Fundamentals of Databases notes

Todd Davies

December 22, 2014

Introduction

Databases are core, if largely invisible, components of modern computing architectures in both commercial and scientific contexts. The management of data has evolved from application-specific management of myriad files to organisation-wide approaches that see data as one of the most important assets of modern organisations and, as such, a key factor in their ability to compete and thrive. At this organisation-wide scale, database management systems (DBMSs) are the crucial piece of software infrastructure needed to achieve the desired results with consistent quality and robust efficiency. A modern DBMS is a thing of wonder and embodies in its internal construction and in its wide usability many advances in algorithms and data structures, programming language theory, conceptual modelling, concurrency theory, and distributed computing. This makes the study of databases a data-centric traversal of many of the most exciting topics in modern computing.

Aims

The aim of this course unit is to introduce the students to the fundamental concepts and techniques that underlie modern database management systems (DBMSs).

The course unit studies the motivation for managing data as an asset and introduces the basic architectural principles underlying modern DBMSs. Different architectures are considered and the application environments they give rise to.

The course unit then devotes time to describing and motivating the relational model of data, the relational database languages, and SQL, including views, triggers, embedded SQL and procedural approaches (e.g., PL/SQL).

The students learn how to derive a conceptual data model (using the Extended Entity Relationship paradigm), how to map such a model to target implementation model (for which the relational model is used), how to assess the quality of the latter using normalisation, and how to write SQL queries against the improved implementation model to validate the resulting design against the data requirements originally posed. For practical work, the Oracle DBMS is used.

The course unit also introduces the fundamentals of transaction management including concurrency (e.g., locking, 2-phase locking, serialisability) and recovery (rollback and commit, 2-phase commit) and of file organisation (e.g., clustering) and the use of indexes for performance.

Finally, the course unit addresses the topic of database security by a study of threats and countermeasures available.

In the case of the former, these include potential theft and fraud as well as loss of confidentiality, privacy, integrity and availability. In the case of the latter these primarily include mechanisms for authorization and access control, including the use of views for that purpose. The course unit also addresses the topic of legal frameworks that give rise to obligations on the part of database professionals by introducing exemplars such as the 1995 EU Directive on Data Protection and the 1998 UK Data Protection Act.

Additional reading

Contents

1 An introduction to Database Management Systems

DataBase Management Systems (DBMS's) are a type of middleware that provide a layer of abstraction for dealing with databases. It is nearly always unnecessary to write software from scratch that interfaces with a database, since a lot of database operations will share a significant amount of logic.

Henceforth, a lot of the functionality required of applications that make use of a database is placed into a DBMS, which application developers can make use and save time. The DBMS acts as a service, that is well implemented and is able to enforce good practices and advanced techniques such as concurrency, sharding, recovery management and transactions.

Some advantages of using a DBMS include:

- It decouples data inside a database from the application using it. Either can be re-written at any time so long as they still provide/use the same interface.
- Since the data is decoupled from the application, using a DBMS (in theory) lowers the development cost of the application.
- Most DBMS are scalable, concurrent, fault tolerant, authorisation control (often role based for organisations).

Even though the DBMS aims to provide a layer of abstraction for a user application, there are several layers of abstraction within the DBMS itself. These are:

Physical	Deals with the file(s) that is written to the storage medium that will hold the database. Needs to know about file formats, indexing, compression, etc.
Logical	Mainly concerned with mapping the raw data into database 'concepts' such as tables, views etc. It is here that the formal specification of the database is defined, commonly used models include *relational, XML based and document based*
View	Ensures that only authorised people can view the data.

If the database is using a relational format, then it will be defined by a schema. A schema dictates how the database is formatted; what tables there are, and what datatypes their columns take. An instance of a database is the content (data) inside of the database at a particular point in time. There is a certain isomorphism between relational databases and imperative programming languages; a schema would be akin to the declaration of variables (i.e. their names and types), while the instance would be their values at a particular point in the program's execution.

Irrespective of what logical model a database uses, most DBMS use between one and three languages to interface with a user/application. These are:

- Data Definition Language - used for specifying schema.
- Data Manipulation Language - used for mutating the data in the database.
- Data Query Language - used to access data in the

database.

Often DBMS languages will be both a DML and a DQL, and sometimes a DDL too! One such example is SQL, does all of the above!