6 Database

(a) • Determine the attributes of a database: table, record and field.
(b) • Explain the purpose of and use primary, composite and foreign keys in tables.
(c) • Explain with examples, the concept of data redundancy and data dependency.
(d) • Reduce data redundancy to third normal form (3NF).
(e) • Draw entity-relationship (ER) diagrams to show the relationship between tables.
(f) • Sqlite3

- A database is a **structured collection of organized data** that stored data persistently.
- It is designed to efficiently (CRUD operations)
  - Store data/ create data
  - Retrieve data
  - Update data
  - Delete data

how fast you can conduct these

-> stored persistently
- store it in a way that you can retrieve it in its original form

- A database system (DBMS) is a software that manages the database in order to provide
  - data integrity
  - data security
  - efficient access to data

So how do you store unstructure data or raw data ?

Flat Files as a Database

When referred as a medium of storing data, a **flat file** is usually a plain text file or spreadsheet document, where records usually follow a uniform format, but there are no structures for indexing or recognizing relationships between records. E.g, Consider a tution centre storing its students' information using a text file with the following content

-> storing data in its raw/original form
- not compressed or edited
- database can't store any sort of data, must be the same data as the database.

```
Name,   Gender, Age,    Contact,    Subjects
Alex,   M,      15,     91234567,   (Math, Science, English)
Ben,    M,      13,     ,           (English,Math)
Cindy,  Female, Fifteen,,
Damian, M,      15,     91111111,   (Science,Math,English)
Erica,  F,      13,     82222222,
Fanny,  F,      ,       93456788,   (English)
Gopal,  Male,   15,     82343434,   (Science)
Damian, M,      14,     61234562,   (English,Math)
Ben,    M,      13,     66-12345678,(English,Math)
```

-> structure present, has new line character (/n or \n idk)

What are some potiential problems with the flat file ?

- inconsistent data types used(integers and strings)
- empty data fields
- variable number of subjects

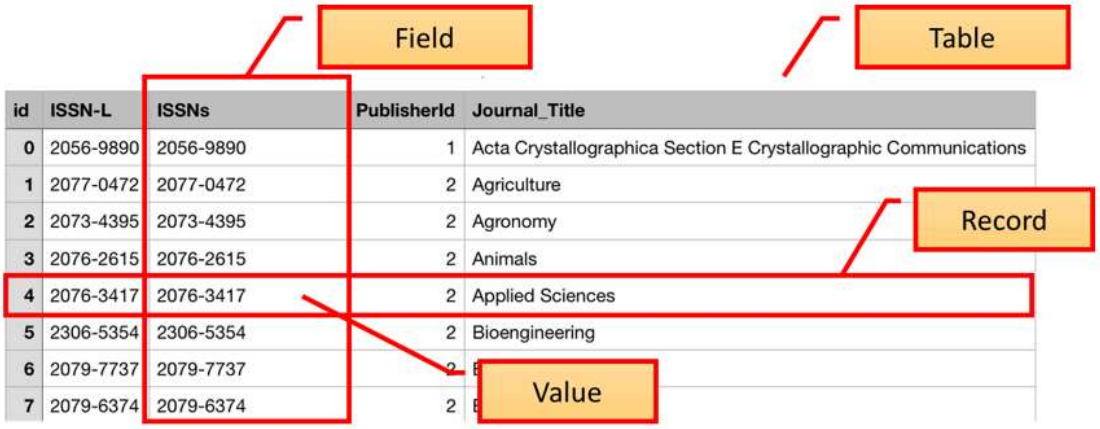
6.1 Relational Database

A Relational database is a database where data are organised in one or more tables with relationships between them, i.e. a collection of relational tables.

A **table** (also called **relation** in relational database) is a two-dimensional representation of data stored in rows (records) and columns (fields). A table stores data about an **entity** – i.e. some "thing" about which data are stored, for example, a customer or a product.

In each table, a complete set of data about a single item is called a **record**, i.e. it's a row in a table.

