

Review topics

1. Overview of Database Management Systems

1.1 What is a database management system?

1.2 Advantages of DBMS:

Easier definition of applications

Integration of data from different applications

Different users need to "see" different aspects of a record in a format that they need.

Important to an organization

Avoid redundancy

Data independence

Conceptual level vs. internal level

Easy to adapt to changing environment

Performance

Response time

Flexibility of searching

1.3 Data Models

Hierarchical

Network

Relational

Object-oriented (picks some features of the hierarchical data model)

1.4 What is a relational DBMS?

2. System Design Process

2.1 Determine user requirements

A list of users --- Who needs information? Who searches? In what level is their knowledge about the computer, DBMS, and query languages?

A list of expected queries (including importance, frequency and required answer speed, etc.)

How important the data are to the user

How important the validity or integrity of the data are to the user

Who is authorized to access/receive what information

Methods for determining user requirement:

Problem analysis (check job description, etc.)

Talk to the user

(See Soergel, Organizing Information, Chapter 7)

2.2 Develop user-system interface specifications

Query input (screen formats for requesting data about the query, menus where appropriate)

Output screens and report formats

2.3 Construct a conceptual schema

Determine the entity types needed

Consider inherent attributes in defining entity types; do not establish several entity types on the basis of different roles. For example, person is a proper entity type, physician and patient (persons in different roles) are not. The same real-world object should not belong to two different entity types or change its entity type over time.

Consolidate entity types where appropriate

Result: the number of entity types is kept as small as possible while still expressing all the information required

Consider hierarchical relationships among entity types

Determine the relationship types needed (the lists of entity types and of relationship types are usually developed in tandem)

A relationship type must express a direct relationship between its left argument (usually one entity) and its right side (one entity or a combination of several entities); a relationship is direct when it is not possible to introduce two separate relationships from which the original relationship can be deduced. For example, the relationship between a book and its place of publication is not direct because it can be deduced from the relationship between book and publisher and the relationship between publisher and place. This rule guarantees a redundancy-free (normalized) database. (In the definition of tables one may want to introduce redundancy in storage in as controlled fashion for the sake of speed.)

Use relationships that have more than two arguments when needed, but only then

Consider hierarchical relationships among relationship types

Express the conceptual schema in form of table definitions for a relational database

As a general rule, the number of tables should be kept as small as possible while being consistent with the relational model and with internal requirements for storage and processing speed

Arrange the relationship types into groups such that the relationship types in a group all have the same first argument (change the order of the arguments where appropriate); this argument is called the head entity type of the group

Within a group, relationships for which the database can contain at the most one statement for each head entity value can be combined in one multi-relationship single-record table, i.e. a table with one record (row) for each head entity value; relationship types for which there can be multiple statements for each head entity value must be expressed by their own multiple-record table, i.e. a table with 0, 1, or more rows for each head entity value (unless the DBMS allows repeating fields and thus deviates from the relational model)

If the DBMS uses up space in a table even if a field contains a null value, then a relationship type for which only a fraction of the head entity values have statements in the database should not be included in a multi-relationship table in order to conserve space.

If data captured by a given relationship type are needed very frequently, consider a separate table to minimize disk input/output.

With straightforward translation from the entity-relationship schema, table name and column together define the meaning of a piece of data. However, it is often useful to combine several multiple-record table for the same head entity type into one table and add a data-defining column; the value in that column indicates the relationship type to which the record belongs. This method can also be used to unburden a multiple-relationship, multi-record table from an excessive number of columns without increasing the overall number of tables too much. This method has the further advantage that some new relationship types can be accommodated without adding table definitions.

Consider the amount of data and storage requirements under the schema developed

2.4 **Develop input specifications**

Sources of data and the cost of data collection

data input path: through what steps are the data transferred from the source into the system; this may involve intermediate storage on paper or in another computer.

Recency (how quickly and how often to collect the data)

Sources of error and how to avoid, catch, or correct errors

Authority check for input

2.5 **Implementation**

2.6 **Cost-benefit considerations**

During the entire process

3. **DBMS Features** (See handout)

4. Data warehousing and data mining

4.1 Data warehousing

Combining data from many different operation support systems

Cleaning data, making them compatible

Aggregating data, store aggregated data

Data and storage models: Star schema, data cube (multidimensional data model)

Data marts

OLAP (OnLine Analytical Processing)

4.1 Data mining

Discovering patterns in data

Relies on clean and compatible data (much easier if there is a data warehouse)

5. The Database Environment

5.1 Data Dictionary

5.2 Integrity constraints

Constraints that enforce integrity rules for the database - error control

Can specify in index that data values entered must be unique

Can also specify a sort of authority control that supported by some DBMSs.

5.3 Query optimization

Use knowledge of internal data structure to adjust query for fastest processing

Use of indexes

An index might not always be faster than sequential access (index is better if only a few records searched, sequential search is better if many records need to be accessed)

5.4 Transactions and recovery

COMMIT and ROLLBACK operations

5.5 **Concurrency and locking**

Multi-user systems that need to allow simultaneous access to a system, or a table

5.6 **Security**

Security specification could go to any levels

The user will only see those parts that he has security for

5.7 **Distributed systems**

Becoming more important

Parts of the data reside in different location. No one has a complete data file. Each computer has only the data that its local users need most frequently

Makes internal structure & query optimization more complex

Ideal distributed systems is one in which the user does not know it is distributed (the distribution of the data is transparent to the user)

5.8 **Web-accessible databases**

Native to the DBMS (e.g., Access, Oracle)

Middleware (e.g., ColdFusion)

6. **System Structure**

6.1 **Integrated systems**

Integration with word processing; for example, issue a database command which would introduce data immediately put into text, or, embed database commands in a text file which are executed at time the report is printed out or put on the screen.

Integration with spreadsheets and other data analysis programs, such as project planning or computer-assisted design

Integration with graphics

Integration with telecommunication