Lecture Slides available: PDF PowerPoint

Introduction

Contents

Relational database systems have become increasingly popular since the late 1970's. They offer a powerful method for storing data in an application-independent manner. This means that for many enterprises the database is at the core of the I.T. strategy. Developments can progress around a relatively stable database structure which is secure, reliable, efficient, and transparent.

In early systems, each suite of application programs had its own independent master file. The duplication of data over master files could lead to inconsistent data.

Efforts to use a common master file for a number of application programs resulted in problems of integrity and security. The production of new application programs could require amendments to existing application programs, resulting in `unproductive maintenance'.

Data structuring techniques, developed to exploit random access storage devices, increased the complexity of the insert, delete and update operations on data. As a first step towards a DBMS, packages of subroutines were introduced to reduce programmer effort in maintaining these data structures. However, the use of these packages still requires knowledge of the physical organization of the data.

The Database Approach

A database system is a computer-based system to record and maintain information. The information concerned can be anything of significance to the organisation for whose use it is intended.

The contents of a database can hold a variety of different things. To make database design more straight-forward, databases contents are divided up into two concepts:

- Schema
- Data

The Schema is the structure of data, whereas the Data are the "facts". Schema can be complex to understand to begin with, but really indicates the rules which the Data must obey.

Imagine a case where we want to store facts about employees in a company. Such facts could include their name, address, date of birth, and salary. In a database all the

information on all employees would be held in a single storage "container", called a *table*. This table is a tabular object like a spreadsheet page, with different employees as the rows, and the facts (e.g. their names) as columns... Let's call this table EMP, and it could look something like:

Name	Address	Date of Birth	Salary
Jim Smith	1 Apple Lane	1/3/1991	11000
Jon Greg	5 Pear St	7/9/1992	13000
Bob Roberts	2 Plumb Road	3/2/1990	12000

From this information the *schema* would define that EMP has four components, "NAME","ADDRESS","DOB","SALARY". As designers we can call the columns what we like, but making them meaningful helps. In addition to the name, we want to try and make sure that people dont accidentally store a name in the DOB column, or some other silly error. Protecting the database against rubbish data is one of the most important database design steps, and is what much of this course is about. From what we know about the facts, we can say things like:

- NAME is a string, and needs to hold at least 12 characters.
- ADDRESS is a string, and needs to hold at least 12 characters.
- DOB is a date... The company forbids people over 100 years old or younger than 18 years old working for them.
- SALARY is a number. It must be greater than zero.

Such rules can be enforced by a database. During the design phase of a database schema these and more complex rules are identified and where possible implemented. The more rules the harder it is to enter poor quality data.

User Types

When considering users of a Database system, there are three broad classes to consider:

1. the application programmer, responsible for writing programs in some high-level language such as COBOL, C++, etc.
2. the end-user, who accesses the database via a query language
3. the database administrator (DBA), who controls all operations on the database

Database Architecture

DBMSs do not all conform to the same architecture.

- The three-level architecture forms the basis of modern database architectures.
- this is in agreement with the ANSI/SPARC study group on Database Management Systems.
- ANSI/SPARC is the American National Standards Institute/Standard Planning and Requirement Committee).
- The architecture for DBMSs is divided into three general levels:
- external
- conceptual
- internal

Three level database architecture

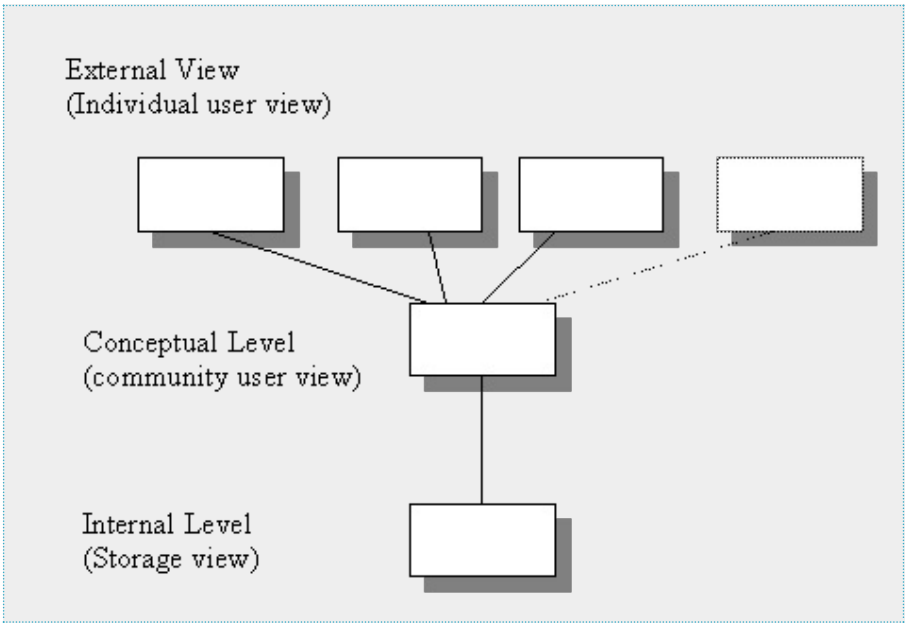

Figure 1: Three level architecture

1. the external level : concerned with the way individual users see the data
2. the conceptual level : can be regarded as a community user view a formal description of data of interest to the organisation, independent of any storage considerations.
3. the internal level : concerned with the way in which the data is actually stored

