Database Fundamentals

INTRODUCTION TO DATABASES

Database Overview

- What is a database?
  - A handful of tables containing data that are used to
  - A collection of potentially useful data that models r **objects** and their **relationships**
    - Objects/Entities like People, items, Purchases, Students etc
    - Relationships like Jim purchased a book
    - Conceptual like songs, movies
  - It is managed by a Database Management System (DBMS)
    - This system may be manual or computerised
    - Card Catalogue vs modern Computerised equivalent
  - For us, the DBMS is software designed to store and manage data across one or more databases

A typical example of a database is a Library

- Books = objects containing data
- Catalogue system = DBMS

Database Overview

Problems with **Traditional Physical** Data Storage

- Limited by Physical size
  - MS SQL will store ~2 trillion records in a fraction of the space!

- Limited Multi-access
  - A borrowed book can be viewed by only one person

- Limited search methods (Title, Author, Subject)
  - What if the data has 2 Authors?
  - What if the title starts with “The” or “As” (eg “As Time Goes By”)
  - What if I want to find a word in the Title?
  - What if you don’t want to search by title 1st word, Author or Subject?

- Slow turn-around
  - Would you wait 7 minutes?
  - Google can pull up a book in < 7 seconds

Problems with **Traditional Physical** Data Storage

- Problems with traditional **Physical** methods
  - Multiple indexes that must be rigorously maintained
  - Anomalies: - What IFs
    - What if a card is misplaced?
    - An item is deleted in one index but not the others (Deletion Anomaly)
    - An item is entered into one index but not the others (Insertion Anomaly)
    - An item was incorrectly entered in all indexes but only corrected in one? (Update Anomaly!)
    - What if a catalogue card is misplaced? What if there are different versions of the same record?
  - Complex Query Limitations:
    - How quickly can you count the number of books on a topic or by an author?
    - What if a book is on many subjects – it must appear multiple times in the subject catalogue (prone to above anomalies)

Modern Data Storage

- Data can be stored in files such as text files, Excel sheets, Word documents.
  - **Advantages**
    - low cost (is this really true? – think time and effort!)
    - No extra software/hardware is needed (or open source, standard Mac/PC), easy to use?

	A	B	C	D	E	F	G	H
1	Title	FirstName	LastName	Suburb	State	PostalCode	AddressLine1	AddressLine2
2		Devin	Adams	Cranbourne	Victoria	3977	8590 High Maple Court	Cranbourne, VIC 3977
3		Isabella	Adams	Townsville	Queensland	4810	1933 Rock Creek Pl.	Townsville, QLD 4810
4		Sydney	Adams	Hervey Bay	Queensland	4655	8128 Kane Circle	Hervey Bay, QLD 4655
5		Amber	Adams	Brisbane	Queensland	4000	9720 Morning Glory Dr.	Brisbane, QLD 4000
6		Savannah	Adams	Sunbury	Victoria	3429	8825 Walters Way	Sunbury, VIC 3429
17954		Sandra	Zhu	St. Leonards	New South Wales	2065	7001 Lanitos Ct	St. Leonards, NSW 2065
17955		Alan	Zhu	Darlinghurst	New South Wales	2010	8995 Stanford St.	Darlinghurst, NSW 2010
17956		Wesley	Zhu	Rhodes	New South Wales	2138	2553 Croyden Dr.	Rhodes, NSW 2138
17957		Jorge	Zhu	Townsville	Queensland	4810	3632 Ramsay Circle	Townsville, QLD 4810
17958		Krystal	Zimmerman	Bendigo	Victoria	3550	515 Bayview Ct.	Bendigo, VIC 3550
17959		Candice	Zimmerman	Melton	Victoria	3337	4643 Elkwood Dr.	Melton, VIC 3337
17960		Tiffany	Zimmerman	Geelong	Victoria	3220	4701 Mt. Dell Drive	Geelong, VIC 3220
17961		Jenny	Zimmerman	Darlinghurst	New South Wales	2010	546 Leonard Ct	Darlinghurst, NSW 2010
17962								

Problems with **Modern** Data Storage

- Data can be stored in files such as text files, Excel sheets, Word documents.
  - **Disadvantages**
    - Size - Retrieval of data is slow when the volume of data is large (but better than physical data)
    - Updates/Synchronisation – files are hard to share. Multi-user updates often end up with multiple versions of the same file or overwriting someone's changes!
    - Accuracy – Duplication is wasteful and can cause confusion (inconsistent data)
    - Security – restriction to some fields is more difficult (eg TFN, studentID + username)
    - Programming is required for automatic processing: different files require different programs for processing
      - **Inability to process complex queries** (at least with any efficiency)

Modern Data Storage

- Did you know data is still stored on Tape?
  - Tape is the most affordable form of data storage (costing < 10¢/GB)
  - Tape cartridges can store up to 6.25TB on a single cartridge
    - More than 6 Peta Bytes (6 x 1000TB) can be stored every 1.3m^2 in a data centre

  - Has lower error rates that HDD and less prone to viruses
  - Lasts longer

  - But ~90% of business data is rarely accessed again! - so why bother?
    - Compliance with laws and company regulations
    - Data Mining – maybe the data holds some hidden detail?

Why Store Data?

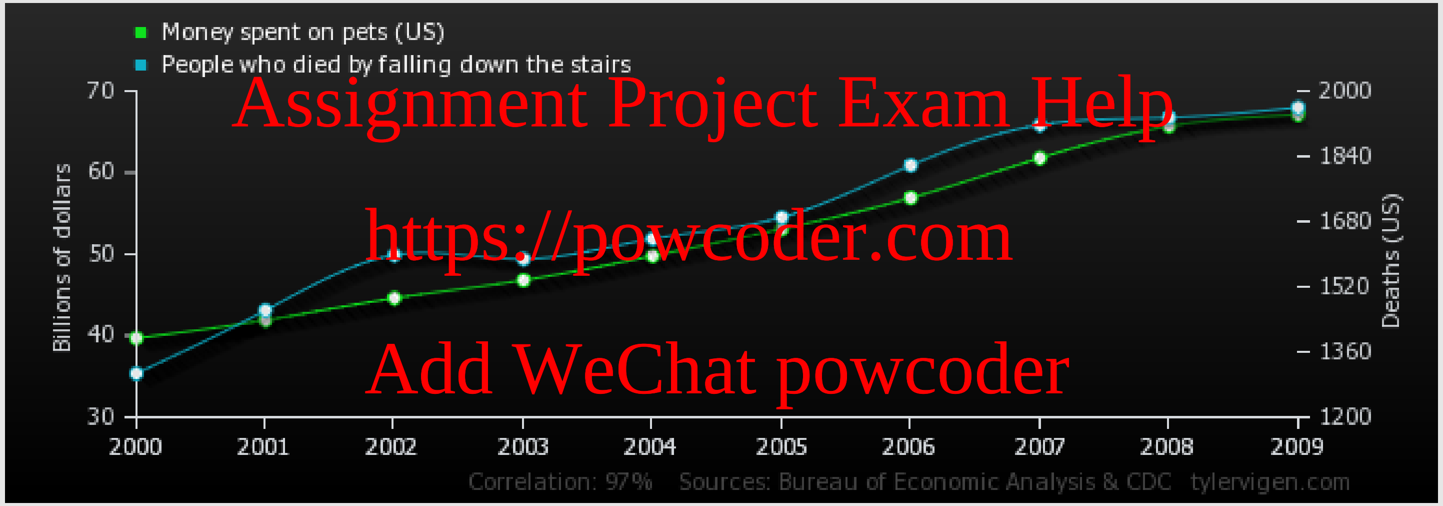
- So we can extract facts (information) and use it for decision making
  - Car airbags kill 1 person for every 22 lives that they save
  - A new baby usually deprives each of it's parents around 350-400 hours of sleep in the first year
  - On average, 100 people choke to death on ballpoint pens every year
- Annoy friends and family with facts inferred from data

Why Store Data?