On the other hand, a column in a table is called an **field**. **Attributes** are the describing characteristics or properties that define all items pertaining to a certain category applied to all cells of a column.



id	ISSN-L	ISSNs	PublisherId	Journal_Title
0	2056-9890	2056-9890	1	Acta Crystallographica Section E Crystallographic Communications
1	2077-0472	2077-0472	2	Agriculture
2	2073-4395	2073-4395	2	Agronomy
3	2076-2615	2076-2615	2	Animals
4	2076-3417	2076-3417	2	Applied Sciences
5	2306-5354	2306-5354	2	Bioengineering
6	2079-7737	2079-7737	2	B
7	2079-6374	2079-6374	2	B

A database management system (DBMS) is a piece of software that provides the following features :

- basic database design, including tables, relationship and user queries
- a **data definition language (DDL)** that the database designer uses to define the tables of the database
- a data dictionary that includes:
  - the descriptions of tables, relationships and all design information such as indexing
  - the rules about data integrity including validation rules for all attributes.
- a **data manipulation language (DML)** called SQL:
  - for inserting, amending and deleting data records
  - backup of the database data

- control of multi-user access to the data.

Example 1

The following table has 5 records and 3 fields, and the attributes are `Colour`, `Price` and `Stock`.

Colour	Price	Stock
Red	0.50	30
Green	0.50	18
Yellow	0.80	43
Blue	0.90	66
White	0.85	39

can compare tables to OOP

Example 2

Attributes can be used to describe a table. The following table has the following description:

> `Student` (RegNo, Name, Gender, MobileNo)

name of table must be a noun, singular and first letter must be capitalised

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98671715
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

In general, a table in a relational database can be described as:

```
TABLE_NAME(ATTRIBUTE_1, ATTRIBUTE_2, ATTRIBUTE_3,
ATTRIBUTE_4,....)
```

Usually, the description of the entity are used for `TABLE_NAME` as well.

Exercise 3

Provide the table description of the following table on number of balloons sold and in stock.

Colour	Price	AmountSold	Stock
Red	0.50	40	30
Green	0.50	17	18
Yellow	0.80	57	43
Blue	0.90	24	66
White	0.85	36	39

In [ ]:
```
#YOUR_ANSWER_HERE
```
Balloon (Colour, Price, AmountSold, Stock)

6.2 Properties of a table

Table in a relational database if it fulfills the following conditions:

- Values are **atomic**, i.e., for each record, each entry contains only 1 piece of information, e.g. in Example 2, a student cannot have 2 mobile phone numbers in the table.
- Columns are of the same kind i.e, Same Data type
- Rows are unique, i.e no repeated rows
- The order of columns is insignificant
- Each column must have a unique name
- Fixed number of columns

6.2.2 Key Fields

A **key field**, or **key** in short, are fields in a table that serve some special fuctions

There are different types of keys.

- A **primary key** is either a column or a combination of columns in a database that uniquely identifies the specific record in question. denoted by underlining (row)

(full) name - you're the only person with that name)

- A **composite key** is a combination of two or more fields in a table that can be used to uniquely identify each record in a table. Uniqueness is only guaranteed when the fields are combined. (≥ 2 columns to form primary key)

example: gender and cca - the only guy that's in indian dance

- A **foreign key** is a column (field) in one table that refers to the primary key in another table, i.e. it links to a primary key in a second table and form relationships between the tables. Foreign keys is indicated by using a dashed underline or asterisk *

$$\text{Student}(\underline{\text{MatricNo}},\ \text{Name},\ \text{Gender},\ \text{CivicsClass}^*)$$

$$\text{ClassInfo}(\underline{\text{CivicsClass}}, \text{CivicsTutor}, \text{Homeroom})$$

- A candidate key is a column/columns that **can be used** as the primary key. i.e. a primary key is a candidate key.
- A secondary key is a candidate key that is **not chosen** to be used as a primary key. They are usually used as an index to optimise searches in a table rejects / every other column

Exercise 4

Consider the following table.

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98671715
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

- What is/are the candidate key(s)?
- What is the primary key?

In [ ]:
```
#YOUR_ANSWER_HERE
```
RegNo or MobileNo can be candidate key(s) - unique/no repeated info

Exercise 5

Consider the following table.

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Agnes	F	18S12
4	Aisha	F	18S12

RegNo	Name	Gender	CivicsClass
5	Ajay	M	18S12
6	Alex	M	18S12
7	Alice	F	18S12
8	Amy	F	18S12
9	Andrew	M	18S12
10	Andy	M	18S12
1	Adam	M	18A10
2	bala	M	18A10
3	Bee Lay	F	18A10
4	Ben	M	18A10
5	Boon Kiat	M	18A10
6	Boon Lim	M	18A10
7	Charles	M	18A10
8	Chee Seng	M	18A10
9	Cher Leng	F	18A10
10	Choo Tuan	M	18A10

- What is/are the composite key(s)?

