Learning Unit 1

CHECK POINT 1

1. A relation is a table in a relational database. It is represented as a list of fields following the table name and the field that is the primary key is underlined. tableName(primary_key field, field1, field2, field3, ..fieldn)

2. Row is a record, entity is a record, attribute is a field, record is a collection of fields, table is a collection of records, column is a field and relation is a table.

3. Deleting a record is removing an entire row of data relating to one entity. Deleting an attribute is removing an entire column. This means that each record in the database loses one of its fields. The seriousness depends on the data deleted.

4. Pronounce skee-ma, the structure of a database system, described in a formal language supported by the database management system (DBMS). In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables.

5. Super key - A combination of fields in a table which uniquely identify each row. In general a super key is any number of fields that will uniquely identify a record.

   Candidate key - Candidate keys are minimal super-keys. In other words there must be no attributes included in the examples above which don’t contribute to the uniqueness of the rows.

   Primary key - A minimal candidate key that uniquely identifies a record.

6. clientID and carName. This assume no client has two cars with the same car type.

7. You could combine all three fields name, class and house but that would assume that you cannot have two people with the same name and class in the same house. This is unlikely but not impossible.

CHECK POINT 2

1. Normalisation is a technique for designing relational database tables to minimize duplication of information and safeguard the database against data anomalies.

2. A repeating group is where one field contains many values. Data redundancy is a result of representing a repeating group in first normal form.

3. Update anomalies occur when we need to update the same data in more than one place. Deletion anomalies occur when a deletion causes loss of data unnecessarily. Insertion anomalies occurs when we add a record that does not satisfy the design or the primary key requirements.

4. Update anomaly – If a person’s email address changes then it must be changed in more than one place.

   Deletion anomaly – If we delete John Doe we will lose information about the tank.

   Insertion anomaly – a new person cannot be added if they do not have a car yet.

5. 

   5.1. Repeating group is Donations

   5.2. Purchases

   5.3. Ingredients and Recipes
6. Each person's name and email address is repeated for each sale.

CHECK POINT 3

1. When a field has a relationship with another field. Such as a student ID can be used to obtain the student's name and address.

2. Derived data is calculated from existing fields. Data dependency is where one field is linked to another but cannot be calculated.

3. A partial dependency is when a field's value is dependent on only part of a composite key. A transitive dependency is when a field's value is dependent on a non-key field.

4. Redundant data is produced as a result of first normal form and repeating groups. Duplicate data is data where one field has the same value as another for example two people living in the same city. Both are valid and the data must not be eliminated.

5. Atomic data field should contain a single data item. For example the name and surname fields should be stored as separate fields.

6. 1NF
   - Flattened the table
   - Choose a primary key

2NF
   - Normalise to 1NF
   - Remove partial dependencies

3NF
   - Normalise to 1NF
   - Remove transitive dependencies

EXERCISE 1

1.

1.1.

1.1.1. No.

1.1.2. There are repeating groups and no primary key.

1.1.3. Project (ProjID, ProjName, EmpID, EmpName, JobTitle, ChargePerHour, HoursWorked)

1.2.

1.2.1.
1.2.2. **Project** (ProjID, ProjName, EmpID, EmpName, JobTitle, ChargePerHour, HoursWorked)

1.2.3. Each employee works on many projects. By combining the **ProjID** (which uniquely identifies a project) and the **EmpID** (which uniquely identifies an employee), we can create a field that will uniquely identify each record.

1.3.

1.3.1. Insert, a project cannot be added if an employee is not allocated to the project.

Update, If a project changes name, it must do so in multiple places.

Delete, if we delete the Sanlam project, we will lose all information about Janet Jackson.

1.3.2. The details of the projects are redundant, the details about the employees and their job titles together with charge per hour.

1.4.

1.4.1. Relations are in 1 NF and no partial dependencies.

1.4.2. Relations are in 1 NF and no partial dependencies.

1.4.3. **Employee** (EmpID, EmpName, JobTitle, ChargePerHour)

**Project** (ProjID, ProjName)

**EmpProj** (EmpID, ProjID, HoursWorked)

1.5.

1.5.1. Relations are in 2 NF and no transitive dependencies.

1.5.2. See 1.4.2

1.5.3. **Employee** (EmpID, EmpName, JobTitle)

**Project** (ProjID, ProjName)

**EmpProj** (EmpID, ProjID, HoursWorked)

**Jobs** (JobTitle, ChargePerHour)

1.6. What happens if we need to change a JobTitle? We need to change in more than one place! Update Anomaly.

**Employee** (EmpID, EmpName, JobID)

**Project** (ProjID, ProjName)

**EmpProj** (EmpID, ProjID, HoursWorked)

**Jobs** (JobID, JobTitle, ChargePerHour)
2.

2.1.