Money spent on pets (US)

correlates with

People who died by falling down the stairs



Upload this chart to imgur

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Money spent on pets (US) *Billions of dollars (Bureau of Economic Analysis)*	39.7	41.9	44.6	46.8	49.8	53.1	56.9	61.8	65.7	67.1
People who died by falling down the stairs *Deaths (US) (CDC)*	1,307	1,462	1,598	1,588	1,638	1,690	1,818	1,917	1,935	1,960

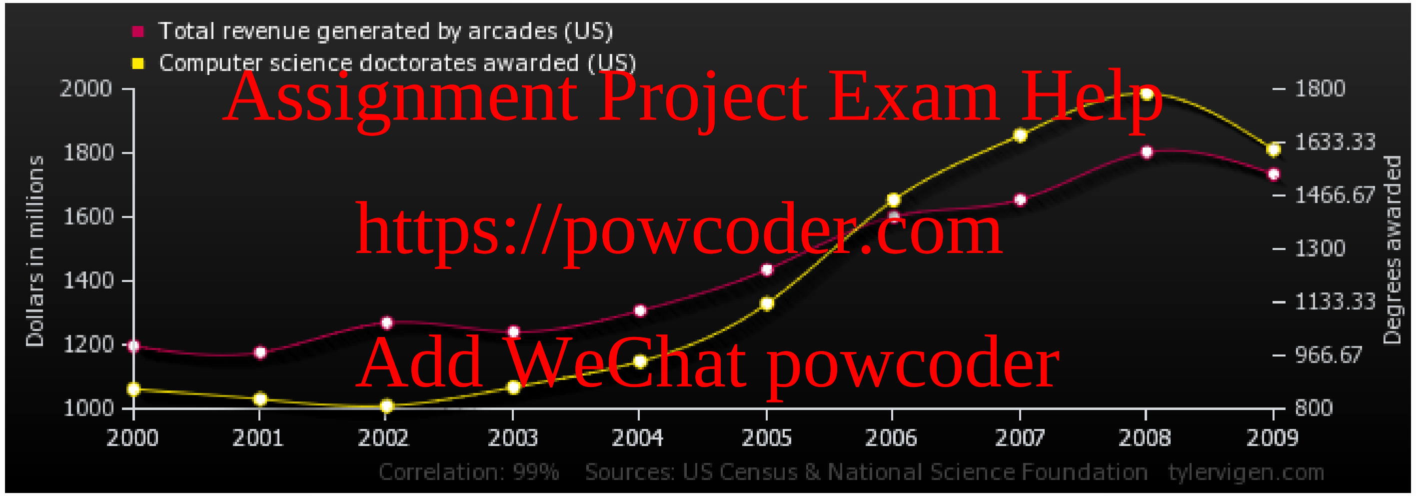
Correlation: 0.971791

Why Store Data?

Total revenue generated by arcades (US)

correlates with

Computer science doctorates awarded (US)



Upload this chart to imgur

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Total revenue generated by arcades (US) *Dollars in millions (US Census)*	1,196	1,176	1,269	1,240	1,307	1,435	1,601	1,654	1,803	1,734
Computer science doctorates awarded (US) *Degrees awarded (National Science Foundation)*	861	830	809	867	948	1,129	1,453	1,656	1,787	1,611

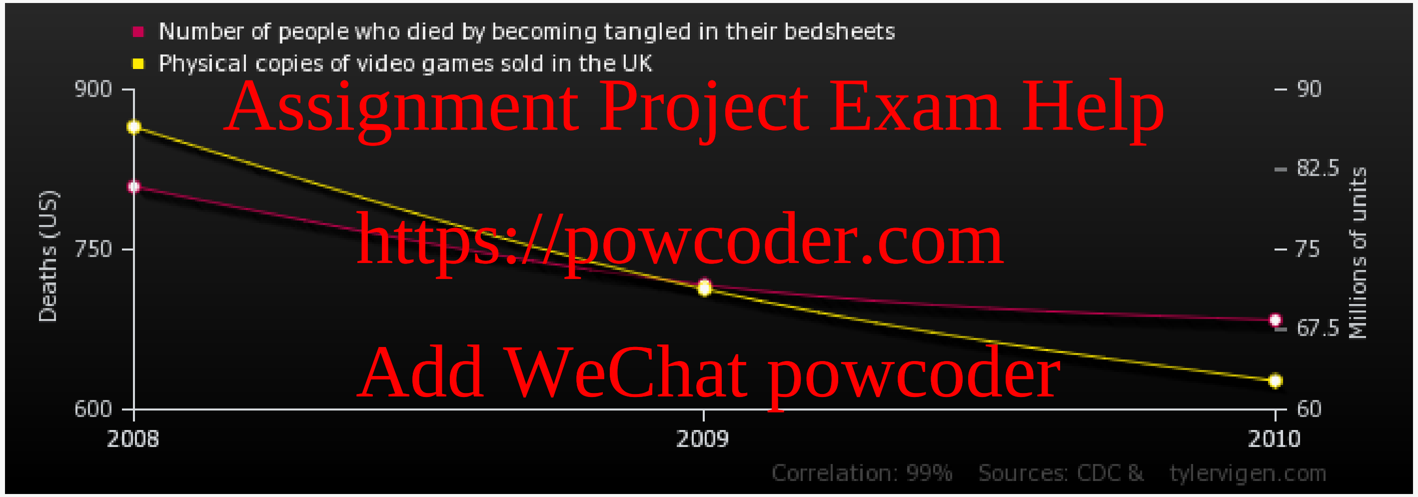
Correlation: 0.985065

Why Store Data?

Number of people who died by becoming tangled in their bedsheets

correlates with

Physical copies of video games sold in the UK



Upload this chart to imgur

	2008	2009	2010
Number of people who died by becoming tangled in their bedsheets *Deaths (US) (CDC)*	809	717	684
Physical copies of video games sold in the UK *Millions of units ()*	86.5	71.3	62.7

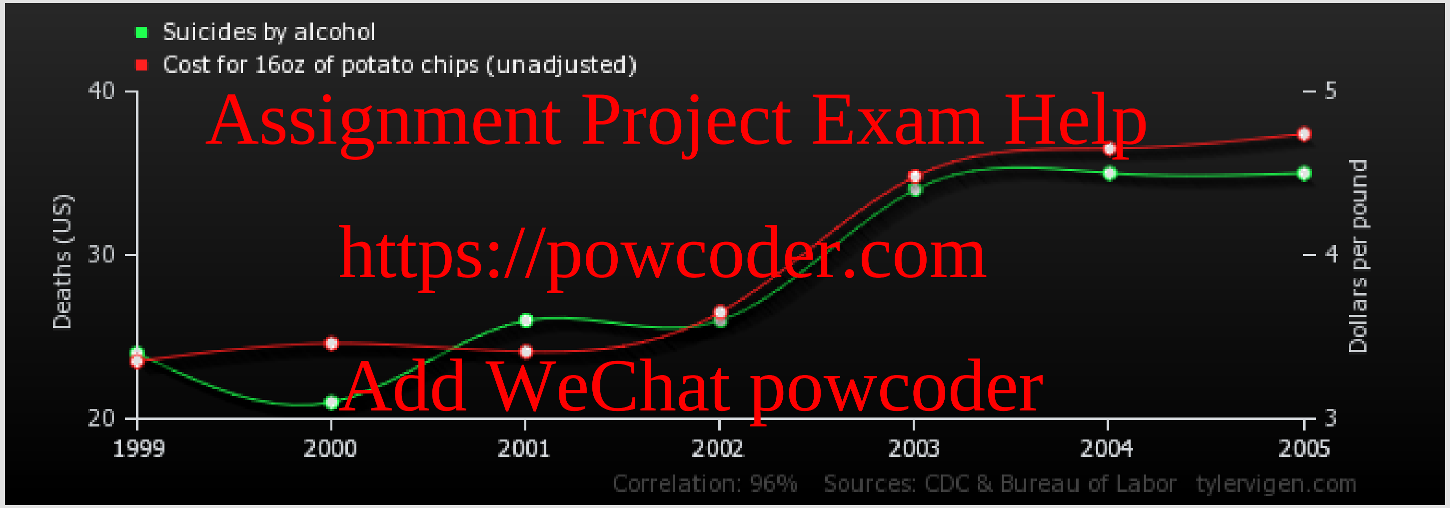
Correlation: 0.99425

Why Store Data?

