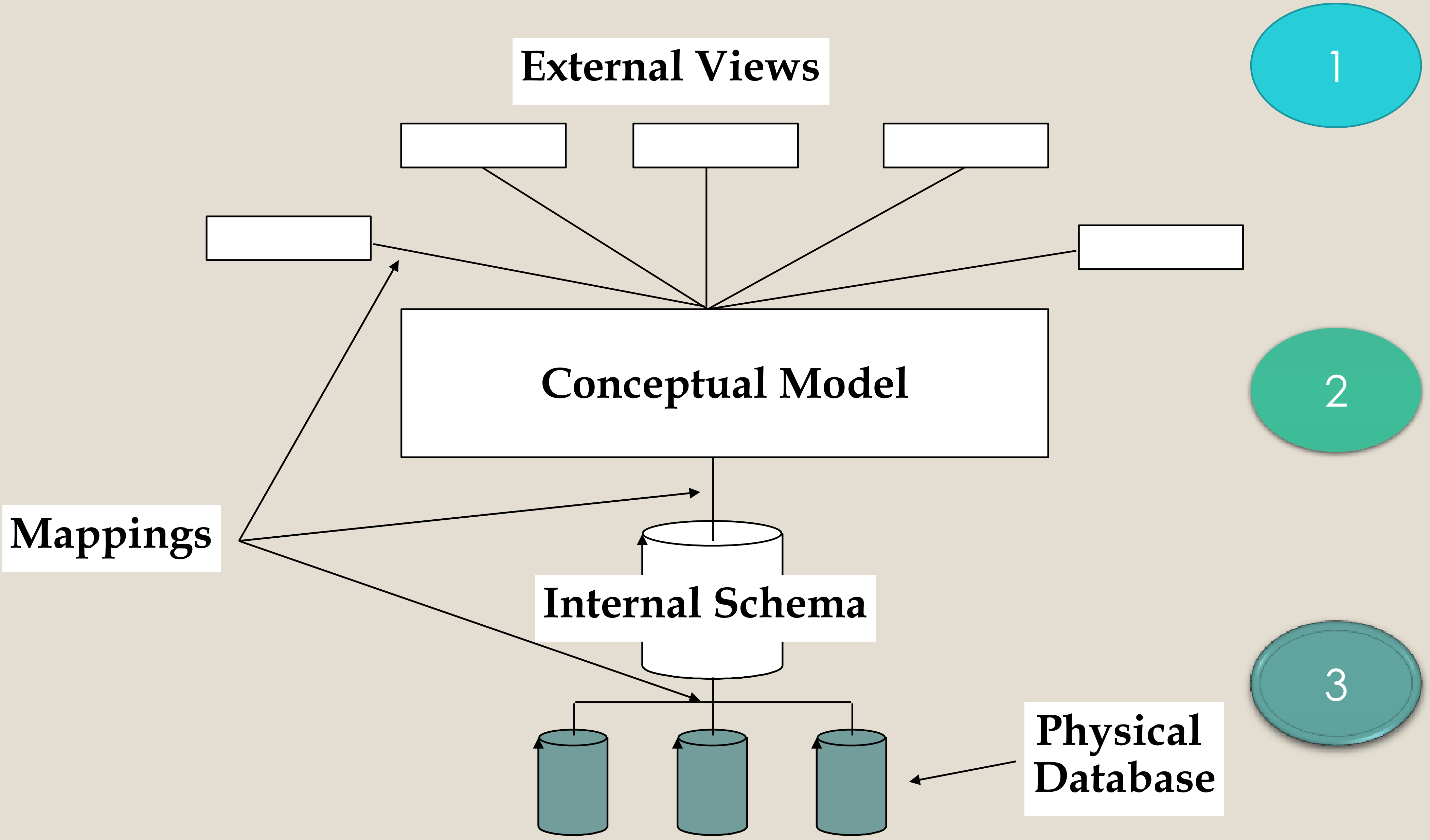
Topics

Topic 3: Database Design (lecture 6 - 9)

(ERDs, Relational Model and Normalisation)

- ANSI/SPARC Architecture
- ERD - Entities, Relationships, Cardinality, Participation, attributes
- Relational Model - FK to model relationships
- Normalisation - 3rd Normal form, well-structured tables