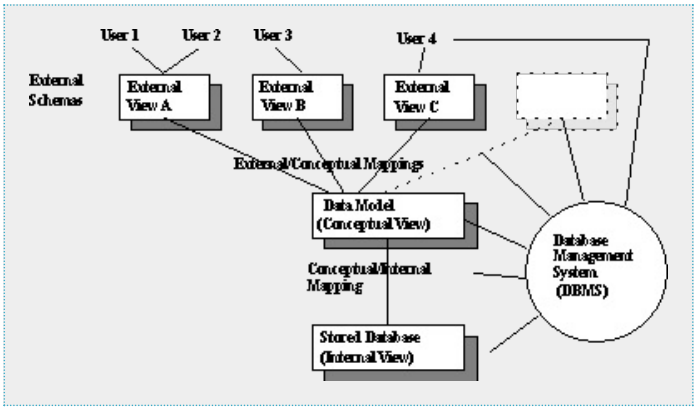

Figure : How the three level architecture works

External View

A user is anyone who needs to access some portion of the data. They may range from application programmers to casual users with adhoc queries. Each user has a language at his/her disposal.

The application programmer may use a high level language (e.g. COBOL) while the casual user will probably use a query language.

Regardless of the language used, it will include a data sublanguage DSL which is that subset of the language which is concerned with storage and retrieval of information in the database and may or may not be apparent to the user.

A DSL is a combination of two languages:

- a data definition language (DDL) - provides for the definition or description of database objects
- a data manipulation language (DML) - supports the manipulation or processing of database objects.

Each user sees the data in terms of an external view: Defined by an external schema, consisting basically of descriptions of each of the various types of external record in that external view, and also a definition of the mapping between the external schema and the underlying conceptual schema.

Conceptual View

- An abstract representation of the entire information content of the database.

- It is in general a view of the data as it actually is, that is, it is a `model' of the `realworld'.
- It consists of multiple occurrences of multiple types of conceptual record, defined in the conceptual schema.
- To achieve data independence, the definitions of conceptual records must involve information content only.
- storage structure is ignored
- access strategy is ignored
- In addition to definitions, the conceptual schema contains authorisation and validation procedures.

Internal View

The internal view is a low-level representation of the entire database consisting of multiple occurrences of multiple types of internal (stored) records.

It is however at one remove from the physical level since it does not deal in terms of physical records or blocks nor with any device specific constraints such as cylinder or track sizes. Details of mapping to physical storage is highly implementation specific and are not expressed in the three-level architecture.

The internal view described by the internal schema:

- defines the various types of stored record
- what indices exist
- how stored fields are represented
- what physical sequence the stored records are in

In effect the internal schema is the storage structure definition.

Mappings

- The conceptual/internal mapping:
  - defines conceptual and internal view correspondence
  - specifies mapping from conceptual records to their stored counterparts
- An external/conceptual mapping:
  - defines a particular external and conceptual view correspondence
- A change to the storage structure definition means that the conceptual/internal mapping must be changed accordingly, so that the conceptual schema may remain invariant, achieving physical data independence.
- A change to the conceptual definition means that the conceptual/external mapping must be changed accordingly, so that the external schema may remain invariant, achieving logical data independence.

DBMS

The database management system (DBMS) is the software that:

- handles all access to the database
- is responsible for applying the authorisation checks and validation procedures

Conceptually what happens is:

1. A user issues an access request, using some particular DML.
2. The DBMS intercepts the request and interprets it.
3. The DBMS inspects in turn the external schema, the external/conceptual mapping, the conceptual schema, the conceptual internal mapping, and the storage structure definition.
4. The DBMS performs the necessary operations on the stored database.

Database Administrator

The database administrator (DBA) is the person (or group of people) responsible for overall control of the database system. The DBA's responsibilities include the following:

- deciding the information content of the database, i.e. identifying the entities of interest to the enterprise and the information to be recorded about those entities. This is defined by writing the conceptual schema using the DDL
- deciding the storage structure and access strategy, i.e. how the data is to be represented by writing the storage structure definition. The associated

internal/conceptual schema must also be specified using the DDL
- liaising with users, i.e. to ensure that the data they require is available and to write the necessary external schemas and conceptual/external mapping (again using DDL)
- defining authorisation checks and validation procedures. Authorisation checks and validation procedures are extensions to the conceptual schema and can be specified using the DDL
- defining a strategy for backup and recovery. For example periodic dumping of the database to a backup tape and procedures for reloading the database for backup. Use of a log file where each log record contains the values for database items before and after a change and can be used for recovery purposes
- monitoring performance and responding to changes in requirements, i.e. changing details of storage and access thereby organising the system so as to get the performance that is `best for the enterprise'