Death by alcohol

correlates with

Cost for 16oz of potato chips (unadjusted)



Upload this chart to imgur

	1999	2000	2001	2002	2003	2004	2005
Suicides by alcohol *Deaths (US) (CDC)*	24	21	26	26	34	35	35
Cost for 16oz of potato chips (unadjusted) *Dollars per pound (Bureau of Labor)*	3.35	3.46	3.41	3.65	4.48	4.65	4.74

Correlation: 0.957817

Database Overview

- Why store data and not information?
  - Data is simple **re-usable** facts
    - Is non-specific and does not carry any meaning
    - Can be processed, organised, analysed to give it meaning (to provide information)
      - This makes data a valuable resource with a very long life cycle
      - Data + Context = Information

- Different questions of the same data will provide different information
  - This makes data more flexible
  - Data can be reused!

- What can you find out?
  - Data = the future
  - Data management = increased productivity
  - Better customer relations (sometimes!)
  - Knowing how to manage and use data = $$

Do **YOU** own a store card?

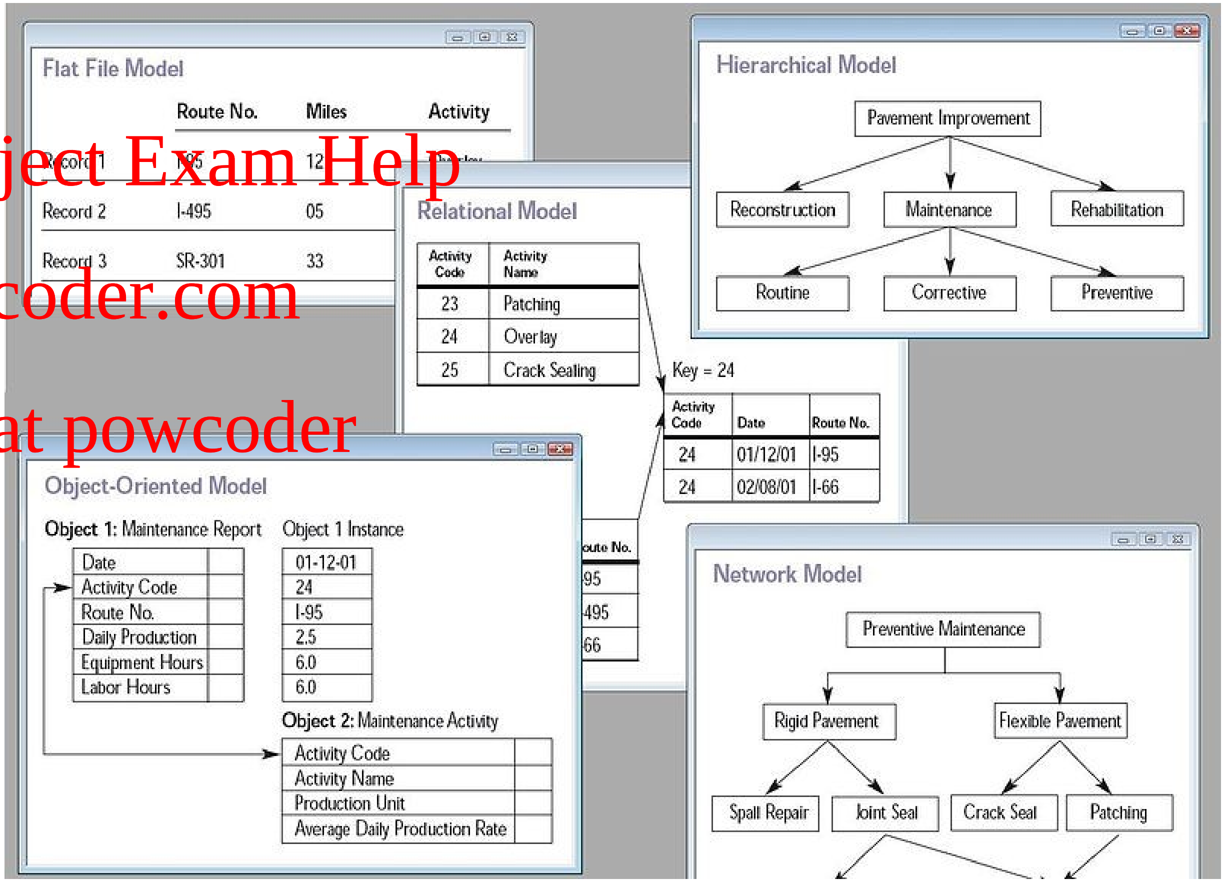
Target broke through to a new level of customer tracking with the help of statistical genius Andrew Pole, according to a New York Times Magazine cover story by Charles Duhigg.

Pole identified 25 products that when purchased together indicate a women is likely pregnant. The value of this information was that Target could send coupons to the pregnant woman at an expensive and habit-forming period of her life.

Plugged into Target's customer tracking technology, Pole's formula was a beast. Once it even exposed a teen girl's pregnancy:

Databases - The Data Model

- Data Model
  - The data model dictates how the data will be structured and stored
    - The **Relational Model** (most common)
    - Introduced in the 70s
    - **Structured data** stored in **tables** with **relationships**
    - Good for up to 100TB (scales up)
  - Alternatives exist and these will be covered in Enterprise Databases next year
    - Objects, XML, Hierarchical, Network
    - **NoSQL**/key-value and others
      - Good for 100TB+ (scales out)
      - Semi-structured data



Source: wikipedia.org

Database Overview

- How do we ask questions (query) of our database?
  - Need a Structured Query Language to Ask Questions of Structured Data.
    - **Structured Query Language** (SQL)
    - Remember this you will use it **forever**!!

  - A Structured language uses **keywords** in a **logical** pattern to perform tasks
    - It's easier to understand than other programming languages

  - A **DBMS** (database management system) interprets **SQL** and actions it
    - Determines what needs to be done, how and then returns the result

Databases in Computers – The DBMS

- A **D**ata**B**ase **M**anagement **S**ystem performs all the maintenance and management functions for the data model:
  - Managing disk space (shrink/expansion)
  - Communic ... ce versa)
  - Managing t
  - Managing t ... cluding performing
  - A **DBMS** Pe ... ng) of records
    - With greater efficiency and accuracy than us!

The DBMS and its users communicate using a *mutually understandable language*: **SQL** (**S**tructured **Q**uery **L**anguage!)

