NATIONAL OPEN UNIVERSITY OF NIGERIA

SCHOOL OF SCIENCE AND TECHNOLOGY

COURSE CODE: DAM 461

COURSE TITLE: STATISTICAL DATABASE SYSTEM.

COURSE CODE: DAM 461

COURSE TITLE: STATISTICAL DATABASE SYSTEM.

COURSE GUIDE

DAM 461

Statistical Database System.

Course Developer/Writer	Dr. ADEKUNLE, Yinka A. Computer Science & Mathematics Babcock University, Ilishan-Remo, Ogun State, Nigeria.
Programme Leader	
Course Coordinator	ONWODI, Greg. O

The study units in this course are as follow:

Module 1 Fundamentals of Database Systems

Unit 1 Databases and Database Users

Unit 2 Database System Concepts and Architecture

Unit 3 Data Modelling Using the Entity-Relationship Model

Module 2 The Statistical database system

Unit 1 Statistical Database Concepts

Unit 2 Statistical Data Analysis, Mining and Decision Tree

Unit 3 Computer Security and Statistical Databases

Module 3 Application of Statistical Database System

Unit 1 SPEA SMART Airport Statistical Data Management System (SMART STAT)

Module 1 Fundamentals of Database Systems

Unit 1 Databases and Database Users

1.0	Introduction
2.0	Objective
3.0	Definitions and functions
3.1	Characteristics of the Database Approach
3.2	Actors on the Scene
3.3	Advantages of Using a Database Management System
3.4	Implications of the Database Approach
3.5	When Not to Use a Database Management System
4.0	Conclusion
5.0	Summary
6.0	Tutor Marked Assignment
7.0	Further Reading and Other Resources

1.0 Introduction

In our life we have to remember so much of data. And it is easier for us to remember all information for a few individuals but it is too difficult for us to memorise all these information for a large number of individuals. There is simply too much data to be managed in the minds and in order to store all the new information, humanity invented the technology of writing.

Then, in a Competitive and global economy - data resources are essential for survival, Information required for competitive initiatives. Accuracy & timeliness very important, Managers must understand the competitive advantages available through innovative use data. To this end, Databases should be understood within the larger context of Information Resources Management. The concept that information is a major corporate resource and must be managed using the same basic principles used to manage other assets (e.g. employees, materials, equipment, financial resources).

2.0 Objectives

At the end of this unit, you should be able to:

- ❖ explain the concept of database
- ❖ appreciate the features of database management system (DBMS)
- ❖ State the major advantages of database approach
- ❖ Draw the major components of database environment

3.0 Definitions and Functions

Data are known facts that can be recorded, and have an implicit meaning. (e.g., names, telephone numbers, addresses, etc). Data are raw facts concerning things such as people, objects, or events.

Information on the other hand is data that have been processed and presented in a form suitable for human interpretation, often with the purpose of revealing trends or patterns.

What is a Database?

A database is a collection of related information (data) in a structured way. It is simply a bunch of information (data) stored on a computer. This could be a list of all your clients, a list of the products you sell, the results of a chess tournament or everyone in your family tree. The most common type of database is a relational database. Relational databases consist of tables of data with clearly defined columns. Database could also be a piece of software used to store data. This is because the word "database" can refer to both the software and the actual data.

What is a DBMS and what are its functions?

A DBMS *(Database Management System)* is best described as a collection of programs that manage the database structure and that control shared access to the data in the database. It is a software package to facilitate the creation and maintenance of a computerized database.Current DBMSs also store the relationships between the database components; they also take care of defining the required access paths to those components.

The functions of a current generation DBMS may be summarized as follow:

1. Stores the definitions of data and their relationships (metadata) in a data dictionary; any changes made are automatically recorded in the data dictionary.
2. Creates the complex structures required for data storage.
3. Transforms entered data to conform to the data structures in item 2.
4. Creates a security system and enforces security within that system.
5. Creates complex structures that allow multiple user access to the data.
6. Performs backup and data recovery procedures to ensure data safety.
7. Promotes and enforces integrity rules to eliminate data integrity problems.
8. Provides access to the data via utility programs and from programming languages interfaces.
9. Provides end-user access to data within a computer network environment.