In [ ]:
```
#YOUR_ANSWER_HERE
```

Name and CivicsClass ✗

RegNo and CivicsClass (people with the same name in the same class)

Example 6

Consider the following tables `Student` and `ClassInfo` respectively.

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12

<- foreign key (can assume a foreign key points to the primary key, because its unique, of another table)

CivicsClass	CivicsTutor	HomeRoom
18S12	Mr Tan	CR1
18A10	Ms Aishya	CR2

- What is/are the primary key(s) in each table?

- What is the attribute in the table `ClassInfo` that is the foreign key in the `Student` table?

In [ ]:
```
#YOUR_ANSWER_HERE
```

6.3 Data Normalisation

- The objective of normalisation is to
  - reduce data redundancy
  - maintain data integrity (accuracy and consistency of the data)

Consider an application that stores the products ordered by a customer in a flat file:

(Un-normalised form) **UNF**

```
CustNum,    CustName,   City,       Country, Products (ProductName,
Price, ProductName, Price, ...)
005,        Bill Jones, London,     England,
(Table,$50,Desk,$25,Chair,$10)
008,        Amber Arif, Lahore,     Pakistan,(Desk,$25,
Cupboard,$60)
014,        M. Ali,     Kathmandu,  Nepal,   (Cabinet,$65)
002,        Omar Norton,Cairo,      Egypt,
(Cupboard,$60,Table,$50,Desk,$25)
```

Can we import these data into a relational database table ?

def must know

6.3.1 First Normal Form (1NF)

For a table to be in 1NF:

- all columns must be atomic, i.e.in the database, entities (objects of interest, e.g. person, item, place) do not contain repeated groups of attributes.
  - Columns would not hold a collection such as an array or another table. This means the information in each column cannot be broken down further.
- every row must be unique, i.e. identify a primary key

insert, update and delete

anomaly: data that leaves blank space (example: company releases a new product but no one has bought it yet, so everything box in a row is blank except product name)

We can create a `ORDER` table and insert the rows as follows to satify the 1NF requirement

redundant ->

OrderNum	CustNum	CustName	CityName	CountryName	ProductName	Price
1	005	Bill Jones	London	England	Table	50
2	005	Bill Jones	London	England	Desk	25
3	005	Bill Jones	London	England	Chair	10

CustNum	CustName	CityName	CountryName	ProductName	Price
008	Amber Arif	Lahore	Pakistan	Desk	25
008	Amber Arif	Lahore	Pakistan	Cupboard	60
014	M. Ali	Kathmandu	Nepal	Cabinet	65
002	Omar Norton	Cairo	Egypt	Cupboard	60
002	Omar Norton	Cairo	Egypt	Table	50
002	Omar Norton	Cairo	Egypt	Desk	25

The table definition is thefore:

Order(CustNum, CustName, CityName, CountryName, ProductName, Price)

The table above is now in **1NF**

What is wrong with this table ?

6.3.2 Second Normal Form (2NF)

if not composite, immediately 2NF (?)

To continue with our process of normalisation, we will look at the dependency between the attributes in a table.

Let x, y be attributes in a table. We say that attribute y is **functionally dependent** on attribute x (usually the primary key), if for every valid instance of x, the value of x **uniquely determines** the value of y ($x \rightarrow y$).

for non key attributes, do ALL of them depend on primary key

Let y be an attribute and S be a set of attributes of a table. y is **fully dependent** on S if all the attributes in S are required to **uniquely determine** the value of y. If not all the attributes are required, we say that y is **partially dependent** on S.

For a table to be in 2NF:

- it has to be in 1NF
- every non-key attribute must be **fully** dependent on **all** of the primary key. This means no attribute can depend on part of the primary key only
- if the primary key of a table is **NOT** a composite key, the table is alreday in 2NF

We can transform the `Order` table into 2NF by

- moving the set of dependent attributes to new tables
- linking the new table to the original table with a foreign key.