Popular free **DBMS** include:

- SQLite
- MySQL
- MS-SQL Express
- PostgreSQL

- Commercial: MS-SQL and Oracle, DB2 + the rest

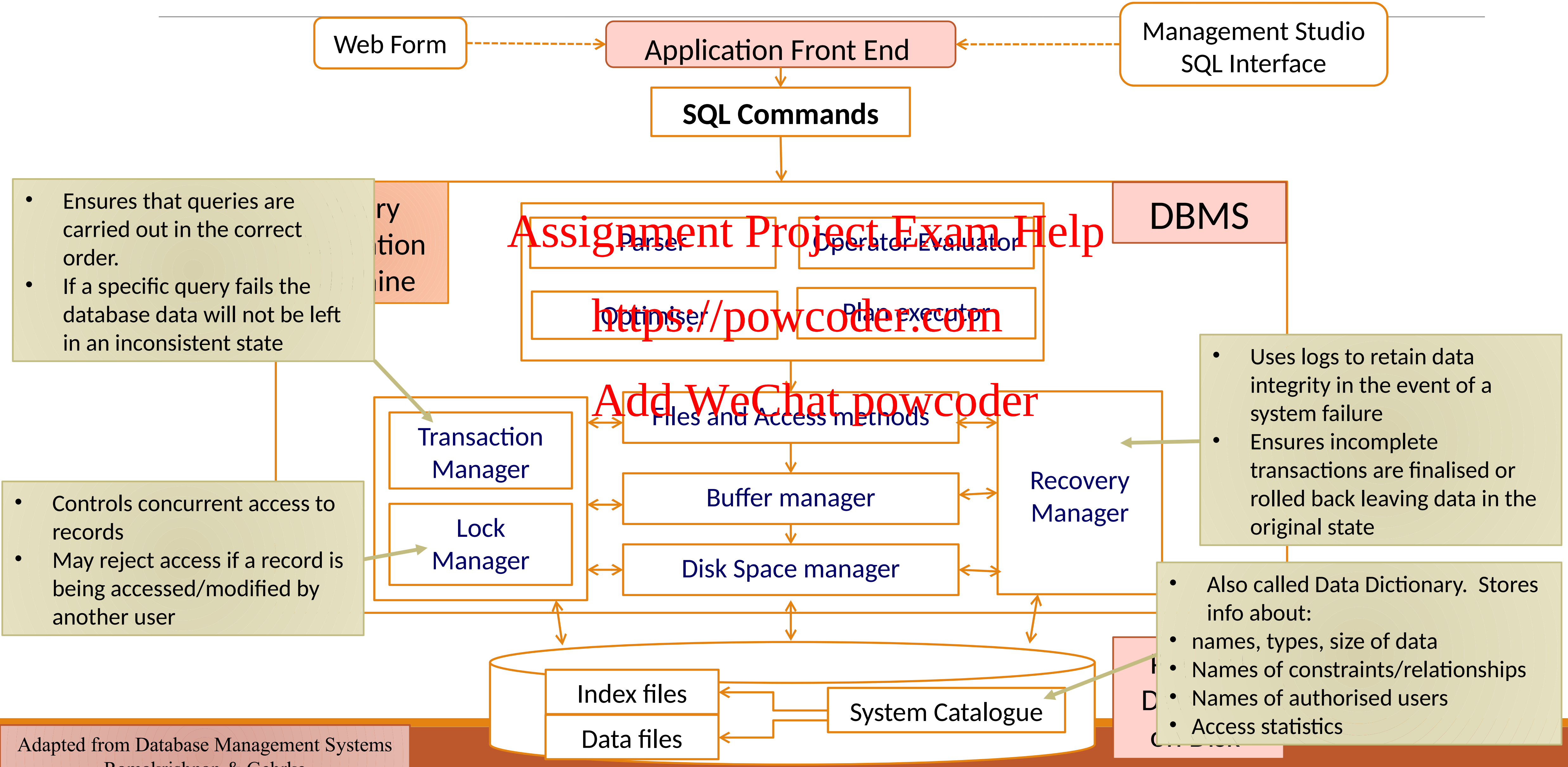
Databases in Computers – DBMS Properties

- A DBMS is an application that manages collections of data:
  - It can handle a large amount of data (more than RAM can hold)
  - It can share the data between various applications and users
    - Web interfaces, desktop/console applications incl. command line via the DBMS!

- A DBMS ensures
  - Data is **persistent** (it can be retrieved once it is created)
  - The data is **reliable** (it can be retrieved after hardware/software failures)
    - It can also ensure data meets specific requirements ("constraints": eg the student mark is between 0 and 100)
  - Privacy is maintained (it controls access and manipulation)

- A DBMS must also
  - Be **Efficient** (using the appropriate amount of resources, not flooding CPU/RAM)
    - Preferably no 7 min queries please!

Structure of a DBMS



Why Study Databases?

- Gain knowledge and skills to interact with a DBMS
  - Example: working in MS-SQL/Oracle
  - Migrating existing data from legacy systems ($$ data is money! $$)

- Gain skills for using a database driven application
  - Data insertion and deletion with or without GUI
  - Data queries with or without GUI

- Why things work the way they do (or fail to perform up to expectations)

