Chapter 8 Databases

8.1 Database Concepts

8 Databases	
8.1 Database Concepts	
Candidates should be able to:	Notes and guidance
Show understanding of the limitations of using a file-based approach for the storage and retrieval of data	
Describe the features of a relational database that address the limitations of a file-based approach	
Show understanding of and use the terminology associated with a relational database model	Including entity, table, record, field, tuple, attribute, primary key, candidate key, secondary key, foreign key, relationship (one-to-many, one-to-one, many-to-many), referential integrity, indexing
Use an entity-relationship (E-R) diagram to document a database design	
Show understanding of the normalisation process	First Normal Form(1NF), Second Normal Form (2NF) and Third Normal Form (3NF)
Explain why a given set of database tables are, or are not, in 3NF	
Produce a normalised database design for a description of a database, a given set of data, or a given set of tables	

Limitations with file-based approach as compared to a relational database

- Data redundancy – data is repeated, in different files
- No data integrity – might be changed by one program but not other, reduces consistency
- Data less secure
- Complex queries cannot be performed
- Data not independent of the program

Why database should be upgraded from file-based approach to relational database

- To reduce data redundancy, File based approach has data redundancy issues, data is repeated in different files, repetition of data. In a relational data base, data in tables in database are linked together using foreign keys.

- Improved data integrity, Data is stored in separate tables in a file based approach, it has inconsistency in data, data might be changed by one application but not by other, in relational data base, data is only entered once and is related, so any changes made by one application would mean that the data would be changed for other applications as well.

- Data dependency is overcome, File based approach also has data dependency issues, in relational database data is not depended on the database, data structure can used independent of application and query program. structure of data can change and doesn't affect program. Data can be accessed by any program

- Referential integrity can be enforced

- More complex searches and queries can be executed.

- Privacy is improved

- Linked tables can be set up

- Multiple tables are linked together
    - It helps to reduce data redundancy, as data is not repeated, it's only stored once
    - It improves data integriity
    - Data only stored once, so if it's changed by one application it's changed for the other applications accessing it as well

- Improved security using access rights

Database terminology

Entity - anything that can have data stored about it, for example, a person, place, event, thing.
Table - a group of similar data, in a database, with rows for each instance of an entity and columns for each attribute.
Record - a row in a table in a database.
Field - a column in a table in a database.
Tuple - one instance of an entity, which is represented by a row in a table.
Attribute - an individual data item stored for an entity, for example, for a person, attributes could include name, address, date of birth.
Primary key - a unique identifier for a table. It is a special case of a candidate key.
Candidate key - an attribute or smallest set of attributes in a table where no tuple has the same value.
Secondary key - a candidate key that is an alternative to the primary key.
Foreign key - a set of attributes in one table that refer to the primary key in another table.
Relationship (one-to-many, one-to-one, many-to many) - situation in which one table in a database has a foreign key that refers to a primary key in another table in the database.
Referential integrity

- Referential integrity is making sure that the data that is being referenced actually exists.
- The foreign key that is used to refer to a primary key, must be valid, primary key must exist.
- Primary key and foreign key must have same data type
- Every foreign key value has a matching value in the corresponding primary key
- Cascading delete
    - If a record is deleted in primary table
    - All corresponding linked records must also be deleted
- Cascading update
    - If a record is modified in primary table
    - All linked records must be modified as well

Indexing - a data structure built from one or more columns in a database table to speed up searching for data

Database has a secondary key describe reason for this?

- Database would frequently want to search using secondary key, e.g. contact number
- Allows for faster search using contact number.
-

E-R Diagrams

A graphical representation of a database and the relationships between the entities. Relationships may be mandatory or optional. For example, in a workroom with desks, each employee has one desk, but there could be spare desks. The relationship between desk and employee is zero or one, so this relationship is optional. The relationship between mother and child is mandatory because every mother must have at least one child, so the relationship is one or many. The type of relationship and whether it is mandatory or optional gives the cardinality of the relationship. (***To properly understand E-R Diagrams, do a handful of past papers for E-R Diagrams)***

Normalisation

Normalisation (database) – the process of organising data to be stored in a database into two or more tables and relationships between the tables, so that data redundancy is minimised.

