AIT 524

This is the Screenshot of the Slides used in the following Video url: https://www.youtube.com/watch?v=6u2zsJOJ_GE

Sunday, October 8, 2017 12:53 PM

Overview

- Applications of database technology
- Definitions
- File based versus database based approach to data management
- Elements of a database system
  - Data model
  - Schemas and instances
  - The three-schema architecture
  - Data dictionary (catalog)
  - DBMS languages
- Advantages of database design

Definitions

- ***Database***: collection of interrelated data, which
  - represents some aspects of (a subset of) the real world: the Universe of Discourse (UoD)
  - is logically coherent with some inherent meaning
  - has an intended group of users and applications
- ***Database Management System (DBMS):*** a collection of programs that enables us to create and maintain a database. It is a general-purpose software system that facilitates the process of *defining*, *constructing* and *manipulating* databases for various applications.
- A ***Database system*** consists of the combination of a DBMS and a database.

File based approach to data management

- Waste of storage space because of duplicates (= redundant data)
  - Inconsistent data
    e.g. Customer data changed in only one file
- Strong dependency between applications and data
  - Change in data definition necessitates changes in all applications that use the data
- Difficult to integrate various applications
  - High difficulty → high cost

Applications of database technology (1)

- Storage and retrieval of "traditional" numeric and alphanumeric data
  - stock administration: keeping track of the number of products in stock
- Storage and retrieval of multimedia data (pictures, video, sound, ...)
  - YouTube : Popular video database for audiovisual data
- Monitor data, autonomously take action when required
  - High-frequency trading: constantly maintain a database of stock prices, process updates in less than 1 ms, including buying/selling stocks

Applications of database technology (2)

- Storage and retrieval of Web content (HTML, PDF, images, XML, ...)
  - Google : Google caching to retrieve websites which are currently offline
- Storage of large datasets for analysis (data warehouses)
  - WAL•MART : Collection and analysis of customer purchasing behavior for shop floor optimization

File based approach: example

File-based (pseudo-code)

```
find person (name:input, record:output)
begin
open people
repeat while people has next
  people → go to next record
  record := people → current record
  if record → person is name
        done
  else
        continue
  end
end
record := invalid
end
```

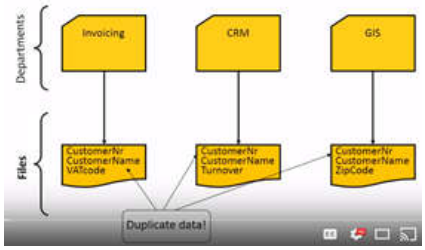
Database (SQL)

```
SELECT *
FROM people
WHERE
name = $NAME
```

A database-oriented approach to data management: advantages

Database	File-based
The DBMS contains both the data and a description of the database structure and constraints. These metadata are stored in the *catalog*.	Only the data is stored in a file. The structure of this data is stored in the applications that will access the file.
Changing the structure of the DBMS would require updating the catalog, but not the applications. "program-data independence"	Changing the structure of a file would require changing each application that accesses it.
Different views over (a subset of) the data can be defined. This allows different users with different needs to use the same data.	Different views would require copying (parts of) some files. Any changes made to those duplicates should be carefully merged in all other duplicates.
Multiple users can access the same data at the same time. For this, the DBMS includes *concurrency control* to ensure no conflicting operations are executed.	It is impossible to have multiple people or applications working on the same file.

A step back in time: File based approach to data management



Data model

- Definition: collection of concepts that can be used to describe the structure of a database, i.e. the data types, relationships, constraints, …
- Types of data models:
  - *Conceptual data model:* high-level concepts, a representation of (part of) reality, close to how the user perceives the data
    → e.g.: (E)ER, object-oriented models (Unified Modeling Language)
  - *Implementation data model:* concepts that may be understood by end users but are not too far removed from physical data organization They hide some details of data storage but can be implemented on a computer system in a direct way.
    → e.g. the hierarchical model, the network model, the relational model, the ODMG model
  - *Physical data model:* low-level concepts that describe the data's physical storage details

Schemas, instances and database state

In any data model it is important to distinguish between the *description* of the data and the *data itself*:

- ***Database schema***: description of a database, which is specified during database design and is not expected to change too frequently.
  = the information stored in the catalog
- ***Database state***: the data in the database at a particular moment, also called the current set of *instances*.
  = the data currently in the database

Sample extract of a university database (1)

Database schema

STUDENT (number, name, address, email)

COURSE (number, name)

BUILDING (number, address)