<table>
<thead>
<tr>
<th>Matric_no</th>
<th>Name</th>
<th>Date_of_birth</th>
<th>Subject</th>
<th>Teacher</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>960100</td>
<td>Smith, J</td>
<td>14/11/1977</td>
<td>Databases</td>
<td>Jobs</td>
<td>C</td>
</tr>
<tr>
<td>960100</td>
<td>Smith, J</td>
<td>14/11/1977</td>
<td>Soft_Dev</td>
<td>Gates</td>
<td>A</td>
</tr>
<tr>
<td>960100</td>
<td>Smith, J</td>
<td>14/11/1977</td>
<td>ISDE</td>
<td>Torvald</td>
<td>D</td>
</tr>
<tr>
<td>960105</td>
<td>White, A</td>
<td>10/05/1975</td>
<td>Soft_Dev</td>
<td>Gates</td>
<td>B</td>
</tr>
<tr>
<td>960105</td>
<td>White, A</td>
<td>10/05/1975</td>
<td>ISDE</td>
<td>Torvald</td>
<td>B</td>
</tr>
<tr>
<td>960120</td>
<td>Moore, T</td>
<td>11/03/1970</td>
<td>Databases</td>
<td>Jobs</td>
<td>A</td>
</tr>
<tr>
<td>960120</td>
<td>Moore, T</td>
<td>11/03/1970</td>
<td>Soft_Dev</td>
<td>Gates</td>
<td>B</td>
</tr>
<tr>
<td>960120</td>
<td>Moore, T</td>
<td>11/03/1970</td>
<td>Workshop</td>
<td>Lovelace</td>
<td>C</td>
</tr>
<tr>
<td>960145</td>
<td>Smith, P</td>
<td>09/01/1972</td>
<td>Databases</td>
<td>Jobs</td>
<td>B</td>
</tr>
<tr>
<td>960150</td>
<td>Black, D</td>
<td>21/08/1973</td>
<td>Databases</td>
<td>Jobs</td>
<td>B</td>
</tr>
<tr>
<td>960150</td>
<td>Black, D</td>
<td>21/08/1973</td>
<td>Soft_Dev</td>
<td>Gates</td>
<td>D</td>
</tr>
<tr>
<td>960150</td>
<td>Black, D</td>
<td>21/08/1973</td>
<td>ISDE</td>
<td>Torvald</td>
<td>C</td>
</tr>
<tr>
<td>960150</td>
<td>Black, D</td>
<td>21/08/1973</td>
<td>Workshop</td>
<td>Lovelace</td>
<td>D</td>
</tr>
</tbody>
</table>

2.2. Insert, a student cannot be added if they have not taken a subject.

Update, If a subject changes name, it must do so in multiple places.

Delete, if we remove the subject Databases, we will lose all information about P Smith.

2.3.

```
EXERCISE 2

1.

1.1. ParentIDNumber as each person has more than one parent.

1.2. Person (IDNumber, Name, Age, BirthPlace, ParentIDNumber)
```

```
**Parent (ParentIDNumber, IDNumber)**

2.

2.1. Address

2.2. Manufacturer and category are redundant data. duplicate data is the price and product name.

2.3. **Manufacturer** (Manufacturer, Address, CategoryID)

   **Product** (Product Name, Description, Price)

   **Category** (CategoryID, Category name)

3.

3.1. One doctor can have many patients and one patient can have many doctors so the patient to doctor relationship is many to many.

3.2. **Doctor** (DoctorID, Doctor name)

   **Patient** (Patient name)

   **DocPat** (DoctorID, Patient name)

4.

4.1. One prison can have more than one prisoner and one prisoner can be in more than one prison which is a many to many relationship.

4.2. **Prison** (Prison Name)

   **Prisoner** (Prisoner Name, start date, release date)

   **PrisonerPrison** (Prison Name, Prisoner Name)

4.3. Already indicateds in the relations above.

---

**EXERCISE 3**

1. **Donor** (Donor)

   **Project** (ProjectName, Description, Amount, Donor)

2.

2.1. No primary key and repeating data.

2.2. Eliminate repeating data and choose primary key.

2.3. Choose concatenated primary key Outfit and Accessory.

2.4. Minnie mouse is data about the same outfit whereas black describes different accessories.

2.5. Insert – add a record that violated the primary key
2.6. Update – edit a record in more than one place
   Delete – delete a record and lose relevant data
   Update

2.7. If Princess deleted the Fairy outfit she would lose data about her wings

2.8. She would need an outfit to put them with otherwise it would violate the primary key.

2.9. 1NF and no transitive dependencies

2.10.

<table>
<thead>
<tr>
<th>Outfit</th>
<th>Description of Outfit</th>
<th>Warm or Cool Weather</th>
<th>Accessory</th>
<th>Accessory Colour</th>
<th>Does it go?</th>
</tr>
</thead>
</table>

2.11. Outfit (Outfit, Description of Outfit)
      Accessory (Accessory, Accessory Colour)
      OutfitAcc (Outfit, Accessory, Does it go?)

2.12. In 2NF and no transitive dependencies

2.13. There are no transitive dependencies, no non key field depends on another.

2.14. No, because she could use the same outfit next year which would violate the primary key

---

**EXERCISE 4**

1. Referential integrity is where a record cannot refer to a record that does not exist.

2. An runner cannot be deleted if runners still have results for the race. A runner cannot be made the captain of a school if they do not exist in the database. All race results must be for existing runners. All runners must exist before they can be part of a running school.