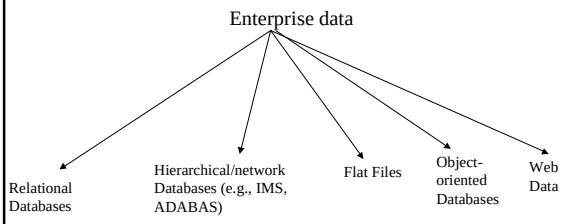
Database Technologies and SQL

- Database technologies
- Relational databases
- SQL
- Object Oriented Databases and object-orientation

Amjad Umar

Enterprise Data



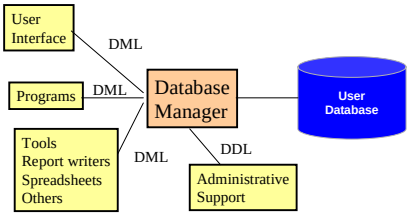
Enterprise Data: Enterprise Data (also known as Corporate Data) is the information that is used or created by a corporation in conducting business and is shared across the business processes of the corporation.

Database Technologies

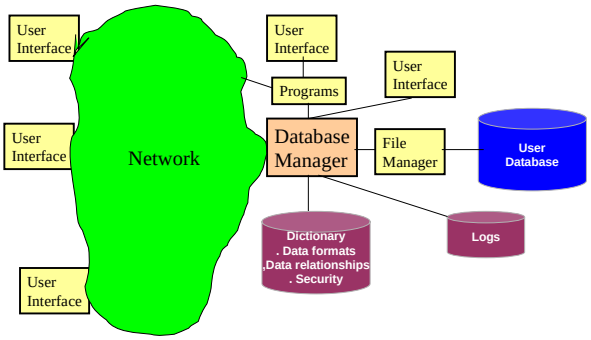
- Definitions:
  - Data item (e.g., name, address, age)
  - Data record: collection of data items (e.g., student record)
  - File: collection of similar records (e.g., student file)
  - Database: collection of files (e.g., university database)
- Informally: a database is a pool of data
- Databases are managed by a database management system (DBMS)

DML: Data Manipulation Language

DDL: Data Definition Language

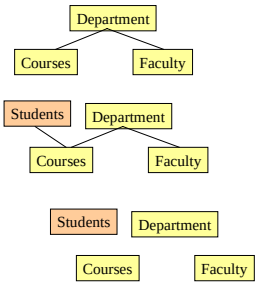


Database Management Systems

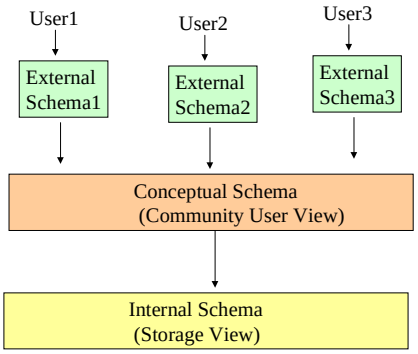


Data Models Supported by DBMS

- Hierarchical Data Model
  - One parent per record
  - Example is IBM's IMS
- Network Data Model
  - More than one parent per record
  - Example: IDMS
  - IMS support through "logical relationships"
- Relational Data Model
  - All data is viewed as tables ("relations")
  - Input is tables
  - Output (result of processing) is also tables
  - Three basic operations: selection, projection, join
  - SQL is the standard language
  - Relationships are identified through keys
- Object-oriented databases
  - View data as objects, not as tables
  - Use complex relationships
  - Employ AI concepts



Database Schema



Relational Database Technologies

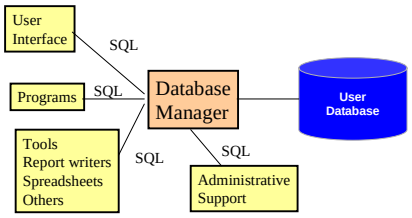
- Idea introduced by E.F. Cod (1971), products introduced around 1982
- Data viewed as tables
- Only three basic operations for data retrieval
  - selection; choose roows
  - Projection: choose columns
  - Join: Combine two or more tables on a column

Name	Age	Salary (K)	Location
Joe	35	42	NY
Pat	29	60	LA
Bruce	25	42	Chicago
Sam	40	75	NY

Manager	Phone	Location
Donna	555-1000	NY
Roger	555-1111	LA
Dave	555-2222	Chicago

- Can reach data through different tables (joins need only one common field)
- Theoretical terminology:
  - Relation = table or file (fixed fields in each record, each row is unique)
  - Tuple= row; Attribute= field
  - Degree = number of attributes; Domain = range of values