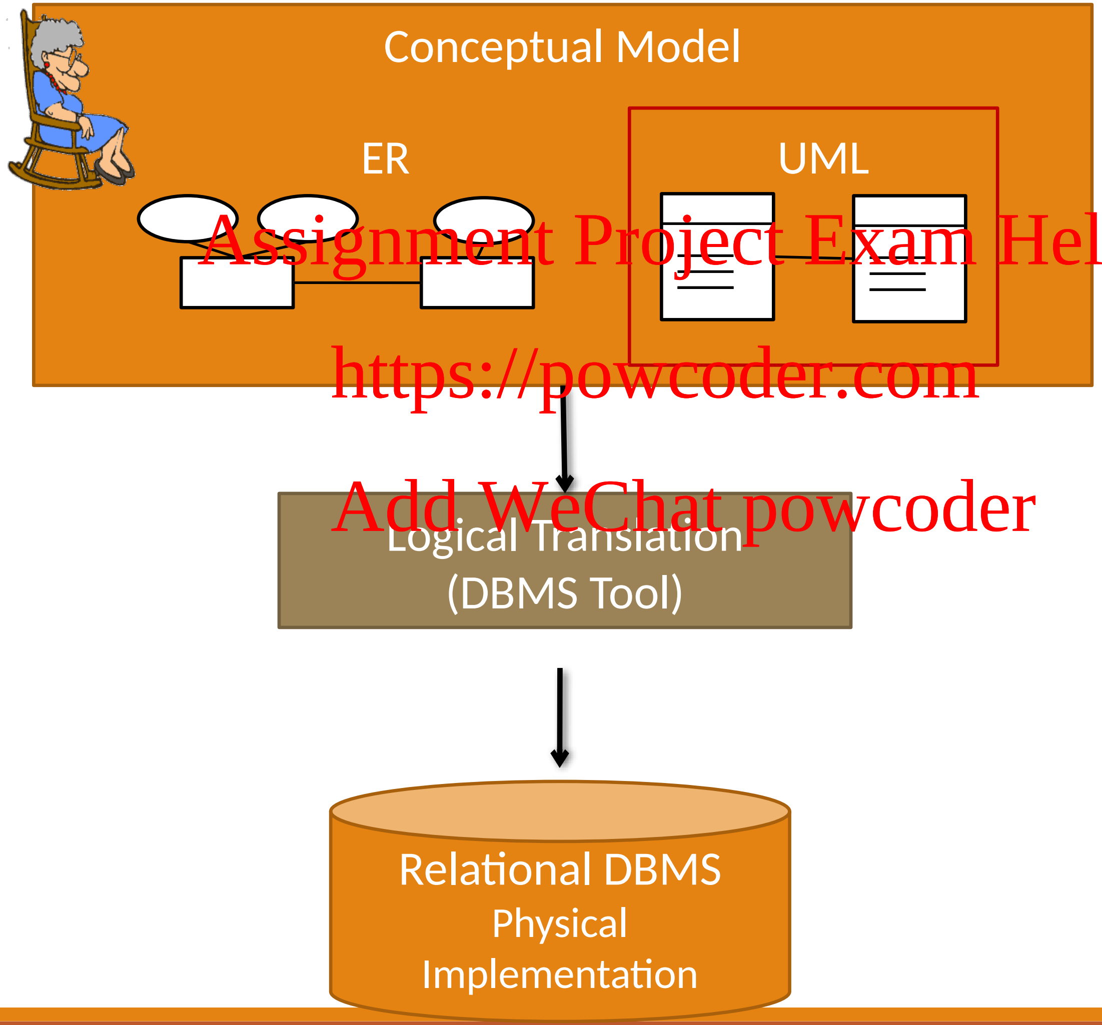
Database Fundamentals

THE RELATIONAL MODEL

The dominant method for managing data in todays world

Relational Database Modelling Overview

- Conceptual Model Translation Process for Relational Model



Week 2

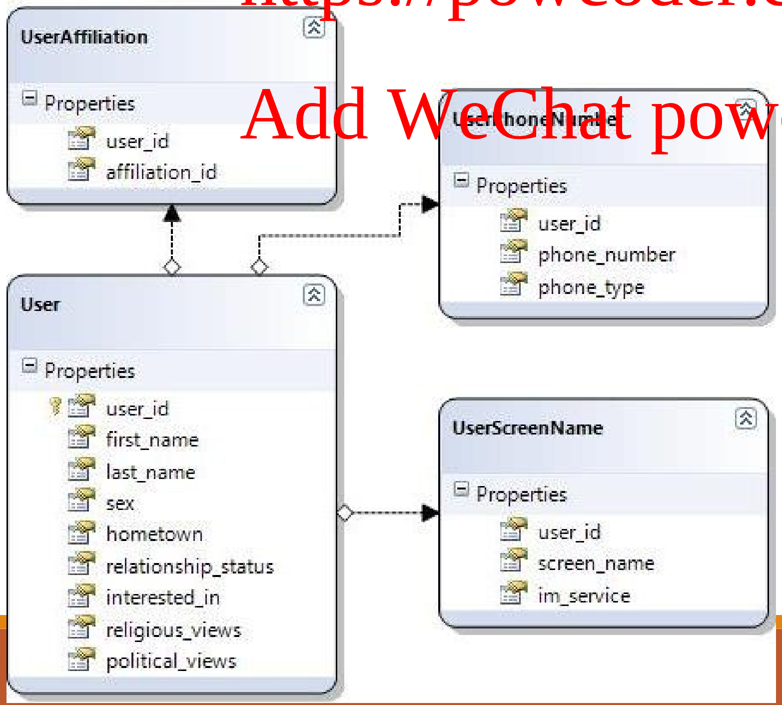
Week 3

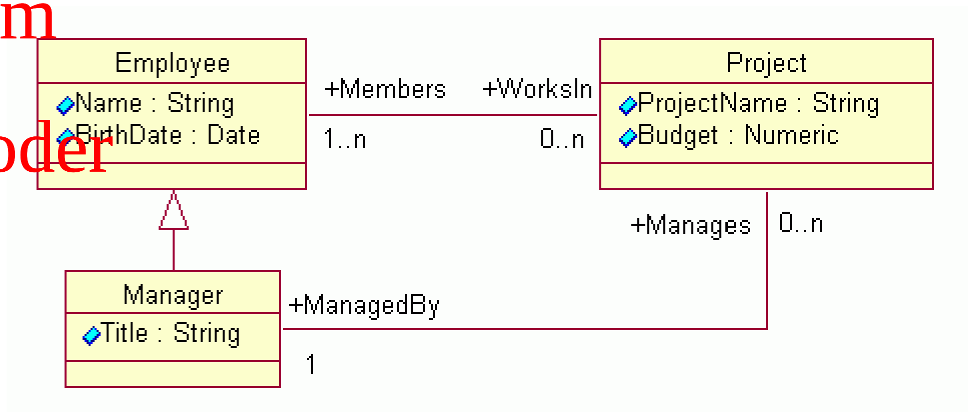
Week 3+

Relational Database Modelling Overview

- Conceptual Modelling
  - Database requirements are collected and visualised as a UML diagram (Unified Modelling Language)
    - Uses higher level modelling to abstract away the complexities of implementation
    - A graphical way of conceptualising data capture requirements to ensure the required data is collected by the database design!

  - Some examples:





Relational Database Modelling Overview

Week 3

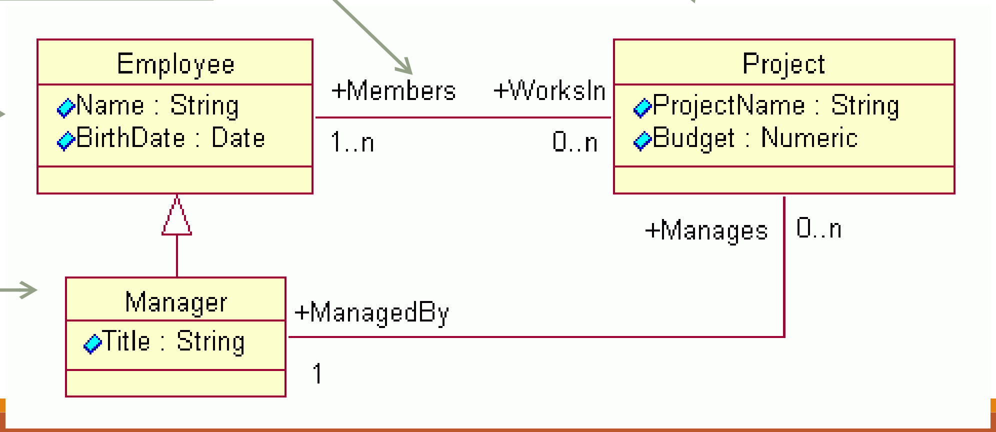
- Logical Modelling
  - The next phase is to create functional relational schemas based on the conceptual design.
    - A written description of how the relational database will be implemented
    - This includes deciding which **candidate key(s)** will become the **primary key** - the value used to locate a specific record (like your studentID or emailID)
  - Example:

ProjectMembers (ProjectName, Employee)
PK(ProjectName, Employee)
FK(projectName) → Project(ProjectName)
FK(Employee) → Employee(Name)

Project(ProjectName, Budget, Manager)
PK(ProjectName)
FK(Manager) → Manager(Name)

Employee(Name, BirthDate)
PK(Name)

Manager(Name, BirthDate, Title)
PK(Name)



PK = Primary Key
FK = Foreign Key

Relational Database Modelling Overview

Week 3+

- Physical Design
  - Take the relational schemas and implement them using a DBMS
  - **Structured Query Language (SQL)** is used to create the relational database and manage data within the database


```
CREATE TABLE Employee (
        Name varchar(200),
        BirthDate date,
        CONSTRAINT employeePk  PRIMARY KEY (Name)
);
```

```
CREATE TABLE Manager (
        Name varchar(200),
        CONSTRAINT managerPk PRIMARY KEY (Name)
);
```

```
CREATE TABLE Project(
        ProjectName varchar(200),
        Budget decimal(6,2),
        Manager varchar(200),
        CONSTRAINT projectPk PRIMARY KEY (ProjectName),
        CONSTRAINT managerFk FOREIGN KEY (Manager)  REFERENCES Manager(Name)
);
```

Relational Concepts

JARGON FOR TABLE TERMINOLOGY

Relational Model – Key Concepts

- In a Relational DB data is held in a series of **Relations**
  - A **Relation** is a table that consists of columns and rows of data

Relational Name	Common Name	Alternative
Relation	Relation Table	Table
Attribute	Column	Field
Tuple	Row	Record

- What makes a table with rows and columns a **Relation**?
  - Within a Table **every column name** must be **UNIQUE**
  - Within a Table **every row** must be **UNIQUE** (no duplicate data!)
  - Every Row must have a **unique Primary Key** that can identify that data row only!