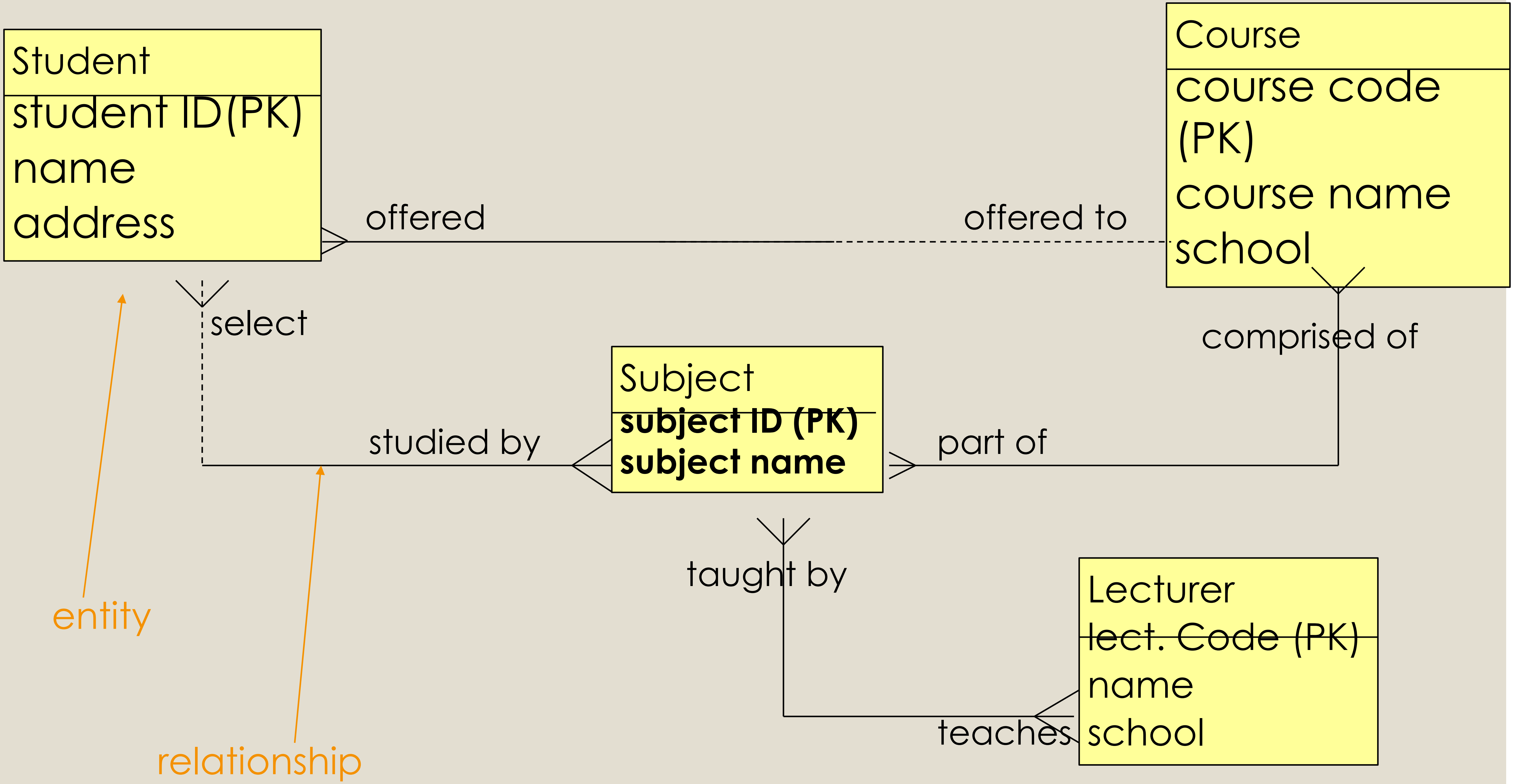
3 levels of a DBMS - *ANSI/SPARC Architecture*



Entity Relationship Diagrams

- The initial model for a database is a conceptual model to determine what **entities** a system needs to store data about.
- The most common conceptual model used is an **Entity Relationship Diagram**.
- It models Entities, and the relationships between entities.

ERD – elements/features

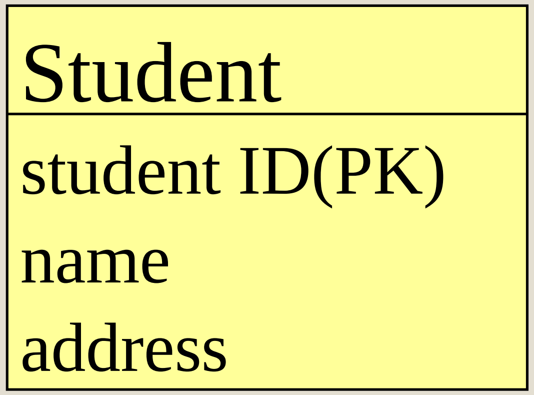


Stage 2: Relational Model

The next stage in the database table design is to convert the Entity Relationship Diagram into a Relational Model.