SQL - The Core of Relational Databases



SQL - A Quick Overview

- Structured Query Language (SQL) is the standard query language for relational databases.
- SQL provides interactive ad hoc queries as well as program interfaces in C, Cobol, Fortran, ADA, PL1 and many other programming languages.
- SQL language consists of a set of facilities for defining, manipulating, and controlling data in a relational database.
- SQL provides DDL and DML facilities
- DDL: Allows you to define tables
  - Example: create a parts and a customers table:

```
CREATE TABLE parts (part_no char(4), part_name char(5)), price numeric(5)) ;
CREATE TABLE customers (cust_name  char(30) , address  char(30) ,  cust_id
 char(12), part_no  char(4)).;
```

- SQL has powerful data manipulation facilities.
- There are four basic SQL operations:
  - SELECT
  - UPDATE
  - INSERT
  - DELETE

- All data retrievals are invoked by a SELECT command:

  SELECT <a1,a2,a3,...,an> FROM <t1,t2,...,tm> WHERE <conditions>;

  where a1, a2,,, an are the attributes; t1, t2,, tm, are the tables; and the conditions, if specified, indicate the retrieval criteria.
- Examples:

  " SELECT part_no, part_name FROM parts WHERE price > 100 "

  " SELECT part_no, part_name FROM parts"

  " SELECT * FROM parts WHERE price > 100 "
- Joins: use multiple tables and equate joining field:

  " SELECT part_no, part-price, cust_name FROM parts, customers WHERE customer.part_no = parts.part_no ".
- Complex queries: imbed queries within queries (complex queries)

  SELECT part-no, part-name FROM parts WHERE part-no in

  (SELECT * FROM customers WHERE city = "Detroit");
- SQL display facilities: order, avg, sum, count, group by, etc.

  EXAMPLE: SELECT part_no, part_name FROM parts ORDER BY part-no;

  EXAMPLE: SELECT AVG(price), MIN(price), MAX(price), SUM(price), COUNT(*) FROM parts;

  EXAMPLE: SELECT AVG(salary) FROM employees group by title;

- Predicates (used in where)
  - Comparison; =, <>, <, >, <=, >=
  - Between/not between: where price between 5 and 10;
  - in/not in; where price not in (5, 7, 10);
  - Like/not like: pattern recognition

  For single character, % for 0 to n characters

Example; where cust-name like "A%".

- SQL data modification: allow insertion, deletion and update of data in tables

  INSERT INTO parts(part_no, part_name, price) (xy22, rods, 100);

  INSERT INTO parts(part_no, part_name, price)

  (select * FROM parts WHERE price >100);

  DELETE FROM parts where part_no = xy20;

  UPDATE PARTS set price=120 where part_no=xy22;
- Views can be created to operate on portions of table

  "CREATE VIEW salesman AS SELECT name, id FROM employee WHERE job="sales";

  Views are treated as tables, operations are performed on actual tables
- Administrative facilities

  GRANT access-type ON tablename TO id;

  where access-type: all priviliges, update, select, insert

  REVOKE access-type from id;

- Embeddded SQL:
  - SQL statementes can be embedded in programs

  EXEC SQL

  SQL statements

  END SQL
  - CURSOR: To handle multiple rows returned
  - FETCH to retrieve each row
  - Many programming details
- Performance
  - Query optimization is responsibility of DBMS
  - Clever, implementation dependent techniques are not recommended
  - Indexes can be created for fast access

  CREATE INDEX indexname ON tablename;
  - Many indexes can be created on a table
  - SQL determines when to use index, what index to use, etc.
  - Index not used if "where' clause not specified
- Many SQL products commercially available

Table definition:

```
CREATE TABLE parts (part_no numeric (4), part_name char(5)), part-price numeric(5)) ;
```

Sample code to Extract and display information from parts table;

```
Include SQLCA
Int p-number, p-price;
...
Exec SQL select part-no, part-price into :p-number, p-price from parts where part-no =75567;
Print (p-number, p-price);
.....
Exec SQL cursor CS
        select part-no, part-price into   where part-price >200;
Exec SQL select part-no, part-price into :p-number, p-price from parts where part-price =200;
Exec SQL open CS;
Do {Exec SQL Fetch CS into :p-number, p-price; } while sqlcode =0;
        Exec SQL close CS;
```


Typical DBMSs

Mainframe DBMSs

- Information Management System (IMS)
  - Introduced in 1969 for IBM mainframes
  - Uses a hierarchical data model, can represent network data model ("logical" relationships)
  - Can be accessed through IMS TM and CICS
  - Database language is DL1,
  - Extremely well suited for large scale OLTP
  - Not suited for ad hoc processing
  - Extensive operational support for large scale systems
  - "Open" IMS is current direction
- DB2
  - Introduced by IBM as the main relational DBMS
  - Currently available on MVS, AS400, UNIX, NT
  - Can be accessed through IMS TM, CICS and TSO
  - Extensive operational support for large scale systems
  - Embedded SQL programming in PL1, COBOL, C