Relational Model – Key Concepts

- A **Relation** defines a real world or conceptual object we collect information about
  - When a relation is implemented in a DBMS, it is often called a **Table**
  - A relational database consists of relations with **distinct** names
    - Eg, a table that stores Customer Information, Bank accounts, Uni Courses

- An **Attribute** is a name for a column in a relational table
  - Each Attribute must have a unique name in a given relational table
  - Every attribute has a **domain**
    - Eg, the Customer table has columns (attributes) for collecting the customer's Family Name, First Name, Address, Age

Relational Model – Key Concepts

- A **Domain** dictates the acceptable values of an attribute
  - E.g. int, decimal(5,2), varchar(200), bit
  - We will refer to domains as Data Types (consistent with Programming Fundamentals)
    - Eg. FirstName is a string of characters of varying length (varchar), or the customer age may be an integer number (int)

- A **Table Schema** defines the structure of a relational table in a relational database (table name + attributes)
  - Customer(CustomerID, FirstName, FamilyName, Address, Age)
    - Relation Name + attributes

Relational Database – Key Concepts

- **Tuples** are instances of a relation or entity and contain the actual raw data
  - A Tuple is a row of data
    - A **relation** consists of one or more unordered tuples
    - We use special commands to order the records returned by the DBMS when necessary

- A **Key** is a set of one or more attributes whose values can uniquely identify a given row of data in a relation
  - The **Primary Key (PK)** is a value that identifies a particular table row of data
  - Candidate (name given to all keys)
  - Unique
  - Surrogate
  - Natural

Relational Database – Key Concepts

- Terms and Notation

Relation name

Attribute name

schema

Physical Implementation:

Students

studID	emailID	studName
50001	leej001	John Lee
50002	bump001	Peter Buman
50003	brod001	David Browns

a tuple

Equivalent Conceptual Design UML:

Students
studID (PK) emailID studName

Students
studID (PK)(varchar) emailID(varchar) studName(varchar)

Attribute ***domain (data type)***

Logical Schema: Students(<u>studID</u>, emailID, studName)

Key Concepts

MORE KEYS THAN DIABLO III

Relational Database – Key Concepts

- **Candidate Key(s) – a** minimum combination of attributes that could become the **Primary Key** when designing a database
  - If there is only one Candidate Key (CK) it is generally selected as the Primary Key
  - Other times, the developer must choose which CK will be used as the Primary Key

- **Primary Key** – the key chosen from the candidates to be implemented in the relation

  - All relations require a **Primary Key (PK)**
  - It is a set of one or more attributes chosen to **uniquely identify** each tuple (or row) of data
    - If you are given a PK, you can locate the specific data
    - Sometimes it is an automatically generated number (eg the current row number in excel)

Relational Database – Key Concepts

- **Primary Key**
  - Every relation has a special attribute or a set of attributes whose values can be used to **uniquely identify** a single row of data (a tuple)
  - It serves to distinguish one row from every other
  - The value(s) must be unique for each row

Student

StudentID	StudentName
1011	Jane Lane
1012	Jeff Wilson
1013	Jane Pain

Course

CourseID	CourseName
INFS1091	Proj Dev
COMP2064	FIT
LMIF3621	Math

Unique value for every row

Relational Database – Key Concepts

- **Composite key**
  - A **Key** (Primary or Other!) that is composed of more than one attribute
    - Example shows a **Composite** Primary Key!

Enrolment

Course	Student	Mark
INFS1091	1011	87
COMP2064	1013	94
LMIF3621	1012	68

Unique values for every row

Relational Database – Key Concepts

- A **Unique Key** is a set of attribute(s) whose values are always unique
  - Any candidate key that is not selected as the Primary Key can be implemented as a **Unique Key** constraint in the DBMS
    - The data is checked to ensure each entry is a unique value and does not already exist in other rows

- **Natural Key** is a minimum set of one or more attributes that exist in the table data that can be used or combined to uniquely identify each row:

  - Bank account/Telephone numbers, email address etc, Name + Birth Date

- **Surrogate Key** – ultimate lazy approach to implementing a Primary Key
  - An incremental number stored as an ID column in a table
  - A randomly generated alpha-numeric value
  - Has nothing to do with the table data

What Key is That?

UserName (PK)	FirstName	LastName
davop001	Paul	Davos
adamj003	Jennifer	Adams
duffl001	Lenny	Duff

userName:

- Candidate Key
- Natural Key
- Surrogate Key
- Unique Key
- Primary Key

What Key is That?

studentID (PK)	userName	FirstName	LastName
10011	davop001	Paul	Davos
10093	adamj003	Jennifer	Adams
10178	duffl001	Lenny	Duff

studentID:

- Candidate Key
- Natural Key
- Surrogate Key
- Unique Key
- Primary Key

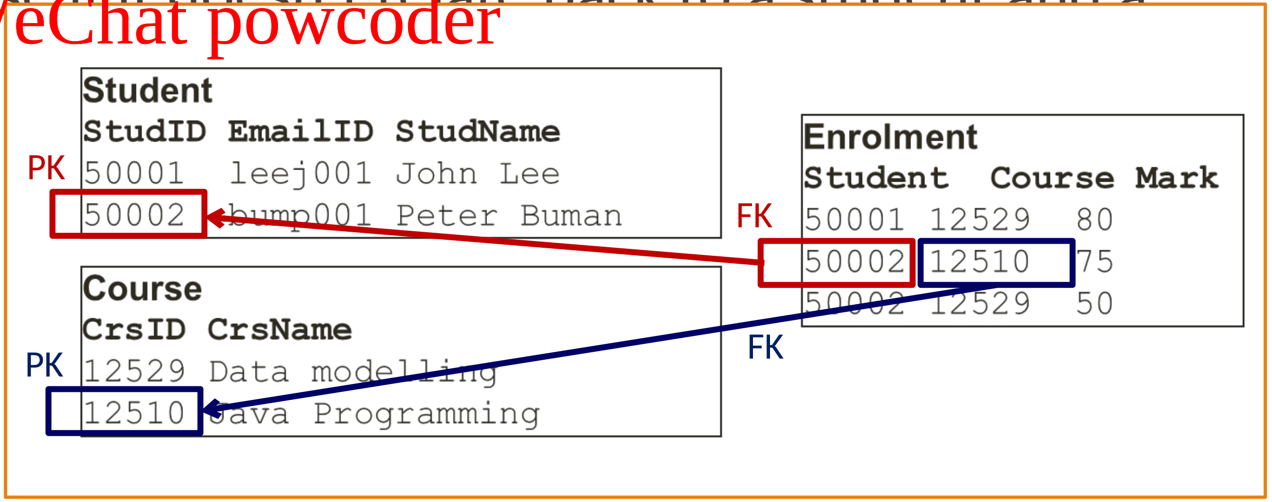
userName:

- Candidate Key
- Natural Key
- Surrogate Key
- Unique Key
- Primary Key

Relational Database – Key Concepts

- **Foreign Keys** are attributes whose values are the **Unique Key** of another Table (primary or unique!)
  - The PK values are copied and <u>stored</u> in the other table as a **Foreign Key** value
  - Foreign keys allow one table to be linked to one or more tuples in another table.
    - hence the term "Relational Database"

  - Example - A course mark is of no use if it doesn't relate back to a student and a course:
    - Enrolment(StudentID) is a foreign key referencing Student(StudentID)
    - Enrolment(CourseID) is a foreign key referencing Course(CourseID)



Relational Database – Key Concepts

- **Foreign Key** Violation:
  - Student 50002 took course 12550 which is not a valid code for any course.
  - Course 12529 was taken by Student 50009 which is not a valid id for any student.
    - **Foreign Keys** are **used** to prevent these scenarios from taking place
    - These are "**orphan**" records that have lost their meaning

Student

StudID	EmailID	StudName
50001	leej001	John Lee
50002	bump001	Peter Buman

??