Producing a Relational Model: Representing entities as relations

Each entity is written as follows:



Student

student ID(PK)

name

address

becomes:

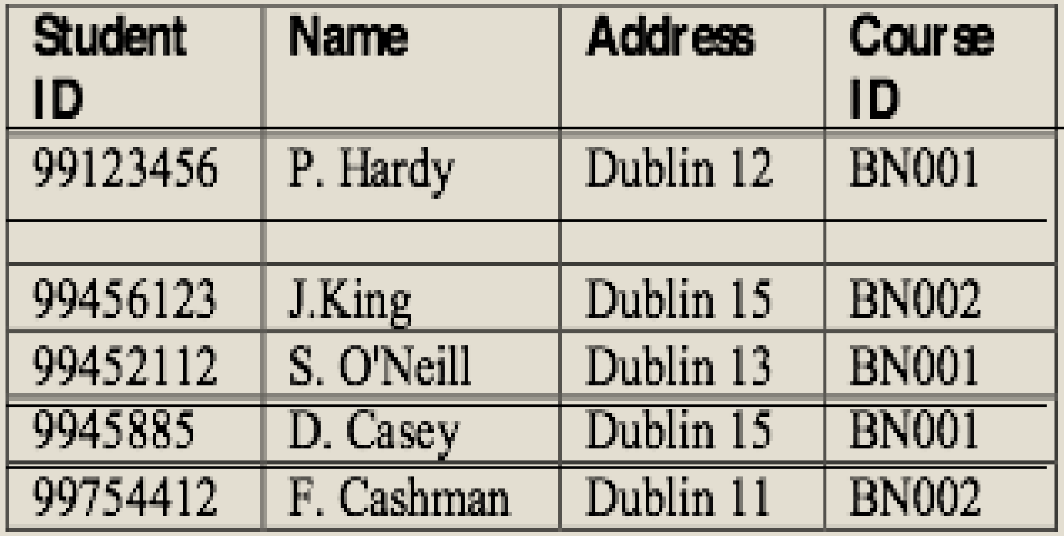
Student (Student ID(PK), name, address)

Producing a Relational Model: Foreign Keys

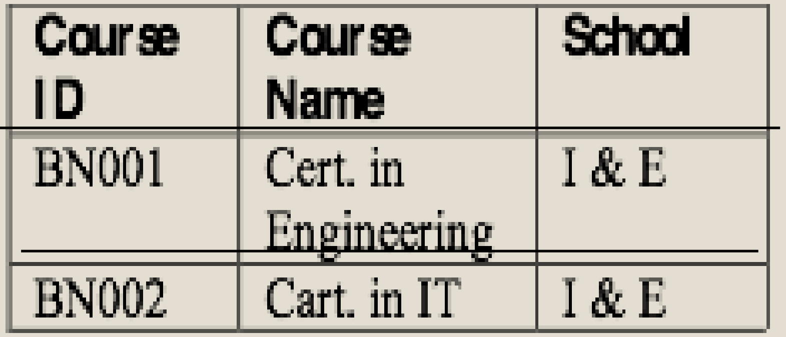
- A foreign key is an attribute in a table which is the primary key of another table

- e.g. CourseID in the student table below is a foreign key

Student (Student ID(PK), name, address, Course ID(FK))
Course (Course ID(PK), Course name, School)



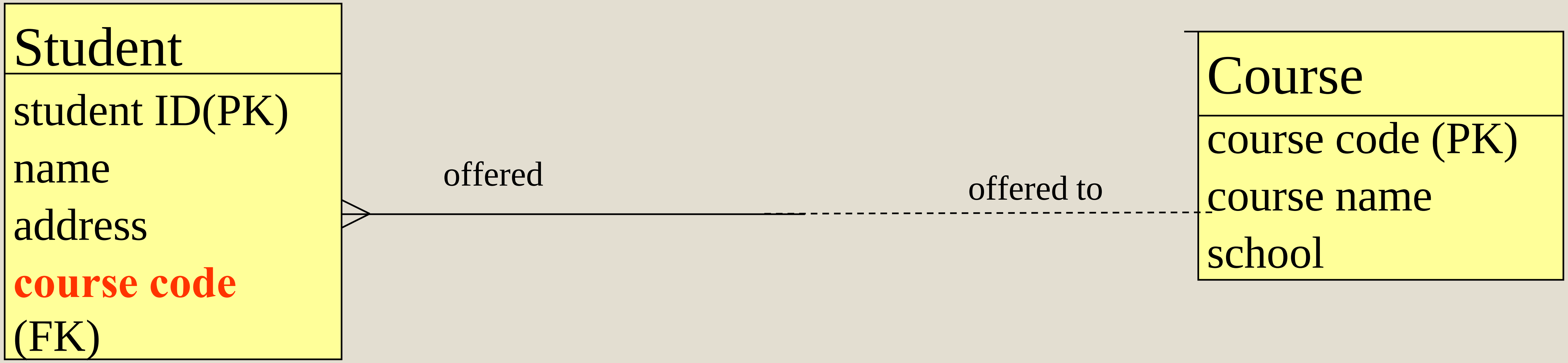
Student ID	Name	Address	Course ID
99123456	P. Hardy	Dublin 12	BN001
99456123	J.King	Dublin 15	BN002
99452112	S. O'Neill	Dublin 13	BN001
9945885	D. Casey	Dublin 15	BN001
99754412	F. Cashman	Dublin 11	BN002