Typical UNIX DBMSs

- Many vendors (Oracle is primary)
- Well integrated CASE and decision support tools around the database
- Support for large scale OLTP and operational facilities maturing

Typical LAN and PC DBMSs: mainly Microsoft SQL (e.g., MS Access, SQL server)

Object-Oriented Databases - Overview

- Relational view not adequate for complex relationships (e.g., car design)
- OODBMS allow complex relationships (like network data model)
  - Combination of DBMS, AI and software engineering
- Data is viewed as objects:
  - Simple object is a relational table
  - Complex objects are built from simple objects
  - Each object has properties (attributes)
- Objects are inter-related through complex relationships
- Relationships may carry semantics (meaning)
  - inheritance: is-a, ako
  - aggregation: part-of
- DDL may use inheritance to create new objects
- DML can be customized
  - general query language
  - use AI pattern matching
- Can store procedures with the data
- Limitations: No agreed upon data model, no agreed upon query language
- Knowledgebases: Store data plus rules
- Objectstore is a popular OODBMS

Object-Oriented Database Manifesto

OODatabase Conference (Kyoto, Japan,1989)

Object-oriented Features

- Complex objects
- Object identity
- Encapsulation
- Types and classes
- Inheritance
- Overriding, overloading, and late binding
- Computational completeness
- Extensibility

Database Features

- Persistence
- Secondary storage management
- Concurrency
- Recovery
- Ad hoc queries

OBJECT ORIENTED SYSTEMS

- View the world in terms of objects ("things")
- Easier to think of systems as objects (customer, shipping clerk,,)
- main driver: reusability

Example; program with 1 million line of code

effort and cost:

Object-orientation appears as:

- Object-oriented user interfaces (90% of new projects)
- Object-oriented programming (80% of new projects)
- Object-oriented databases (5% or less)
- OO analysis and design (30 to 40%)

Trends:

- Business objects
- Components (VB, JB, EJB)

Object-Oriented Concepts

Definition: Objects are data guarded by a protective layer of code (Methods)

- Key concepts of the object-oriented paradigm:

– object

– attributes of objects

-- methods of object show behaviour (what it can provide)

– class : template of objects

-- inheritance (lower level objects can inherit from higher level)

Example: items in a department store :

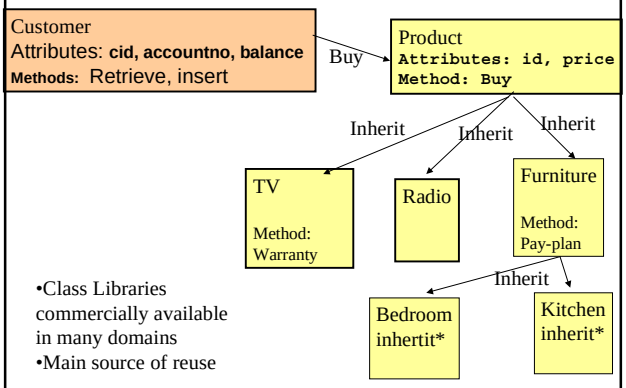
– TV is an object. The attributes of this object are model, year, cost, etc.The methods, for store are, purchase, repair

– Radio is another object, so is a dishwasher

- many of these objects have similar properties (attributes, methods)

-- can define a hierarchy .

OO Example



- Class Libraries commercially available in many domains
- Main source of reuse

Typical OO Program Layout

```
class customer {                        // class definition
        int cid, accountno, balance;    // private variables
    public
            retrieve (int,  int);
             insert (int, int, int);          }
class product  {                  // superclass
            id, price, ,,,.    // parameters
     public
            buy  (customer::cid)  // method indicating that a customer buys TV}
class Radio: product { // dervived class, inherits from product}
class TV: product {      // derived class; inherits from product
         int warrantor, ..... //   parameters
     public
              getwarranty ()   // method
//  Methods
 retrieve (int,  int);    {    ........  }
insert (int, int, int);     {    ........  }
Main  ....   The code to initialize objects and perform operations
```


Attempts to make Databases OO

- Motivation:
  - Can store, retrieve, manipulate objects
  - Most programs work on objects - should not have to translate between object and relational views
  - RDBMS may be too simple for complex objects
- Approaches;
  - BLOB (binary large objects) support in RDBMS
  - ORDBMS
    - basic RDBMS
    - support OO views
  - Full OODBMS
    - Not very succesfull (why not?)

A Framework for Analysis