Course

CrsID	CrsName
12529	Data modelling
12510	Java Programming

??

Enrolment

Student	Course	Mark
50001	12529	80
50002	$\boxed{12550}$	75
$\boxed{50009}$	12529	50

Database Fundamentals

LOGICAL SCHEMA DESIGN

Deciding what you need to capture and how it should be stored

Databases – Relational Model Schema

- Database Designs
  - A Database design <u>should</u> capture **all** the required information
    - Both current and **future requirements**

  - More than one design may be plausible
    - One design may be better than another
    - A **good design** in one situation may <u>perform poorly in another</u>

  - A good design should take into account how the data will be used
    - Frequent user searches
    - Auditing purposes
    - Predictions/data mining

Database – Normalisation

- Database Designs
  - Consider what data needs to be stored and how it will be used

Databases – Relational Model Schema

- Bad Database Designs can create data quality issues
  - Tracking customer shopping habits
    - Customer account No
    - Customer Name
    - Customer Address
    - Products purchased
    - Stores they shop

Example Logical Schema: custHabbits(acctNo, custName, custAddress, product, storeName)

acctNo	custName	custAddress	product	storeName
1101	Mary Jane	123 Lane	Coffee maker	Hardly Normal
3311	James	424 Sa	Milk	Foodville
1101	Mary Jane	123 Lane	Knife and fork set	Hardly Norma
1101	Mary Jane	123 Lane	Quilt set	Quilts ’n’ Things
2211	Bob	111 Drive	Coffee maker	Hardly Normal
3311	James	424 Sa	Milk	Coz

Databases – Relational Model Schema

- Bad Database Designs can create data quality issues
  - Tracking customer shopping habits
    - Customer account No
    - Customer Name
    - Customer Address
    - Products purchased
    - Stores they shop

Every time our user creates a new record They need to re-enter **all** this information again and again

Example Logical Schema: custHabbits(acctNo, custName, custAddress, product, storeName)

acctNo	custName	custAddress	product	storeName
1101	**Mary Jane**	123 Lane	Coffee maker	**Hardly Normal**
3311	James	424 Sa	Milk	Foodville
1101	**Mary Jane**	123 Lane	Knife and fork set	**Hardly Norma**
1101	**Mary Jane**	123 Lane	Quilt set	Quilts 'n' Things
2211	Bob	111 Drive	Coffee maker	Hardly Normal
3311	James		Milk	Coz

Redundancy

Databases – Relational Model Schema

- Bad Database Designs can create data quality issues
  - Tracking customer shopping habits
    - Customer account No
    - Customer Name
    - Customer Address
    - Products purchased
    - Stores they shop

If our user notices a mistake in one record and corrects it, do they remember to correct all the others?

Example Logical Schema: custHabbits(acctNo, custName, custAddress, product, storeName)

acctNo	custName	custAddress	product	storeName
1101	Mary Jane	123 Lane	Coffee maker	Hardly Normal
3311	James	424 Sa	Milk	Foodville
1101	Mary Jane	123 Lane	Knife and fork set	**Hardly Norma**
1101	Mary Jane	123 Lane	Quilt set	Quilts 'n' Things
2211	Bob	111 Drive	Coffee maker	Hardly Normal
3311	James		Milk	Coz

Update Anomaly

Databases – Relational Model Schema

- Bad Database Designs can create data quality issues
  - Tracking customer shopping habits
    - Customer account No
    - Customer Name
    - Customer Address
    - Products purchased
    - Stores they shop

If our user deletes this purchase record we loose **all** our information we had on Bob!

Example Logical Schema: custHabbits(acctNo, custName, custAddress, product, storeName)

acctNo	custName	custAddress	product	storeName
1101	Mary Jane	123 Lane	Coffee maker	Hardly Normal
3311	James	424 Sa	Milk	Foodville
1101	Mary Jane	123 Lane	Knife and fork set	Hardly Norma
1101	Mary Jane	123 Lane	Quilt set	Quilts 'n' Things
2211	**Bob**	**111 Drive**	**Coffee maker**	**Hardly Normal**
3311	James		Milk	Coz

Deletion Anomaly

Databases – Relational Model Schema

- A Good relational schema design should take into account usage, avoid redundancies and anomalies
  - Redundancy
    - Information is recorded unnecessarily in multiple **tuples** (rows)
  - Update Anomaly
    - Information can be changed in one tuple leaving old information in another
  - Insertion Anomaly
    - A bad design that forces unrelated information to be recorded as well and this information may be wrong
  - Deletion Anomaly
    - Deletion of a record results in <u>complete loss</u> of other information

Databases – Relational Model Schema

- Example – A Better Design?
  - Ensure a **Relation** only captures <u>one type </u>of information about an **entity** or **object**
  - **Question:** Is the new design any good for looking up customer records?

customer(<u>acctNo</u>, custName, custAddress)

customerPurchases(<u>acctNo, product</u>, storeName)

customer

acctNo	custName	custAddress
1101	Mary Jane	123 Lane
3311	James	424 Sa
2211	Bob	111 Drive

customerPurchases

acctNo	product	storeName
1101	Coffee maker	Hardly Normal
3311	Milk	Foodville
1101	Knife and fork set	Hardly Normal
1101	Quilt set	Quilts ’n ’Things
2211	Coffee maker	Hardly Normal
3311	Milk	Coz

Databases – Relational Model Schema

A poorly designed database causes anomalies:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	B01
Sam	CS145	B01
..	..	..

If every course is in only one room, contains **redundant** information!

If we update the room number for just one tuple (**Update anomaly)**

Suppose everyone drops the course suddenly… we lose information about where the course is! (**Delete Anomaly)**

We may not be able to create a room reservation without students. Need to know every detail (**<u>Insert anomaly)</u>**

Database – Normalisation

- Normalisation is the process of **decomposing large** relational **schemas** into **smaller** more efficient **schemas**

  - Smaller schemas that contain only **one copy** of any given record
    - Avoids issues that result from having multiple copies of the same record (typos etc)

  - Smaller schemas that contain only **relevant data**
  - Smaller schemas that are more **easily searched**

  - Normalisation **prevents update anomalies and data inconsistencies**
  - Works well where data is updated frequently but at the expense of information retrieval
    - We will see this in coming weeks

Database – Normalisation

- The Problem of Un-Normalised Data:
  - Imagine the table has 1000s of rows
  - How do you find the franchises with stores in London?
  - How do you ensure data integrity if "storeLocations" is an unlimited text field?
    - Can you prevent a person writing **Londn** or **L'dn** intentionally or by accident?

franchiseID	storeLocations
101	Chicago, **London**, New York, Paris, San Francisco
102	Seattle
103	**London**, Munich, Paris, Athens
104	New York, **London**

Database – Normalisation

- The Problem of Poorly Normalised Data:
  - What if there are stores in other cities?
  - How do you get a list of the cities for a particular store?
    - You'd need to check each column to see if it contained data!

FranchiseID	City1	City2	City3	City4	City5
101	Chicago	London	New York	Paris	San Francisco
102	Seattle	*NULL*	*NULL*	*NULL*	*NULL*
103	London	Munich	Paris	Athens	*NULL*
104	New York	London	*NULL*	*NULL*	*NULL*

Database – Normalisation

- How should we represent this information?

FranchiseID	City1	City2	City3	City4	City5
101	Chicago	London	New York	Paris	San Francisco
102	Seattle	*NULL*	*NULL*	*NULL*	*NULL*
103	London	Munich	Paris	Athens	*NULL*
104	New York	London	*NULL*	*NULL*	*NULL*

Database – Normalisation

- How should we represent this information?

