OCR Computer Science A Level

1.3.2 Databases

Concise Notes

Specification:

1.3.2 a)

- Relational Database
- Flat File
- Primary Keys, Foreign Keys, Secondary Keys
- Entity relationship modelling
- Normalisation
- Indexing

1.3.2 b)

- Methods of capturing, selecting, managing, and exchanging data

1.3.2 c)

- Normalisation

1.3.2 d)

- SQL

1.3.2 e)

- Referential Integrity

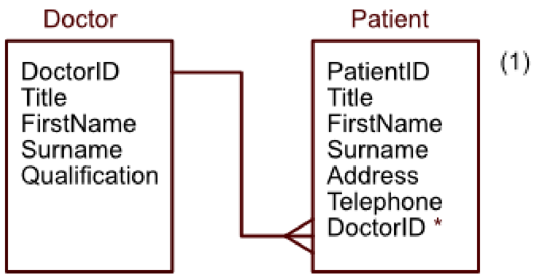
1.3.2 f)

- Transaction processing
- ACID (Atomicity, Consistency, Isolation, Durability)
- Record locking
- Redundancy

Relational Database

Relational Databases

- A relational database is one which uses different tables for different entities.
- An entity is an item of interest about which information is stored.
- The diagram on the right shows a relational database connecting two tables.



Flat File

- A flat file database consists of a single file.
- The flat file will most likely be based around a single entity and its attributes.
- Attributes are the categories about which data is collected.
- Flat files are typically written out in the following way:

Entity1(<u>Attribute1</u>, Attribute2, Attribute3 …)

- For the example in the table below, the description would be laid out as:

Car(<u>CarID</u>, Age, Price)

(2)

Car		
CarID	**Age**	**Price**
Car1	5 years	£1,500
Car2	2 years	£2,400

Primary Key

- The unique identifier which is different for each object added to the database.
- In example (2), the unique identifier is the CarID.
- In example (1), the primary key for the doctor table is DoctorID and the primary key for the patient table is PatientID.

Foreign Key

- A foreign key is the attribute which links two tables together.
- In example (1), DoctorID is the foriegn key, as it exists.

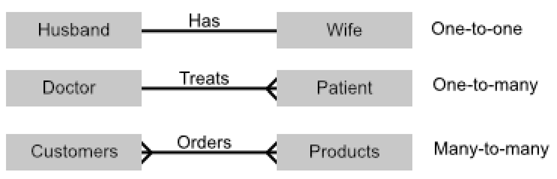
Secondary Key

- A secondary key is used to enable a database to be searched quickly
- In example (1), a secondary index (secondary key) can be set up on the Surname attribute.

- This will allow the table to be sorted on this attribute.

Entity Relationship Modelling

- One-to-one: Each entity can only be linked to one other entity.
- One-to-many: One table can be associated with many other tables.
- Many-to-many: One entity can be associated with many other entities and the same applies the other way round
- The image shows how this is represented diagrammatically.



Normalisation

- The process of coming up with the best possible design for a relational database is called normalisation.
- Normalisation tries to accomplish the following things:
  - No redundancy (unnecessary duplicates)
  - Consistent data throughout linked tables.
  - Records can be added and removed without issues.
  - Complex queries can be carried out.

There are three levels of normalisation:

First Normal Form

- No attribute can contain more than a single value.

Second Normal Form

- No partial dependencies.
- Is in first normal form.

Third Normal Form

- Is in second normal form.
- Contains no non-key dependencies.
- A non key dependency is when the attribute depends on the value of the primary key and nothing else.

Indexing

- Method used to store the position of each record when ordered by a certain attribute.
- Used to look up and access data quickly.
- Primary key is automatically indexed.

Handling Data

Capturing Data

- Data needs to be input into the database and there are various ways of doing this.
- The chosen method is always dependent on the context.
- Data may need to be manually entered or scanned using methods such as Magnetic Ink Character Recognition (MICR) which is used with cheques.

Selecting and Managing Data

- Selecting the correct data is an important part of data preprocessing.
- This could involve only selecting data that fits a certain criteria.
- Collected data can be managed using SQL to sort, restructure and select certain sections.