…

Database state

STUDENT

Number	Name	Address	Email
0165854	John Doe	154 West Str.	John.doe@student.kuleuven.be
0168975	Fred Astere	7 Scene Field Str.	Fred.astere@student.kuleuven.be
0157895	Kyara Welton	89 Fifth Av.	Kyara.welton@student.kuleuven.be

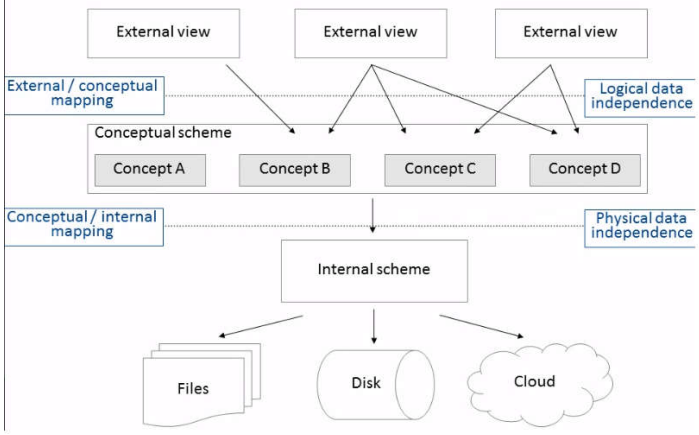
COURSE

Number	Name
D0I69A	ICT Service Management
D0R04A	Strategic management
D0T21A	Macro economics

BUILDING

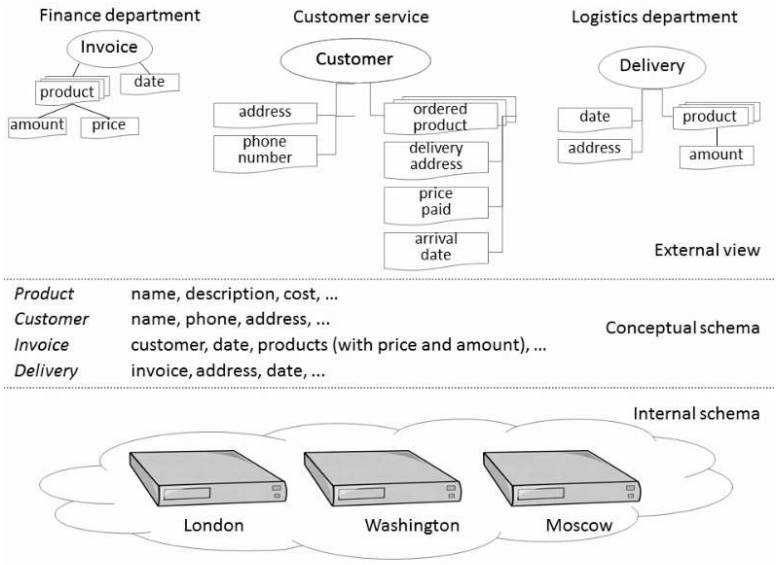
Number	Address
0589	Naamsestraat 69
0365	Naamsestraat 78
0589	Tiensestraat 115

The three-schema architecture



The three-schema architecture

- External view/user views
  - each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. A high-level data model or an implementation data model can be used at this level.
- Conceptual schema
  - Specifies objects, characteristics of objects, relationships between objects, integrity rules, and object behavior
  - It hides the details of physical storage structures and concentrates on describing entities, data types, relationships and constraints. A high-level data model or an implementation data model can be used at this level.
- Internal schema
  - Specifies how the data are stored/organised physically (e.g. indexes, access paths, ...)
- Changes in one layer should have minimal impact on the others



Data dictionary (catalog)

- Heart of the database system
- System database with metadata
- Contains the definitions of:
  - Conceptual schema
  - External view/user views
  - Physical schema

DBMS languages

- *DBMS Languages*
  - ***Data Definition Language (DDL):*** language used by the database administrator (DBA) to define the database's conceptual, internal and external schemas
  - ***Data Manipulation Language (DML):*** language used to retrieve, insert, delete and modify data. DML statements can be entered interactively from a terminal or embedded in a general-purpose programming language.

→ For relational database systems, *SQL (Structured Query Language)* is both the DDL and DML. It can be used interactively (*interactive SQL*) or embedded in a programming language (*embedded SQL*).