Course ID	Course Name	School
BN001	Cert. in Engineering	I & E
BN002	Cart. in IT	I & E

Producing a Relational Model/ Table: Representing relationships as foreign keys 1:M

- Add the Primary Key from the One (1) side of the relation to attributes of the Many side.
- It is known as Foreign key on the Many (m) side.
- Foreign key acts as a Link between the entities



Student

student ID(PK)

name

address

course code

(FK)

offered

offered to

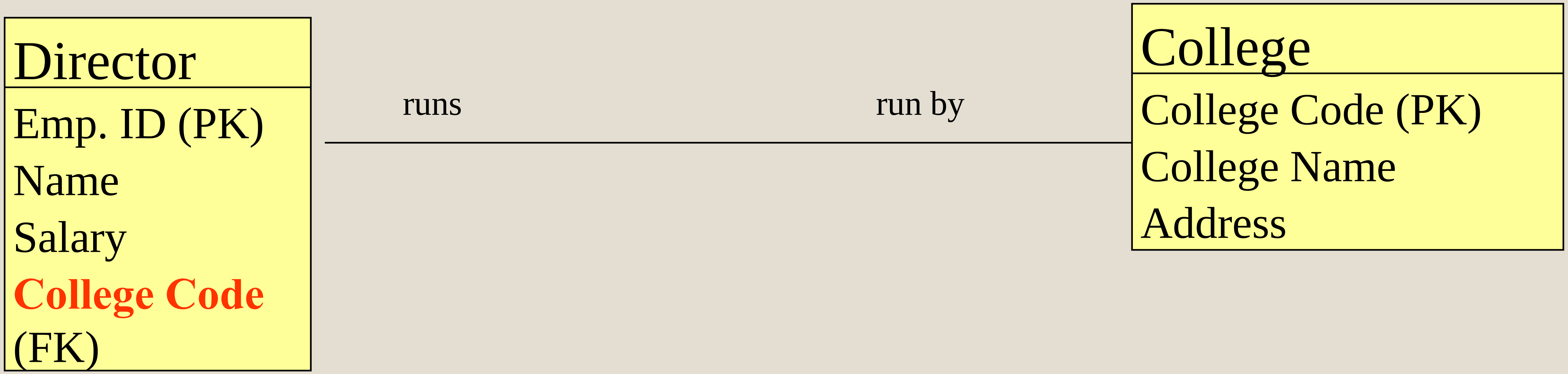
Course

course code (PK)

course name

school

Producing a Relational Model/ Table: Representing relationships as foreign keys 1:1



Director

Emp. ID (PK)

Name

Salary

College Code

(FK)

runs

run by

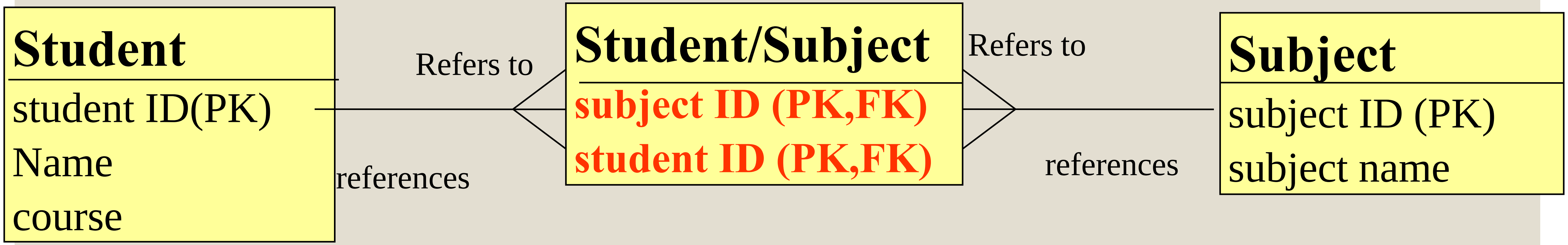
College

College Code (PK)