Exchanging Data

- Exchanging data is the process of transferring the data that has been collected.
- One common example of this is EDI (Electronic Data Interchange).

SQL

- SQL stands for Structured Query Language and is a declarative language used to manipulate databases

Movie				
MovieID	**MovieTitle**	**MovieCompany**	**DatePublished**	**DirectorName**
M0001	Howdy Partner!	Cowboys Inc	04/21/2001	James
M0002	Okay Samantha, just leave.	Sadboys Inc	04/21/2001	Joseph
M0003	Bye Bye Bucky.	Cowboys Inc	05/12/2004	Jeremy
M0004	My wife left me for my dog...	Sadboys Inc	05/14/2004	James
M0005	Cars, Girls, and Money.	Rappers ltd	06/21/2012	James
M0006	Water Bottle Sadness	Sadboys Inc	08/12/2015	Jeremy

SELECT, FROM, WHERE

- The SELECT statement is used to collect fields from a given table.
- The FROM statement specifies which table/tables the information will come from.
- The WHERE statement specifies the search criteria.

For example:

```
SELECT  MovieTitle, DatePublished
FROM Movie
WHERE DatePublished BETWEEN #01/01/2000# AND #31/12/2005#
ORDER BY DatePublished
```

This will produce the following result:

MovieTitle	DatePublished
Howdy Partner!	04/21/2001
Okay Samantha, just leave.	04/21/2001
Bye Bye Bucky.	05/12/2004
My wife left me for my dog...	05/14/2004
Cars, Girls, and Money.	06/21/2012
Water Bottle Sadness	08/12/2015

ORDER BY

- The ORDER BY part of the code specifies whether you want it in ascending or descending order.
- The example below orders selected data in descending order:

```
ORDER BY DatePublished Desc
```

JOIN

- JOIN provides a method of combining rows from multiple tables based on a common field between them.
- The example below shows the joining of two tables, Movies and Directors.

```
SELECT Movie.MovieTitle, Director.DirectorName, Movie.MovieCompany
FROM Movie
JOIN Director
ON Movie.DirectorName = Director.DirectorName
```

CREATE

- The CREATE function allows you to make new databases, as shown below:

```
CREATE TABLE TableName
(
Attribute1          INTEGER NOT NULL, PRIMARY KEY,
Attribute2          VARCHAR(20) NOT NULL,
…
)
```

- You need to specify a few details for each attribute:
  - If it is the primary key,
  - Its data type,
  - Whether it needs to be filled in

ALTER

- This is used to add, delete or modify the columns in a table

Adding a column:

```
ALTER TABLE TableName
ADD AttributeX and their dataTypes
```

Deleting a column:

```
ALTER TABLE TableName
DROP COLUMN AttributeX
```

Modifying the datatype of a column:

```
ALTER TABLE TableName
MODIFY COLUMN AttributeX  NewDataType
```

INSERT INTO

- This is used to insert a new record in a table.

For example:

```
INSERT INTO (column1, column2, …)
VALUES (value1, value2, …)
```

UPDATE

- This is used to update a record in a table.

For example:

```
UPDATE TableName
```

```
SET column1 = value1, column2 = value2 …
Where columnX = value
```

DELETE

- This is used to delete a record from a database table.

For example:

```
DELETE FROM TableName
WHERE columnX = value
```

Referential Integrity

- Referential integrity is the process ensuring consistency.
- This makes sure that information isn't removed if it is required elsewhere in a linked database.

Transaction Processing

- A transaction is defined as a single operation executed on data.
- Transactions must be processed in line with ACID.

ACID(Atomicity, Consistency, Isolation, Durability)

Atomicity:

- A transaction must be processed in its entirety or not at all.

Consistency:

- A transaction must keep the referential integrity rules between linked tables.

Isolation:

- Simultaneous execution of transactions must lead to the same result as if they were executed one after the other.

Durability

- Once a transaction has been executed it will remain so.

Record Locking

- The process of preventing simultaneous access of records in a database.
- This is used to prevent inconsistencies or a loss of updates.
- If anyone tries to access the same record they will not be able to.
- The biggest problem with this is deadlock.

Redundancy

- The process of having one or more copies of the data in physically different locations.
- This means that if there is any damage to one copy the others can be recovered.