- Database designers / database administrators /end users

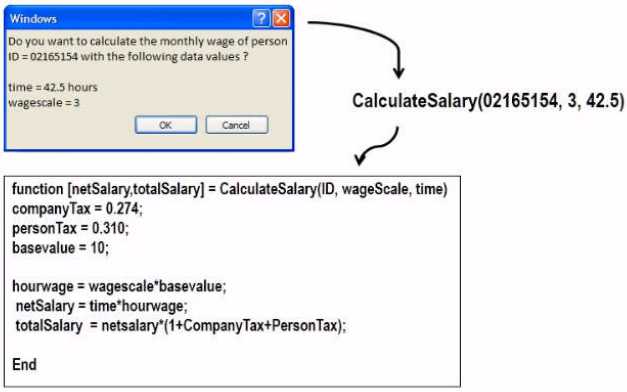
Advantages of using database design

- Data and functional independence
- Database modeling
- Managing data redundancy
- Specifying integrity rules
- Data security
- Backup and recovery facilities
- Performance utilities

Data independence

- Software applications must not be changed when changes occur to data definitions
- Physical data independence
  - Neither applications nor conceptual schema must be changed when changes are made to data storage specifications
  - e.g., new access paths, indices, different storage media, …
  - DBMS provides interfaces between the conceptual and physical data models
- Logical data independence
  - Software applications must not be changed when changes occur in the conceptual schema
  - e.g., adding new objects or new characteristics of objects to the conceptual schema has no impact on software applications because of the external views
  - DBMS provides interfaces between the conceptual schema and external views

Functional Independence: example 1



Functional independence

- Function
  - Interface (signature): name of the function and its arguments
  - Implementation (method): specifies how the function should be executed
- Implementation (method) can change without impact on software applications
- Information hiding
- e.g. sorting

Functional Independence: example 2

Bubble sort

```
procedure Sort(var A: TArray);
var
  i, j: Integer;
  Help: Byte;
begin
  for i := 1 to Size do
    for j := 1 to Size - i do
      if A[j] > A[j + 1] then begin
        Help := A[j]; A[j] := A[j + 1]; A[j + 1] := Help;
      end;
end;
```

Insertion sort

```
procedure Sort(var A: TArray);
var
  i, j: Integer;
  v: Byte;
begin
  A[0] := 0;
  for i := 2 to Size do begin
    v := A[i] ; j := i;
    while a[Pred(j)] > v do begin
      A[j] := A[Pred(j)];
      dec(j);
    end;
    A[j] := v;
  end;
end;
```

"I don't care **how** it is done, just get it done."

Database Modeling

- Data model specifies objects, characteristics of objects, relationships between objects, integrity rules, and functions
- Should provide a formal and perfect mapping of the real world
  - Best-case scenario, mappings are rarely perfect
- Example data models: hierarchical model, CODASYL model, (E)ER model, relational model, object oriented model, …

Managing data redundancy

- Redundant data may be desired
  - Increasing performance
  - Using distributed environments
- DBMS manages redundancy
  - Synchronization
  - Consistency
- Compared to file based redundancy
  - DBMS guarantees correctness
  - No user intervention required
  - Much more efficient
  - Errors are extremely unlikely

Specifying integrity rules (1)

- Integrity rules determine correctness of data
  - Syntactical errors, e.g. customerID not numeric
  - Semantical errors, e.g. customerID not unique; trying to delete a customer who has pending invoices
- Integrity rules are stored
  - Embedded in applications in the file based approach to data management
  - Specified as part of the conceptual schema and stored in data dictionary (catalog) in database approach

Specifying integrity rules (2)

management

- Specified as part of the conceptual schema and stored in data dictionary (catalog) in database approach

Data security issues

- Some users have read access, others write access, some may access the whole database, others only certain parts
- Trends such as e-business (B2B, B2C), CRM, … stress the importance of data security
- Data access can be managed via user accounts and passwords for users or user groups
- Each account has its own authorization rules, which are stored in the catalog

Specifying integrity rules (2)

- Integrity rules are enforced
  - By the applications accessing the files
  - By the DBMS whenever anything is updated (data loading, data manipulation, adding new integrity rules, ...)
- Note: integrity rules can be sensitive to simultaneous usage of data in a distributed environment

  e.g. same product being sold to two different customers

→ Concurrency control!

Performance utilities

- Examples
  - Distributing data storage
  - Tuning indices to allow faster queries
  - Tuning queries to improve application performance
  - Optimizing buffer management
- Parf of the job of the DBA (database administrator)

Backup and recovery facilities

- In case of loss of data due to e.g.
  - Hardware or network errors
  - Bugs in system software or application programs
- Backup facilities perform full or incremental backups
  - This allows you to restore the system to a previous state
- Recovery facilities allow to restore data after loss or damage
  - This allows you to recreate the state of the system prior to the failure