3.1 Characteristics of the Database Approach

There are two approaches to database system, which include:

- ❖ **The Traditional file Processing (File based)** approach: Under this approach, each user defines and implements the files needed for a specific application. The approach is characterized by **redundancy** (duplication) in defining and storing data, inconsistencies or various versions of data, and wastage of storage space.
- ❖ The **database** approach: This approach emphasizes the integration and sharing of data across the organization. It is characterized by a single repository of data (i.e. data located at a single site).

Characteristics of database approach:

- ❖ Self-Describing nature of a database system: complete definition or description of the database structure (structure of each file, type & storage format of each data item), and constraints on data are stored in the system catalogue and the information stored in the catalogue is called **meta-data.**
- ❖ Insulation between Programs and Data (structure): This is called program-data independence or program-operation independence; the characteristics that allow changing data storage structures without having to change the DBMS access programs.
- ❖ Data Abstraction: A situation in which **data model** is used to hide storage details and present the users with a conceptual view of the database.
- ❖ Multiple views of the data: Each user may see a different view of the database, which describes only the data of interest to that user.
- ❖ Sharing of data and multiuser transaction processing: concurrency control (on-line transaction processing (OLTP)) application.

3.2 Actors on the Scene

These include Persons whose job involves daily use of a large database, such persons as**:**

- **Database Administrator (DBA):** This includes person's administering database, DBMS and related software. That is, persons responsible for managing the database system, authorizing access, coordinating and monitoring uses, acquiring resources (database and DBMs).
- **Database designers:** These are persons responsible for designing the database, identifying the data to be stored, choosing the appropriate structures to represent and store this data, and interacting with users**.**
- **End Users:** The persons that use the database for querying, updating, generating reports, etc.

  Casual end users: Occasional users.(middle- or high-level managers)

  Parametric (or naive) end users: They use pre-programmed *canned transactions* to interact continuously with the database. For example, bank tellers or reservation clerk.

  Sophisticated end users: Use full DBMS capabilities for implementing complex applications.

  Stand-alone users (personal databases).
- **System Analysts and Application Programmers (Software engineers):** These are people that determine requirements of end users (specification), implement, test, debug and document.

Workers behind the scene

- **DBMS system designers and implementers:** These are people who design and implement the DBMS modules and interfaces as a software package.
- **Tool developers: Persons who** develops software packages and tools that facilitate database system design and use, and help improve performance. Tools include design tools, performance tools, special interfaces, etc.
- **Operators and maintenance personnel:** Work on running and maintaining the hardware and software environment for the database system.

3.3 Advantages of Using a DBMS

- Controlling redundancy in data storage and in development and maintenance efforts, thereby eliminating duplication of efforts, space wastage and inconsistence.
- Restricting unauthorized access (security and authorization) .
- Providing persistent storage for program objects and data structures: Object-oriented database. OODB are compatible with C++, Java.
- Permitting inference and actions using rules.
- Providing multiple user interfaces, backup and recovery.
- Representing complex relationships among data.
- Enforcing integrity constraints.

- Saving time and aiding communication: The database is a more efficient solution than paper files held in a file folder. Then, larger companies can benefit from databases when information must be spread to various users.
- Databases Are Inexpensive Managers: Smaller businesses are always looking for ways to cut costs without cutting quality. A database can be a hefty investment initially, but, over the long term, it will save money by improving the efficiency of all employees, impressing customers who will not need to repeat their information and saving on paper costs.

3.4 Implications of the Database Approach

The followings are the implications of using database approach in an organisation:

- Potential for Enforcing Standards.
- Reduced Application Development Time.
- Flexibility.
- Availability of Up-to-date Information.
- Economies of Scale.

3.5 When Not to Use DBMS Main Costs of Using a DBMS:

- ❖ When there is high initial investment in hardware, software, training and possible need for additional hardware.
- ❖ When there is high overhead for providing generality, security, recovery, integrity, and concurrency control.
- ❖ Generality that a DBMS provides for defining and processing data.

When a DBMS may be unnecessary:

- ➢ If the database and applications are simple, well defined, and not expected to change. If there are stringent real-time requirements that may not be met because of DBMS overhead.
- ➢ If access to data by multiple users is not required.