ProductName	Price
Table	50

ProductName	Price
Desk	25
Chair	10
Cupboard	60
Cabinet	65

all these attributes depend on CustNum

CustNum	CustName	CityName	CountryName
005	Bill Jones	London	England
008	Amber Arif	Lahore	Pakistan
014	M. Ali	Kathmandu	Nepal
002	Omar Norton	Cairo	Egypt

foreign key

CustNum	ProductName
005	Table
005	Desk
005	Chair
008	Desk
008	Cupboard
014	Cabinet
002	Table
002	Desk
002	Cupboard

The tables definitions in 2NF are therefore:

Product (ProductName, Price)

Customer (CustNum, CustName, CityName, CountryName)

Order (CustNum * , ProductName *)

6.3.3 Third Normal Form (3NF)

- Be in 2NF
- Any non-primary key attribute must **not** be dependent on any other non-key attribute

Consider the `Customer(2NF)` table.

CustNum	CustName	CityName	CountryName
005	Bill Jones	London	England
008	Amber Arif	Lahore	Pakistan
014	M. Ali	Kathmandu	Nepal
002	Omar Norton	Cairo	Egypt

Note that the table is in 2NF but not in 3NF as the attribute City determines the attribute Country, so we have two non-key attributes which are dependent. (if the City is London, then the Country is always going to be England)

To make it 3NF, we break the table down further into the following tables `Customer(3NF)` and `City`.

CustNum	CustName	CityName
005	Bill Jones	London
008	Amber Arif	Lahore
014	M. Ali	Kathmandu
002	Omar Norton	Cairo

CityName	CountryName
London	England
Lahore	Pakistan
Kathmandu	Nepal
Cairo	Egypt

To summarize, during the normalization process we end up with more tables, but each table is small enough to enable us to retrieve the information that we want and by that, we avoid data redundancy and maintain data integrity.

The final table definitions (relations) of the database is as follows:

Product (ProductName, Price)

Customer (CustNum, CustName, CityName *)

City (CityName, CountryName)

Order (CustNum * , ProductName *)

6.4 Entity-Relationship Diagram

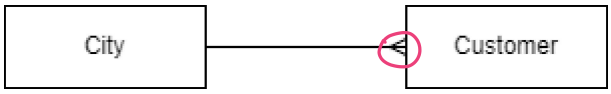
Recall that entities are objects of which data are stored in the database. To illustrate the relationship between entities, an **entity–relationship diagram (E–R diagram)** can be used.

In an E-R diagram,

- entities are represented as rectangles, e.g the diagram below represent the entity customer

- relationships, which are the between two entities, are represented therefore by specific lines connecting the rectangles. There are 3 types of relationships

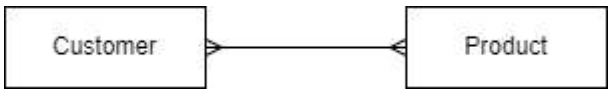
- **one to many** : When a single instance of an entity is associated with more than one instances of another entity, e.g. A city has many customers, A customer can only come from 1 city (remember the backward !!)



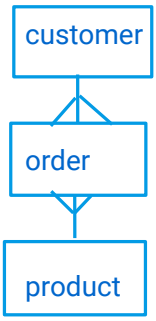
-> crow feet
- implies that entity is a foreign key to another set of data

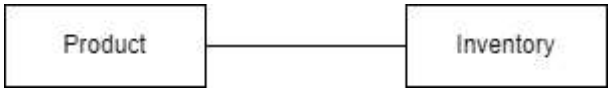
- **many to many** : when more than one instances of an entity is associated with more than one instances of another entity, e.g. A customer can order many products, A product can be order by many different customer

customer and product cannot use foreign and primary key because its not in the 'definitions of database'



- **one to one** : when a single instance of an entity is associated with a single instance of another entity, This is used when you want to extend the attributes of a single entity to another table. e.g when you want to store inventory details for a product.





Exercise 7

a) Transform the following information into 3NF by writing the table definitions

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAInfo
1	Adam	M	18S12	Peter Lim	TR1	Tennis Teacher IC = Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir Teacher IC = Adeline Wong, Student Council Teacher IC = David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby Teacher IC = Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton Teacher IC = Lilian Lim
6	Bee Lay	F	18A10	Pauline Lee	TR2	Choir Teacher IC = Adeline Wong, Chess Club Teacher IC = Edison Poh

primary Key, foreign key *

b) Draw the ERD for the database

Exercises

Worksheet 1

Worksheet 2