FranchiseID	City1	City2	City3	City4	City5
101	Chicago	London	New York	Paris	San Francisco
102	Seattle	*NULL*	*NULL*	*NULL*	*NULL*
103	London	Munich	Paris	Athens	*NULL*
104	New York	London	*NULL*	*NULL*	*NULL*

Franchises
101
102
103
104 …

FranchiseCities	
101	Chicago
101	London
101	New York
…	…
103 …	London

Cities
Chicago
London
New York
Paris …

Database – Normalisation

- 1st Normal Form
  - All tables should be "flat"
  - All occurrences of a record type must contain the same number of fields
    - All values in a given column must be of the same data type (what does this mean?)
    - A value should **NOT be composed of multiple values**

Student	Courses
Mary	{CS145,CS229}
Joe	{CS145,CS106}
…	…

Not in 1^{st} NF

Student	Course
Mary	CS145
Mary	CS229
Joe	CS145
Joe	CS106

1^{st} NF Equivalent

Database – Normalisation

- 1st Normal Form
  - All tables should be “flat”
  - All occurrences of a record type must contain the same number of fields
    - All values in a given column must be of the same data type (what does this mean?)
    - A table should have **NO repeating groups** of values

Student	Course1	Course2
Mary	CS145	CS229
Joe	CS145	CS106
…	…	

Not in 1st NF

Student	Course
Mary	CS145
Mary	CS229
Joe	CS145
Joe	CS106

1st NF Equivalent

Database – Normalisation

How can we improve this?

Student Enrolments

Student	Postcode	Suburb	Course	CourseName	Room	Mark
Mary	5000	Adel	CS145	FIT	B01	D
Mary	5000	Adel	CS229	PF	B01	C
Jeff	5001	Adel	CS229	PF	B01	C
Jane	5092	Modbury Nth	CS106	DBF	B02	HD
Joe	5092	Modbury	CS145	FIT	B01	P
Joe	5092	Modbury	CS106	DBF	B02	HD

Student: Student → Suburb + Postcode

Enrolments: Student + Course → Mark

Courses: Course → Room

Optional:

Suburbs: Suburb → Postcode ??

1st NF

- Separate information into **relations** that represent **real-world or conceptual objects**
  - Student, Courses, Enrolments, Suburbs?
  - Look for attributes that depend on the values of other attributes

Database – Normalisation

How can we improve this?

Student Enrolments

Student	Postcode	Suburb	Course	CourseName	Room	Mark
Mary	5000	Adel	CS145	FIT	B01	D
Mary	5000	Adel	CS229	PF	B01	C
Jeff	5001	Adel	CS229	PF	B01	C
Jane	5092	Modbury Nth	CS106	DBF	B02	HD
Joe	5092	Modbury	CS145	FIT	B01	P
Joe	5092	Modbury	CS106	DBF	B02	HD

Student

Student	Postcode	Suburb
Mary	5000	Adel
Jeff	5001	Adel
Jane	5092	Modbury Nth
Joe	5092	Modbury

Now:

Student → Postcode, Suburb

- The Student always determines the Suburb and the Postcode
- Neither the Suburb or the Postcode is always unique

Database – Normalisation

How can we improve this?

Student Enrolments

Student	Course	CourseName	Room	Mark
Mary	CS145	FIT	B01	D
Mary	CS229	PF	B01	C
Jeff	CS229	PF	B01	C
Jane	CS106	DBF	B02	HD
Joe	CS145	FIT	B01	P
Joe	CS106	DBF	B02	HD

Enrolment

Student	Course	Mark
Mary	CS145	D
Mary	CS229	C
Jeff	CS229	C
Jane	CS106	HD
Joe	CS145	P
Joe	CS106	HD

+

Courses

Course	CourseName	Room
CS145	FIT	B01
CS229	PF	B01
CS106	DBF	B02

Now:

Student + Course →Mark

Course → CourseName, Room

- The Course the student enrols in determines the mark
- The Course always determines the room

Database – Normalisation

Student Enrolments (Original mega table)

Student	Postcode	Suburb	Course	CourseName	Room	Mark
Mary	5000	Adel	CS145	FIT	B01	D
Mary	5000	Adel	CS229	PF	B01	C
Jeff	5001	Adel	CS229	PF	B01	C
Jane	5092	Modbury Nth	CS106	DBF	B02	HD
Joe	5092	Modbury	CS145	FIT	B01	P
Joe	5092	Modbury	CS106	DBF	B02	HD

Student

Student	Postcode	Suburb
Mary	5000	Adel
Jeff	5001	Adel
Jane	5092	Modbury Nth
Joe	5092	Modbury

Courses

Course	CourseName	Room
CS145	FIT	B01
CS229	PF	B01
CS106	DBF	B02

Enrolment

Student	Course	Mark
Mary	CS145	D
Mary	CS229	C
Jeff	CS229	C
Jane	CS106	HD
Joe	CS145	P
Joe	CS106	HD

Database – Normalisation

How can we improve this?

Student

Student	Postcode	Suburb
Mary	5000	Adel
Jeff	5001	Adel
Jane	5092	Modbury Nth
Joe	5092	Modbury

Student

Student	Suburb
Mary	Adel
Jeff	Adel
Jane	Modbury Nth
Joe	Modbury

Suburbs

Suburb	Postcode
Adel	5000
Modbury Nth	5092
Modbury	5092

3rd NF Equivalent

This example does not work for all suburbs!

Now:

Student → Suburb

Suburb → Postcode

- The Student always determines the Student Name and address
- The Suburb always determines the Postcode

Database – Normalisation

How can we improve this?

Student

Student	**Postcode**	**Suburb**
Mary	5000	Adel
Jeff	5001	Adel
Jane	5092	Modbury Nth
Joe	5092	Modbury

Student

Student	**Postcode**	**Suburb**
Mary	5000	Adel
Jeff	5001	Adel
Jane	5092	Modbury Nth
Joe	5092	Modbury

Neither the Postcode or Suburb can uniquely identify the other:

- Sometimes Adelaide is 5000, sometimes it is 5001
- Sometimes 5092 is Modbury sometimes it is Modbury North

Now:

Student → Postcode, Suburb

- The Student always determines the Suburb and the Postcode
- Neither the Suburb or the Postcode is always unique

Database – Normalisation

- Original "mega" Logical Schema:
  - Student(Student, Postcode, Suburb, Course, CourseName, Room, Mark)
    - PK(Student)

- Relations should represent real world or theoretical objects

  - A relation should represent one type of object or a relationship between two objects, not every everything you think you need to capture

Is a student name a good Primary Key (PK)?

- New Logical Schemas:
  - Student(student, postcode, suburb, postcode)
    PK(student)
  - Courses(course, courseName, room)
    PK(course)
  - Enrolments(student, course, mark)
    PK(student, course)

A better schema using a **surrogate key**:

Students(studentID, studentName, suburb, postcode)

Courses(courseID, courseName, room)

Enrolments(studentID, courseID, mark)

Logical Schema Design

- So what does this all mean?
- When you are designing a database you should:

1. Create a List of **all** the **real-world** AND **conceptual objects** you need to store data about

2. To each of those items, list the **attributes** required to capture the desired data – ALL of them!

3. Check that each of your objects only captures only **relevant information** to that object
   - (relevant to that object and **only** to that object)

4. Check to see if you have any **candidate keys**
   - Do they apply to **ALL Records**?
   - If so, pick the best one (usually the smallest one) as the **Primary Key**.
   - If not, add an artificial (**surrogate**) Primary key – a standard ID field with a number that increases for each new row.

Database Fundamentals

NEXT WEEK: CONCEPTUAL DESIGN - UNIFIED MODELLING LANGUAGE (UML)