13 Databases and Software Development

13.1 Entity Relationship Modelling

An **entity** is a category of object about which data is to be recorded.

- Each entity in a database system has **attributes**.
- Each entity needs an **entity identifier** which uniquely identifies the entity.
- The **primary key** is the entity identifier in a **relational database**.

An **entity description** is written using the format.

EntityName (PrimaryKey, Attribute1, Attribute2)

Two entities are said to be **related** if they are linked in some way.

The **degrees of relationship** between two entities can be

- One-to-one
- One-to-many
- Many-to-many

An **entity relationship diagram** is a diagrammatic way of representing the relationship between the entities in a database. Shows the

- Degree of relationship.
- Name of the relationship.

Relational Database

In a relational database, a separate **table** is created for each entity identified in the system.

- Where a relationship exists between entities, an extra field called a **foreign key** links the two tables.
- A foreign key is an attribute that creates a join between two tables - it is the attribute that is **common to both tables**.
- The primary key in one table is the foreign key in the table to which it is linked.

Many-to-many Relationships

Tables cannot be linked directly in a many-to-many relationship. Instead, create a **link table** with two foreign keys, each linking to one of the two tables. The two foreign keys also act as the primary key of the table.

A primary key which consists of more than one attribute is called a **composite primary key**.

13.2 Relational Databases and Normalisation

A **relational database** is a collection of tables which relationships are modelled by shared attributes.

Tables can be linked through the use of **common attributes**. This attribute must be a primary key of one of the tables, and is know as a **foreign key** in the second table.

Normalisation

Normalisation is a process used to come up with the **best possible design** for a relational database. Tables are organised in a way that

- No data is **unnecessarily duplicated**.
- Data is **consistent** throughout the database. This means anomalies will not arise when the data is inserted, amended or deleted.

  Consistency should be an automatic consequence of no holding any duplicated data.
- The structure of each table is enough to allow you to enter **as many or as few items** as required.
- The structure should enable a sure to make all kinds of **complex queries** relating data from different tables.

1. A table is in **first normal form** if it contains **no repeating attribute** or group of attributes.
2. A table is in **second normal form** if it is in first normal form and contains **no partial dependencies**.

   A partial dependency is where one or more attributes **depends on only part of the primary key**, which can only occur if the primary key is a **composite key**.
3. A table is in **third normal form** if it is in second normal form and contains **no non-key dependencies**.

   A non-key dependency is one where the value of an attribute is determined by the value of another attribute which is not part of the key.

Advantages of Normalisation

- Easier to **maintain and change**.

Data integrity is maintained since there is no unnecessary duplication of data, it will also be impossible to reference a non-existing record on another table.

- **Faster** sorting and searching, as normalisation produce **smaller tables with fewer fields**, searching is faster because less data is involved.

  Holding data once saves storage space.

- A normalised data base with **correctly defined relationships** between tables will not allow records in a table on the 'one' side of a one-to-many relationship to be deleted accidentally.

13.3 Introduction to SQL

Structured query language (SQL) is a **declarative language** used for querying and updating tables in a relational database.

SELECT FROM WHERE

```
SELECT      -- List of fields to be displayed
FROM        -- List of tables where the data come from
WHERE       -- List of search criteria
```

ORDER BY

```
ORDER BY    Field1, Field2    ASC/DESC
```

13.4 Defining and Updating Tables Using SQL

CREATE TABLE

Create a new database table.

```
CREATE TABLE TableName (
    Field1     INTEGER NOT NULL PRIMARY KEY,
    Field2     VARCHAR(20) NOT NULL,
    Field3     DATE
)
```

Data types include

Data type	Description
`CHAR(n)`	Character string of fixed length n
`VARCHAR(n)`	Character string of variable length, max n
`BOOLEAN`	True or false
`INTEGER`	Integer
`FLOAT`	Floating point number
`DATE`	Day, month and year values
`TIME`	Hour, minute and second values
`CURRENCY`	Format numbers in currency used in your region

ALTER TABLE

Add, delete or modify columns in an existing table.

```
ALTER TABLE TableName
-- Operation, such as
ADD    FieldName   VARCHAR(10)           -- or
DROP   COLUMN      ColumnName            -- or
MODIFY COLUMN      ColumnName VARCHAR(30) NOT NULL
```

Defining Link Table

```
CREATE TABLE TableName (
    Field1      CHAR(4) NOT NULL,
    Field2      INTEGER NOT NULL,

    FOREIGN KEY Field1 REFERENCES Table1(Field1),
    FOREIGN KEY Field2 REFERENCES Table2(Field2),
    PRIMARY KEY (Field1, Field2)
)
```

INSERT INTO

```
INSERT INTO TableName(Field1, Field2)
VALUES (123, 456)
```

UPDATE

```
UPDATE TableName
SET Field1 = 123, Field2 = 456
WHERE Field1 = 987
```

DELETE

```
DELETE FROM TableName
WHERE Field1 = 123
```

Client-server Databases

Database Management System (DBMS) provides an option for client-server operation.

- DBMS **server software** runs on the network server.
- DBMS **client software** runs on individual workstations.

The server software **processes requests** for data searches and reports that originate from individual work stations running DBMS client software.

If the DBMS does not have client-server capability, the entire database would be **copied to the workstation** and software held on the workstation would search for the requested data.

- A large amount of time is being spent of transmitting irrelevant data.
- A longer search with a less powerful machine.

Advantages of client-server databases

- The **consistency** of the database is maintained because only one copy of the data is held.
- An expensive resource can be made **available to large number of users**.
- **Access rights** and security is managed and controlled centrally.
- **Backup and recovery** can be managed centrally.

Problems with Client-server Databases

Allowing multiple users to **simultaneously update a database table** may cause one of the updates to be lost.

1. When an item is updated, the entire record is **copied to to the user's local memory** at the work station.
2. When the record is saved, the record is rewritten to the server.

Record Locks

Record locking is a technique of **preventing simultaneous access to objects** in a database in order to prevent updates being lost or inconsistencies in the data arising.

- A record is locked whenever a user retrieves it for editing or updating.
- Anyone else attempting to retrieve the same record is **denied access** until the transaction is completed or cancelled.

If two users are attempting to update two records, a situation can arise in which **neither can proceed**, known as **deadlock**.

- **Serialisation** is a technique which ensures the **transactions do not overlap in time** and therefore cannot interfere with each other or lead to updates being lost. A transaction cannot start until the previous one has finished.

  Can be implemented using **timestamp ordering**.

- **Timestamp ordering** - whenever a transaction starts, it is given a timestamp, so if two transactions affect the same object, the transaction with the **earlier timestamp is applied first**.

  To ensure that transactions are not lost

  1. Each object in the database has a **read timestamp and a write timestamp**, which are updated whenever an object in a database is read or written.
  2. When a transaction starts, it reads the data from a record causing the **Read timestamp to be set**.
  3. Before it writes the updated data back to the record, it will **check the read timestamp**.
  4. If this is **not the same** as the value that was saved when this transaction started, it will know that another transaction is also taking place on the record.

  The problem can be identified and avoided.

- **Commitment ordering** is a serialisation technique used to ensure that **transactions are not lost** when two or more users are simultaneously trying to access the same database object.
  - Transactions are ordered in terms of their **dependencies on each other** as well as the time they were initiated.
  - Deadlocks can be prevented by **blocking one request** until another is completed.