Chapter 8 Database

Course Material

Cambridge International AS & A-level 9618 syllabus

8.1 Database Concept

Key terms

- **Entity**: anything that data can be stored as tables like person, event or things.
- **Table**: a group of data with rows and columns.
- **Record/Tuple**: a row in a table in a database.
- **Field/ Attribut**e: a column in a table in a database.
- **Primary key**: an attribute or a set of attributes that have unique value to identify each record. (a special case of candidate key)
- **Foreign key**: an attribute or a set of attributes in one table that refer to the primary key in another table.
- **Candidate key**: an attribute or a set of attributes that don't have the same value.
- **Referential integrity**: property of database that all foreign keys in this database are valid.
- **Index**: a data structure built from one or more columns in a table to speed up searching for data.
- **Relationship**: situation in which one table in a database has a foreign key that refers to a primary key in another table.

File-based approach vs relational database

File-based approach	Relational database
A single table	Several tables
Simple technology for a small amount of data	Complex technology for a large amount of data
Data redundancy issues	/
Data privacy issues	/
Data-program dependency issues	/

Entity - Relationship (E-R) Diagram

- Displays the **relationship of entity** sets stored in a database
- Three basic concepts: **entities**, **attributes** and **relationships**.

Example - Shops (Q1, Answer)

Schema / Relationship Notation

- SHOP(ShopID, ShopName, Location, RetailSpecialism)
- SUPPLIER(SupplierID, SupplierName, ContactPerson, RetailSpecialism)
- SHOP-SUPPLIER(ShopID, SupplierID)

Relationship

- One to many: shop to shop-supplier
- Many to many : shop to supplier
- One to one : (husband to wife)

Normalisation Process

- To minimise redundancy
- To solve data privacy issues
- To make data-program independent
- Extended resource - Example of normalisation with SQL statement

1NF

- No repeated groups of attributes
- All attributes should be atomic
- No duplicate rows

2NF

- Everything from 1NF

- No partial dependencies

3NF

- Everything from 2NF
- all attributes should be fully dependent on primary key
- or non-key dependencies / No transitive dependencies

8.2 DBMS

Key Term

- **Data dictionary**: Metadata including the primary key, attributes, data type, validation.
- **Data modelling**: the analysis and definition of the data structures required in a database.
- **Logical schema:** a data model that is used to build a database.
- **Data Integrity:** the maintenance of data accuracy, completeness, and consistency.
- **Data security:** Methods taken to prevent unauthorised access to data and to recover data if lost or corrupted.
- **Developer Interface:** Feature of a DBMS that provides code editor and code running.
- **Query processor:** Feature of a DBMS that processes and executes queries written in SQL.

8.3 DDL and DML

- Data Definition Language
- Data Manipulation Language

DDL

- To create, modify and remove the **data structure**.
- To produce **data dictionary**

Data Type

Data type	Description
CHARACTER	Fixed length text
VARCHAR	Variable length text

BOOLEAN	True or False
INTEGER	Whole number
REAL	Number with decimal places
DATE	A data usually formatted as YYYY-MM-DD
TIME	A time usually formatted as HH:MM:SS

Commands and Scripts (SQL)

```
CREATE DATABASE School
CREATE TABLE Class(
ClassID CHARACTER,
Location CHARACTER,
Licence Number VARCHAR(12));

ALTER TABLE Class ADD PRIMARY KEY (ClassID)

CREATE TABLE Student(
      StudentID VARCHAR(6),
      FirstName CHARACTER,
      SecondName CHARACTER,
      DateOfBirth DATE,
      ClassID CHARACTER,
PRIMARY KEY (StudentID)
FOREIGN KEY (ClassID) REFERENCES Class(ClassID));
```

DML

- To add, modify, delete and retrieve the data stored in a relational database

Commands and scripts (running online)

SELECT, WHERE, ORDER BY

```
SELECT CustomerName, Address
FROM Customers
```

```
WHERE Country='Mexico'
ORDER BY CustomerName;
```

SELECT2

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders, Customers
WHERE Orders.CustomerID = Customers.CustomerID;
```

OR

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID = Customers.CustomerID;
```

INSERT

```
INSERT INTO Customers
VALUES (92, 'Alfredsson', 'Maria Anders', 'Obere Str. 57',  'Berlin',
'12209', 'Germany')
```

OR

```
INSERT INTO Customers (CustomerID,CustomerName,ContactName, Address,
City, PostalCode, Country)
VALUES (96,'Alfred S',  'Maria Anders','Obere Str. 57','Berlin',
'12209','Germany')
```

DELETE

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

UPDATE

```sql
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

GROUP BY

```sql
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

AVG

```sql
SELECT AVG(Price)
FROM Products;
```

SUM

```sql
SELECT SUM(Quantity)
FROM OrderDetails;
```