College Name

Address

Producing a Relational Model/ Table: Representing relationships as foreign keys M:N



Student

student ID(PK)

Name

course

Refers to

references

Student/Subject

subject ID (PK,FK)

student ID (PK,FK)

Refers to

references

Subject

subject ID (PK)

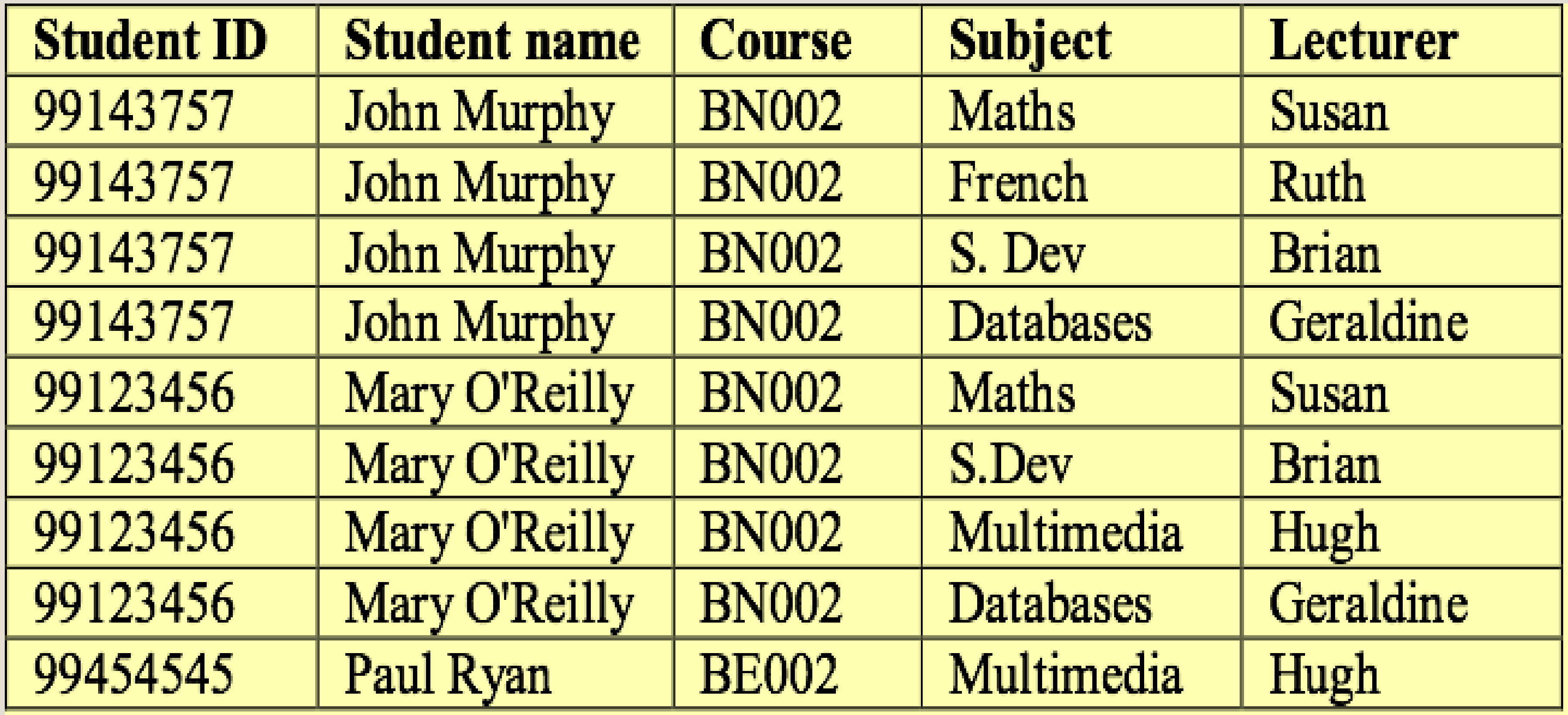
subject name

Student (student ID(PK), name, course)

Subject (subject ID(PK), subject name)

Student_Subject (student ID(PK, FK), subject ID(PK, FK))

Why avoid duplicate data? To avoid the following three anomalies . . .



Student ID	Student name	Course	Subject	Lecturer
99143757	John Murphy	BN002	Maths	Susan
99143757	John Murphy	BN002	French	Ruth
99143757	John Murphy	BN002	S. Dev	Brian
99143757	John Murphy	BN002	Databases	Geraldine
99123456	Mary O'Reilly	BN002	Maths	Susan
99123456	Mary O'Reilly	BN002	S.Dev	Brian
99123456	Mary O'Reilly	BN002	Multimedia	Hugh
99123456	Mary O'Reilly	BN002	Databases	Geraldine
99454545	Paul Ryan	BE002	Multimedia	Hugh

POOR DATA DESIGN

- Update anomaly – suppose the maths lecturer changes from Susan to Colm. How many places would you need to make the change?
- Delete anomaly – if John Murphy leaves the course, there will be no record of who teaches French
- Insertion anomaly – Suppose you want to add a new subject called "modelling and database design", but there is no student registered for the subject yet. How do you add it the the table above?