Facilities and Limitations

The facilities offered by DBMS vary a great deal, depending on their level of sophistication. In general, however, a good DBMS should provide the following advantages over a conventional system:

- Independence of data and program - This is a prime advantage of a database. Both the database and the user program can be altered independently of each other thus saving time and money which would be required to retain consistency.
- Data shareability and nonredundance of data - The ideal situation is to enable applications to share an integrated database containing all the data needed by the applications and thus eliminate as much as possible the need to store data redundantly.
- Integrity - With many different users sharing various portions of the database, it is impossible for each user to be responsible for the consistency of the values in the database and for maintaining the relationships of the user data items to all other data item, some of which may be unknown or even prohibited for the user to access.
- Centralised control - With central control of the database, the DBA can ensure that standards are followed in the representation of data.
- Security - Having control over the database the DBA can ensure that access to the database is through proper channels and can define the access rights of any user to any data items or defined subset of the database. The security system must prevent corruption of the existing data either accidently or maliciously.
- Performance and Efficiency - In view of the size of databases and of demanding database accessing requirements, good performance and efficiency are major requirements. Knowing the overall requirements of the organisation, as opposed to the requirements of any individual user, the DBA can structure the database system to provide an overall service that is `best for the enterprise'.

Data Independence

- This is a prime advantage of a database. Both the database and the user program can be altered independently of each other.
- In a conventional system applications are datadependent. This means that the way in which the data is organised in secondary storage and the way in which it is accessed are both dictated by the requirements of the application, and, moreover, that knowledge of the data organisation and access technique is built into the application logic.
- For example, if a file is stored in indexed sequential form then an application must know
  - that the index exists
  - the file sequence (as defined by the index)

The internal structure of the application will be built around this knowledge. If, for example, the file was to be replaced by a hash-addressed file, major modifications would have to be made to the application.

Such an application is data-dependent - it is impossible to change the storage structure (how the data is physically recorded) or the access strategy (how it is accessed) without affecting the application, probably drastically. The portions of the application requiring alteration are those that communicate with the file handling software - the difficulties involved are quite irrelevant to the problem the application was written to solve.

- it is undesirable to allow applications to be data-dependent - different

applications will need different views of the same data.
- the DBA must have the freedom to change storage structure or access strategy in response to changing requirements without having to modify existing applications.
- Data independence can be defines as
  `The immunity of applications to change in storage structure and access strategy'.

Data Redundancy

In non-database systems each application has its own private files. This can often lead to redundancy in stored data, with resultant waste in storage space. In a database the data is integrated.

The database may be thought of as a unification of several otherwise distinct data files, with any redundancy among those files partially or wholly eliminated.

Data integration is generally regarded as an important characteristic of a database. The avoidance of redundancy should be an aim, however, the vigour with which this aim should be pursued is open to question.

Redundancy is

- direct if a value is a copy of another
- indirect if the value can be derived from other values:
  - simplifies retrieval but complicates update
  - conversely integration makes retrieval slow and updates easier
- Data redundancy can lead to inconsistency in the database unless controlled.
- the system should be aware of any data duplication - the system is responsible for ensuring updates are carried out correctly.
- a DB with uncontrolled redundancy can be in an inconsistent state - it can supply incorrect or conflicting information
- a given fact represented by a single entry cannot result in inconsistency - few systems are capable of propagating updates i.e. most systems do not support controlled redundancy.

Data Integrity

This describes the problem of ensuring that the data in the database is accurate...

- inconsistencies between two entries representing the same `fact' give an example of lack of integrity (caused by redundancy in the database).
- integrity constraints can be viewed as a set of assertions to be obeyed when updating a DB to preserve an error-free state.
- even if redundancy is eliminated, the DB may still contain incorrect data.
- integrity checks which are important are checks on data items and record types.

Integrity checks on data items can be divided into 4 groups:

1. type checks
   - e.g. ensuring a numeric field is numeric and not a character - this check should be performed automatically by the DBMS.
2. redundancy checks
   - direct or indirect (see data redundancy) - this check is not automatic in most cases.
3. range checks
   - e.g. to ensure a data item value falls within a specified range of values, such as checking dates so that say (age > 0 AND age < 110).
4. comparison checks
   - in this check a function of a set of data item values is compared against a function of another set of data item values. For example, the max salary for a given set of employees must be less than the min salary for the set of employees on a higher salary scale.

A record type may have constraints on the total number of occurrences, or on the insertions and deletions of records. For example in a patient database there may be a limit on the number of xray results for each patient or the details of a patients visit to hospital must be kept for a minimum of 5 years before it can be deleted

- Centralized control of the database helps maintain integrity, and permits the DBA to define validation procedures to be carried out whenever any update operation is attempted (update covers modification, creation and deletion).
- Integrity is important in a database system - an application run without

validation procedures can produce erroneous data which can then affect other applications using that data.

Previous Page Up One Level