COMP3

Callum O'Brien

January 8, 2016

Contents

1 Databases & DBMS

A database is collection of non-redundant data that are stored and which the database management system is designed to manipulate. A database management system (DBMS) is a software system that enables the definition, creation and maintenance of a database and provides controlled access to this database.

In a *file-based approach*, these data are stored in a computer file called a flat file. In such a database, the only way to change the record structure of these files is to read the file of records using the old file structure and save the records to a new file using the new record stucture. The file-based approach has many disadvantages;

1. **Data inconsistency:** If there are two adresses for one person, the error in one is not 'spotted. '
2. **Data redundancy:** If someone rents a car more than once their details are stored many times
3. **Inefficiency:** Typing in the adresses and other details every time a car is rented is time-consuming and takes up storage space
4. **Difficult to maintain:** As more records are added the database may need to query and manipulate large amounts of data. This is more difficult using the file-based approach
5. **Program-data dependence:** Because the file structure has not been defined in an accessible way, every computer program has to specify exactly what data fields constitute a record in the file being processed. If the data structure changes, programs also have to be changed to match the new structure.

Relational databases provide an advantage over the file-based approach. Relational databases contain many *tables* that are linked together. This linking is done by *keys* – these are common pieces of information that are shared between tables. In a relational database, the DBMS creates and maintains the *data dictionary*. *Data definition language* is used to create tables. It records the attributes, data types, validation used and realtionship between entities. It defines which atttributes belong to which tables. It also creates users and grants access rights to them. DDL commands are;

1. **CREATE DATABASE**
2. **CREATE TABLE**
3. **CREATE USER**
4. **GRANT**
5. **DROP**

Data Manipulation Language is used to add data to the database and manipulate those data. DML commands are

1. **INSERT**
2. **UPDATE**

Database schema are different 'views' of a database. These can be

1. **External / User Schema:** the way in which each user sees the database, there may be several different schema representing each user's view.
2. **Conceptual or Logical Schema:** describes the entities, attributes and relationships
3. **Internal Schema:** describes how the data will be stored and how they will be accessed and updated.

1.1 Entity Relationships

Entities in a database are tables or users. In an *entity relationship diagram*, they are represented by rectangles. Relationships between entities are lines connecting entities. They can be;

1. One-to-one (e.g. $y = x$)
2. One-to-many (e.g. $y = \pm\sqrt{x}$)
3. Many-to-one (e.g. $y = \sin x$)
4. Many-to-many (e.g. $y^2 = 1 - x^2$)

1.2 Database Normalisation

Normalisation is a technique used to produce a normalised set of entities. This means that the entities will:

1. Contain no redundant data
2. Contain no repeated data
3. Be able to contain as many or as few tuples as we wish in an entity without requiring us to enter data in another entity as well.

The database will now be at its most efficient and there will be no risks of compromising the integrity of the data within it.

Databases that are not normalised to third normal form will have problems with insertion, deletion and updating anomalies. Database normalisation has been a well- developed field since the introduction of Codd's work on normal forms. While it's reasonable easy to identify repeating groups or attributes and remove these for first normal form, the differences between second and third normal form are difficult to secure.

There is a further complication in that repeating groups are understood in different ways by different theorists. As a consequence, there is not universal agreement as to what constitutes first normal form.

DDL	
CREATE DATABASE	Creates a new database
ALTER DATABASE	Modifies a database
CREATE TABLE	Creates a new table
ALTER TABLE	Modifies a table
DROP TABLE	Deletes a table
CREATE INDEX	Creates an index (search key)
DROP INDEX	Deletes an index
GRANT	Grants permissions on a table
DML	
UPDATE	Updates data in a database
DELETE	Deletes data from a database
INSERT INTO	Inserts new data into a database

Table 1: Common DDL and DML commands

1.3 Structured Query Language

SQL is a database language that allows users to create, store, update and manipulate data in a database. The first version of SQL was developed at IBM by Chamberlin and Boyce in the early 1970s. It was formally standardised in 1986 by ANSI.

Almost all modern relational DBMSs use SQL as a standard database language, but many use dialects of SQL.

SQL has two subsets, DML and DDL. DDL is used to create the database structure (i.e. to define which attributes belong in which tables). It also allows you to create users and grant access rights to users.

2 Hashing

A hash function maps from an input x to an output y in a manner requiring (relatively) little processing power, but such that y can not be mapped to x without significantly more processing. Hash functions are many-to-one; an ideal hash function has no inverse.

A collision occurs when x_n and x_m map to the same y. Collisions can be resolved by mapping x_m to the next available y. This approach can be ineffective, as it may ultimately lead to more collisions.

3 Programming Concepts

A paradigm is a typical example or pattern of something; a pattern or model.

4 Pointers

A pointer stores a memory adress rather than a variable. This can be advantageous as it allows dynamic memory allocation; as only the adress is stored in stack memory, the memory to store the actual value is in the heap.

In Pascal, pointers are declared using a caret (^). An example of this syntax is:

```
Var
    CounterPtr : ^Integer;
    AveragePtr : ^Real;
```

To then store a value in a variable;

```
A^ := 0;
```