First normal form (1NF) – the status of a relational database in which entities do not contain repeated groups of attributes.

Second normal form (2NF) – the status of a relational database in which entities are in 1NF and any non-key attributes depend upon the primary key. No primitive dependency.

Third normal form (3NF) – the status of a relational database in which entities are in 2NF and all non-key attributes are independent. No transitive dependencies.

Why database is fully normalised?

- No repeating attributes 1nf
- No partial dependency 2nf
- No non-key or transitive dependency 3nf

8.2 Database Management System (DBMS)

8.2 Database Management System (DBMS)	
Candidates should be able to:	Notes and guidance
Show understanding of the features provided by a Database Management System (DBMS) that address the issues of a file based approach	Including: • data management, including maintaining a data dictionary • data modelling • logical schema • data integrity • data security, including backup procedures and the use of access rights to individuals / groups of users
Show understanding of how software tools found within a DBMS are used in practice	Including the use and purpose of: • developer interface • query processor

Data management
- the organization and maintenance of data in a database to provide the information required.

Data Dictionary
- a set of data that contains metadata (data about other data) for a database.

Items in a data dictionary
- Validation
- Tables
- Data types
- Primary keys
- Attribute names

Data Modelling
- Data modelling is an important tool used to show the data structure of a database. A
- n E-R diagram is an example of a data model.
- A logical schema is a data model for a specific database that is independent of the DBMS used to build the database.

Security Features in DBMS

- Access Rights
  - Restricts actions of specific users
- Encryption
  - Makes the data incomprehensible
- Auto-backup
  - Creates regular copies of data in case of loss
- Passwords
  - Prevents unauthorized access
- Views
  - Restrict which parts of database specifc users can see

Schema of a database

External

- Individual view of the database

Conceptual

- Describes the views the database users might have

Physical

- Describes how data will be stored on the physical media

Logical

- Describes how relationships will be implemented in the logical structure of the database
- the overview of a database structure
- models the problem / situation
- ... by using methods such as an ER diagram
- independent of any particular DBMS

Developer interface

- Reports can be made to output data, e.g marks of a student
- Forms can be set up to input data
- Objects such as drop down can be used to make input easier
- A menu can be added to select options for different actions
- Create a table
- Create reports
- Create forms
- Create relationship between tables
- Create queries
- To create user friendly features e.g. forms
- To create outputs e.g. reports for a given date
- To create interactive features e.g. dropdowns

Query Processor

- To create SQL queries
- To search for data based on set criteria
- To perform calculations on extracted data
- searches for the data that meets the entered criteria
- organizes the results to be displayed to the user

8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)

8.3 Data Definition Language (DDL) and Data Manipulation Language (DML)	
Candidates should be able to:	Notes and guidance
Show understanding that DBMS carries out all creation / modification of the database structure using its Data Definition Language (DDL)	
Show understanding that the DBMS carries out all queries and maintenance of data using its DML	
Show understanding that the industry standard for both DDL and DML is Structured Query Language (SQL)	Understand a given SQL script
Understand given SQL (DDL) commands and be able to write simple SQL (DDL) commands using a sub-set of commands	Create a database (CREATE DATABASE) Create a table definition (CREATE TABLE), including the creation of attributes with appropriate data types: • CHARACTER • VARCHAR(n) • BOOLEAN • INTEGER • REAL • DATE • TIME change a table definition (ALTER TABLE) add a primary key to a table (PRIMARY KEY (field)) add a foreign key to a table (FOREIGN KEY (field) REFERENCES Table (Field))
Write an SQL script to query or modify data (DML) which are stored in (at most two) database tables	Queries including SELECT... FROM, WHERE, ORDER BY, GROUP BY, INNER JOIN, SUM, COUNT, AVG Data maintenance including. INSERT INTO, DELETE FROM, UPDATE