Well Structured Relations

- Once the relational model is created, the final stage in database design is to **NORMALISE** the data, also called producing a well structured relation.

- A relation is well-structured if all the attributes in the relation are functionally dependent on the primary key.
  - i.e. the attribute has one unique value that can be determined from the primary key.

Example 1

- Student (Student ID(PK), student name, lecturer name, course description)

- Does a student ID identify a specific student's name?
- Does a student ID identify a specific lecturer's name?
- Does a student ID identify a specific course description?

Only student name is functionally dependent on Student ID. The other attributes are in the wrong table.

Example 2

- Student_Subject(Student ID(PK), subject ID(FK), grade, student name, subject name)

- Do you need <u>both</u> the student ID and the subject ID to find the grade a student got in a particular subject?
- Do you need <u>both</u> the student ID and the subject ID to get a student's name?
- Do you need <u>both</u> the student ID and the subject ID to get the subject's name?

Only grade is functionally dependent on Student ID **<u>AND</u>** subject ID. The other attributes are in the wrong table.

Example 3

Student ID(PK),	student name,	subject name,	grade
99123456	Kelly	Databases	C
99123456	Kelly	Software Dev	B
99123456	Kelly	Networking	C+

- Does a student ID identify a specific student’s name?
- Does a student ID identify a specific subject’s name?
- Does a student ID identify a specific grade?

Only student name is functionally dependent on Student ID. The other attributes are in the wrong table.

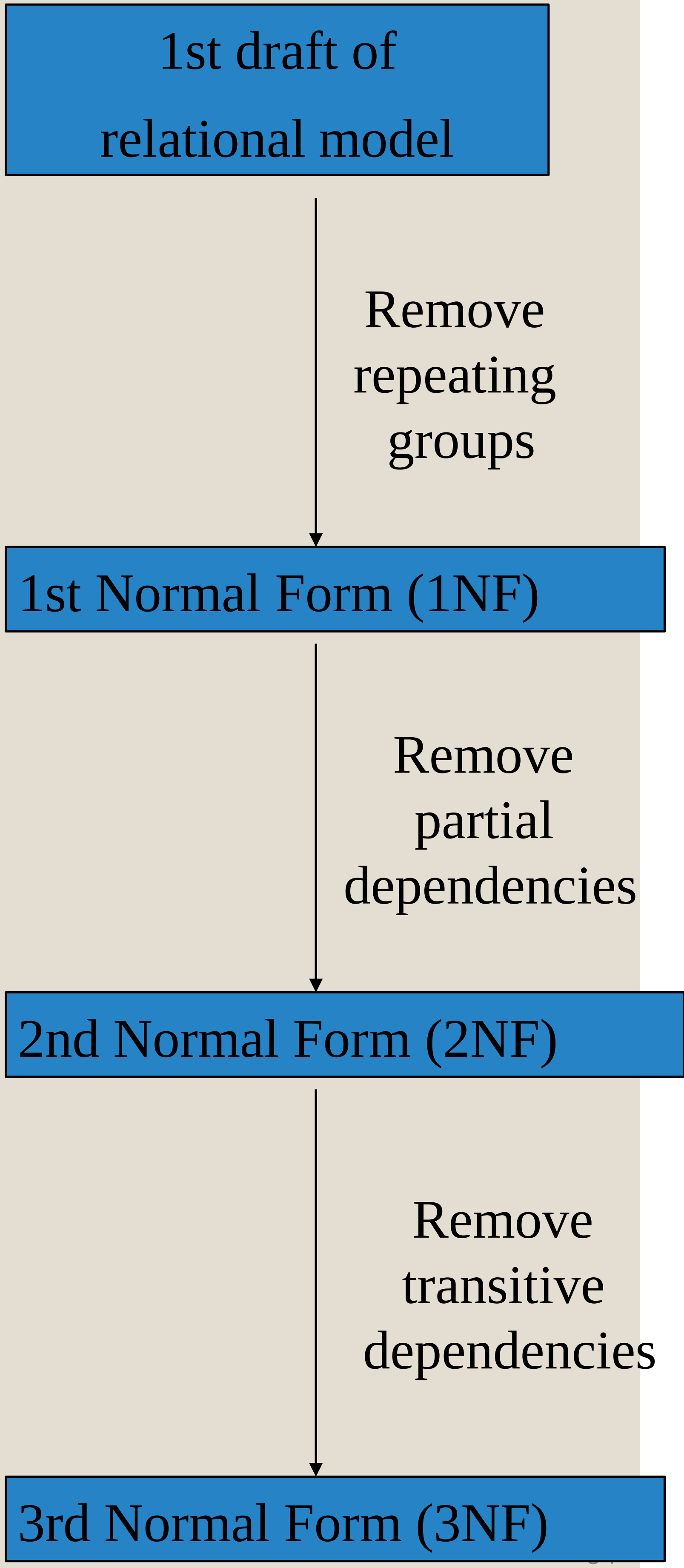
Recap – Functionally dependent means the attribute has one unique value that can be determined from the key field.

