Introduction to Databases and Data Modelling

Oliver Gregory

2024-10-01

Activity 1

Define the following terms: data, database, DBMS, metadata, program-data independence, user view, and transaction-processing application.

1. **Data**: Data are known facts that can be recorded and have implicit meaning.
2. **Database**: A database is a collection of related (logically coherent) data built for a specific purpose.
3. **DBMS**: A Database Management System (DBMS) is a software system that allows users to define, create, maintain, and control access to the database.
4. **Metadata**: Metadata is data that describes other data. It provides information about the data that a database stores.
5. **Program-data independence**: Program-data independence is the separation of data structures from the programs that operate on the data. This should mean that changes in the structure of the database do not affect the applications that use it.
6. **User view**: A user view is a subset of the database that is relevant to a particular user group.
7. **Transaction-processing application**: A transaction-processing application includes concurrency controls to ensure that several users can access and update the database simultaneously without corrupting the data.

Discuss the main characteristics of the database approach and its differences from traditional file systems?

The database approach has several key characteristics that differentiate it from traditional file systems:

1. **Data independence**: The database approach separates the physical storage of data from the logical representation of data. This means that changes to the physical storage do not affect the logical representation of the data.

2. **Data sharing**: Databases allow multiple users to access and modify the data simultaneously. This is achieved through concurrency control mechanisms.
3. **Data integrity**: Databases enforce data integrity constraints to ensure that the data remains accurate and consistent.
4. **Data security**: Databases provide mechanisms to control access to the data and ensure that only authorised users can access the data.
5. **Data redundancy**: Databases reduce data redundancy by storing data in a centralised location and providing mechanisms to ensure that data is consistent.
6. **Data consistency**: Databases enforce consistency constraints to ensure that the data remains consistent and accurate.

This differs from traditional file systems, which do not provide these features. File systems typically store data in individual files and lack many of the mechanisms that databases provide to ensure data integrity, security, and consistency. File systems tend to have issues with data duplication, separation and isolation, and data is often stored in a format that is specific to the application that uses it.

What are the different types of database users and their main activities?

There are several types of database users, each with different roles and responsibilities:

1. **Database Stewards**: Database stewards are responsible for adminstering the organisation's data, applying policies and standards, and ensuring data quality.
2. **Database Administrators (DBAs)**: Database administrators are responsible for managing the database and DBMS, authorising access, monitoring and coordinating the use of the database, acquiring software and hardware resources, and ensuring security and performance.
3. **Database Designers**: Database designers are responsible for identifying data to be stored, defining data structures, eliciting requisites with database users and designing the database, including views.
4. **Database Users**: Database users are the end-users of the database. They interact with the database to retrieve, update, and manipulate data to perform their tasks.

Considering the figure, answer the following questions:

a. Cite examples of integrity constraints that you think can apply to this database.

Integrity constraints are rules that ensure the quality of the information that is stored. Examples of integrity constrains that could be applied to this database include (but are not limited to):

- Requiring unique values for 'Student_number', 'Course_code' and 'Section_identifier'.

- Requiring that a ‘Section’ may not exist with any NULL values.
- Requiring that a ‘Student’ may only have one ‘Major’.

b. If the name of the CS (Computer Science) Department changes to CCSE (Computer Science and Software Engineering) Department and the corresponding prefix for the course number also changes, what columns in the database would need to be updated? Can you restructure this database so that only one column is updated?

If the name of the CS Department changes to CCSE Department, and the corresponding prefix for the course number also changes, the following columns would need to be updated:

- Major (in the STUDENT table)
- Department (in the COURSE table)
- Course_number (in the COURSE, SECTION and PREREQUISITE tables)
- Prerequisite number (in the PREREQUISITE table)

To restructure the database so that only one column is updated, we could introduce a new table to store the department prefixes. This table would contain the department name and the corresponding prefix. The other tables would then reference this new table to determine the course prefix. This would allow us to update the prefix in one place and have it automatically propagate to all other tables.

Discuss the advantages and disadvantages of (i) centralised DBMS architectures, (ii) client-server architectures, and (iii) distributed databases.

Centralised DBMS architectures:

Centralised DBMS architectures have several advantages:

- **Simplicity**: Centralised architectures are simple to implement and manage.
- **Data consistency**: Centralised architectures ensure that data is consistent and accurate.
- **Data security**: Centralised architectures provide mechanisms to control access to the data and ensure that only authorised users can access the data.

However, centralised architectures also have several disadvantages:

- **Single point of failure**: Centralised architectures have a single point of failure. If the centralised server fails, the entire system is unavailable.
- **Scalability**: Centralised architectures can be difficult to scale as the system grows.
- **Performance**: Centralised architectures can suffer from performance issues as the system grows.

Client-server architectures:

Client-server architectures have several advantages:

- **Scalability**: Client-server architectures are more scalable than centralised architectures.
- **Performance**: Client-server architectures can provide better performance than centralised architectures.
- **Flexibility**: Client-server architectures are more flexible than centralised architectures.

However, client-server architectures also have several disadvantages:

- **Complexity**: Client-server architectures are more complex to implement and manage than centralised architectures.
- **Data consistency**: Client-server architectures can suffer from data consistency issues.
- **Data security**: Client-server architectures can be more difficult to secure than centralised architectures.

Distributed databases:

Distributed databases have several advantages:

- **Scalability**: Distributed databases are highly scalable.
- **Performance**: Distributed databases can provide excellent performance.
- **Fault tolerance**: Distributed databases are fault-tolerant.

However, distributed databases also have several disadvantages:

- **Complexity**: Distributed databases are complex to implement and manage.
- **Data consistency**: Distributed databases can suffer from data consistency issues.
- **Data security**: Distributed databases can be difficult to secure.