Structured Query Language

The standard query language used with relational databases for data definition and data modification

Data Definition Language (DDL):

Creating a database:

```
CREATE DATABASE SAMPLE
```

Creating a table:

```
CREATE TABLE SAMPLE_DATA ( EmployeID VarChar(7)
NOT NULL, FirstName VarChar, DateOfBirth Date,
Stock Integer, Price Real, EntryTime Time, Alive
Boolean, PRIMARY KEY (EmployeID)   FOREIGN KEY
(ClassID) REFERENCES CLASS (ClassID) );
```

Adding a new field:

```
ALTER TABLE SAMPLE_DATA ADD Salary Integer;
```

Data Manipulation Language(DML):

Adding the data into the table

When inserting into all columns in logical order (left to right)

```
INSERT INTO SAMPLE_DATA
VALUES (<data in coulumn 1>, (<data in coulumn 1>,….);
VALUES (5, 'Double', #01/01/2023#);
```

When inserting into only some columns

```
INSERT INTO SAMPLE_DATA (Stock, Type) VALUES (5,
'Double');
```

Updating data in the table:

```
UPDATE SAMPLE_DATA
SET <column 1> = <value 1>, <column 2> = <value 2>
SET Stock = 5, Type = 'Double'  WHERE ID = '6A';
```

DELETE statement is used to delete existing records in a table:

```
DELETE FROM SAMPLE_DATA WHERE ID = '6A';
```

Display/return a value

```
SELECT table1.<column 1>, table2.<column 2> FROM
SAMPLE_DATA   WHERE   table.ID   =   `6A';   AND
table2.Price >= 50 GROUP BY Customer ID ORDER BY
Time DESC GROUP BY Customer ID ORDER BY Time ASC;
```

Count, Average and Sum statements

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Write the SQL statement to find the number of products

```
SELECT COUNT (ProductID) FROM PRODUCTS;
```

Write the SQL statement to find the average price of all the products

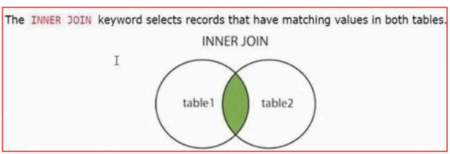
```
SELECT AVG (Price) FROM PRODUCTS;
```

Write the SQL statement to find the sum of all the prices

SELECT SUM (Price) FROM PRODUCTS;

Inner Join

The INNER JOIN keyword selects records that have matching values in both tables.




```
SELECT <column name>  FROM <table 1> INNER JOIN
<table 2> ON table1.column name = table2.column
name;  WHERE ID = `6A'
```

Or simpler method:

```
SELECT <column name>  FROM <table1>, <table2>
WHERE table1.column name = table2.column name
AND ID = `6A'
```

Comparing to current date

Any date greater than today's date condition:

```
WHERE XDate > #######;
```

Selection of specific data

Q: Select all records where the value of the City column starts with 'a'

```
WHERE City LIKE 'a%';
```

Q: Select all records where the value of the City column ends with 'a'

```
WHERE City LIKE '%a';
```

Q: Select all records where the value of the City column contains letter 'a'

```
WHERE City LIKE '%a%';
```

Q: Select all records where the value of the City column starts with 'a' and ends with letter 'b'

```
WHERE City LIKE 'a%b';
```

Q: Select all records where the value of the City column does not start with 'a'

```
WHERE City NOT LIKE 'a%';
```

Q: Select all records where the second letter of the City is an 'a'

```
WHERE City LIKE '_a%';
```

Q: Select all records where the first letter of the City is an 'a' or 'c' or 's'

```
WHERE City LIKE  '[acs]%';
```