Normalisation

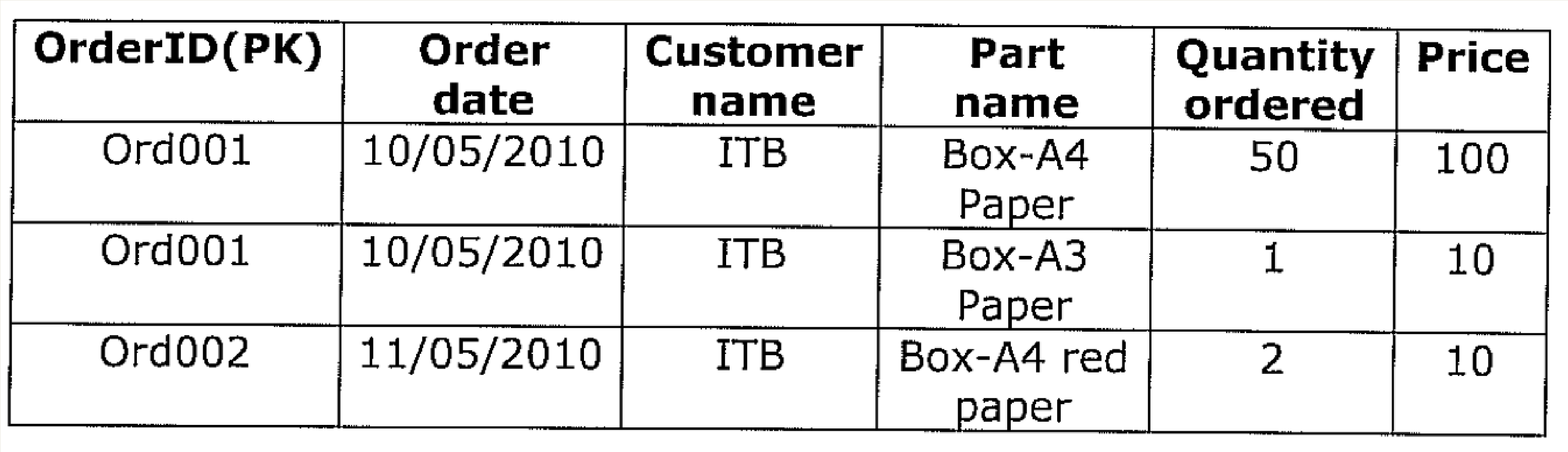
- There are three ways in which an attribute is NOT functionally dependent on the primary key, as illustrated in the three examples done previously.
- Identifying these scenarios is done by following the three steps of Normalization:

1. Bring to 1st normal form – remove repeating groups, i.e Example 3 above.
2. Bring to 2nd normal form – remove partial dependencies, i.e. Example 2 above
3. Bring to 3rd normal form – remove transitive dependencies, i.e. Example 1 above

- Once in third normal form (3NF), the tables are well-structured


1st draft of relational model
Remove repeating groups
1st Normal Form (1NF)
Remove partial dependencies
2nd Normal Form (2NF)
Remove transitive dependencies
3rd Normal Form (3NF)

Normalisation



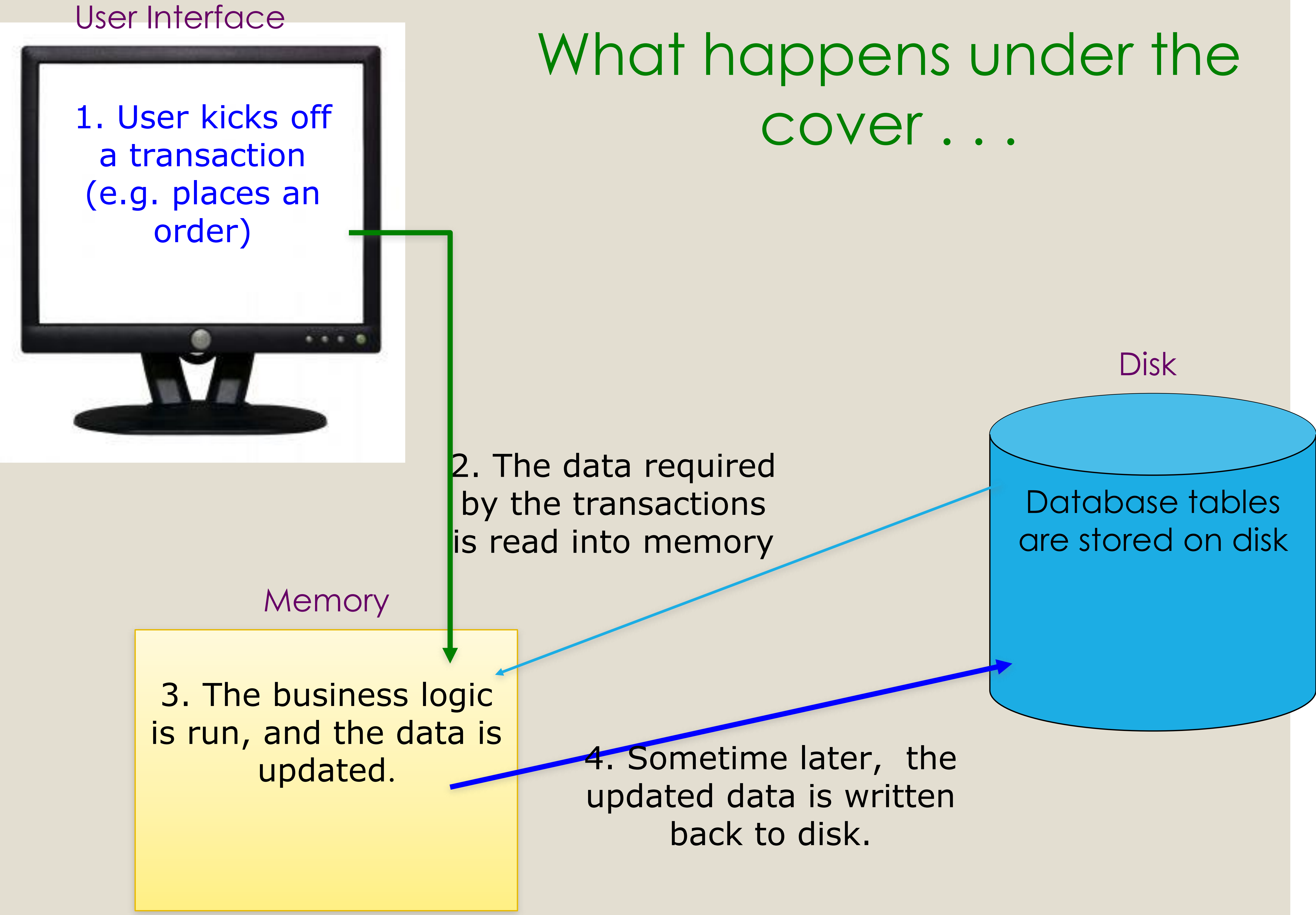
OrderID(PK)	Order date	Customer name	Part name	Quantity ordered	Price
Ord001	10/05/2010	ITB	Box-A4 Paper	50	100
Ord001	10/05/2010	ITB	Box-A3 Paper	1	10
Ord002	11/05/2010	ITB	Box-A4 red paper	2	10

Topics

Topic 4: Transaction Processing, Security

- Database Transactions
- Commit & Rollback
- ACID Properties
- Concurrency Problems
- Locking, two-phase locking, deadlock
- Recovery

What happens under the cover . . .



User Interface

1. User kicks off a transaction (e.g. places an order)

2. The data required by the transactions is read into memory

Memory

3. The business logic is run, and the data is updated.

Disk

Database tables are stored on disk

4. Sometime later, the updated data is written back to disk.

Revision Questions:

- What is a transaction?
- Describe the ACID properties?
- What are the 3 classical concurrency control problems?
- What is meant by the terms: lost update, uncommitted dependency problem, and the inconsistent analysis problem?
- Describe each of the problems briefly.
- What is meant by the term 'granularity of locking'?
- Provide 3 examples of different levels of granularity in locking.
- What is deadlock?
- How can deadlock be dealt with in a DBMS?
- What are the 2 phases in 2-phase locking?
- What is the difference between system failure and media failure.
- Give 5 examples of system failures.
- What is a Checkpoint?
- How does the Recovery Manager recover transactions when a system fails?
- Describe the entries in the Transaction Log?
- How does a database recover from media failure?