When no DBMS may suffice:

- ➢ If the database system is not able to handle the complexity of data because of modeling limitations.
- ➢ If the database users need special operations not supported by the DBMS.

4.0 Conclusion

A database is a collection of related information (data) in a structured way. While database management is a collection of programs that manage the database structure and that control shared access to the data in the database, with different characteristics and users, as well as implications.

5.0 Summary

In this unit we have learnt that:

- ❖ **A** database is a collection of related information (data) in a structured way.
- ❖ Database management is a collection of programs that manage the database structure and that control shared access to the data in the database.
- ❖ There are two approaches to data management with different characteristics and users.
- ❖ The advantages include controlling redundancy, restricting unauthorized access, saving time etc, with flexibility, economies of scale, and potential for enforcing standards, are some of the implications of database approach.

6.0 Tutor Marked Assignment

1. (a) Differentiate between the following:
   (i) Data and information (ii) Database and database management.

   (b) What are the functions of database management?

2 . (a) What are the characteristics of database approach?

(b) Mention the advantages of using database management.

7.0 Further Reading and Other Resources

Relational Database Data www.OLDI.com Automatic RDB Data Logging Enterprise Transaction Modules

ACL Data AnalysisData analysis made easier with ACL's world leading Auto Database RecoveryReal-time database recovery enables 24-7 availability to essential data.

Databases in the cloud: a work in progress; October 2009; ISBN 978-1-60558-765-3.

Advantages of Relational Databases

Database Administrator Job Description

Introduction To Relational Data Model

Database Design As little or as many leads as you need. Check this out! ChainStoreGuide.com/Data

Database Management WinSQL - A Homogeneous Solution for Heterogeneous Environment. www.synametrics.com

The Advantages of Using a Database by: Jacqueline Thomas

Data Masking Softwarewww.orpheus-it.com Anonymize your sensitive data Protect personal and company data

Database/SQL Toolwww.dbvis.com For DB2, SQL Server, Derby, Mimer Informix, Oraclé and more

Sample Database?www.ioglobal.net Hosted essay data management: Safe, Efficient QA, Global 24/7 access.

Database Mgmt & Design Hansen & Hansen

MS-Access, MS FoxPro etc. manuals, books

MIS Demo Databases (Access, Lotus Notes, Paradox)

A Database Wizard for accessing ODBC-compliant databases from Web applications (database-enabled Web applications).

Module 1 Fundamentals of Database Systems

Unit 2 Database System Concepts and Architecture

1.0 Introduction

2.0 Objective

3.0 Data Models and Their Categories

3.1 History of Data Models

3.2 Schemas, Instances, and States

3.3 Data and the Three-Schema Architecture

3.4 Data Independence

3.5 DBMS Languages and Interfaces

3.6 The Database System Environment

3.7 Database System Utilities and Tools

3.8 Centralized and Client-Server Architectures

3.9 Classification of DBMSs
4.0 Conclusion
5.0 Summary
6.0 Tutor Marked Assignment
7.0 Further Reading and Other Resources

1.0 Introduction

The purpose of this Unit is to introduce students to the database approach to information systems development, and to the important concepts and principles of this approach. This is highly imperative because, it should convey a sense of the central importance of databases in today's information systems environment. The idea of an organizational database is intuitively appealing to most students.

However, many students have little or no background or experience of databases. Others have had some experiences with a PC database (such as Microsoft Access), and consequently have a limited perspective concerning an organizational approach to databases. Therefore in this Unit, the basic concepts and definitions of databases are introduced.

2.0 Objective

At the end of this unit, you should be able to:

- ❖ Explain what data and database model are all about
- ❖ Mention different types of data model
- ❖ Vividly explain data and schema architecture
- ❖ Describe the data independence

❖ Explain database environment and its components etc.

3.0 Data Models and Their Categories

Data Model

This is a set of concepts used to describe the structures of a database, and the operations for manipulating these structures, as well as certain constraints that the database should obey. It is simply a diagram that describes the most important “things” in business environment from a data-centric point of view. For example, an **Entity Relationship Diagram** (ERD) below describes the relationship between the data stored about products, and the data stored about the organizations that supply the products.

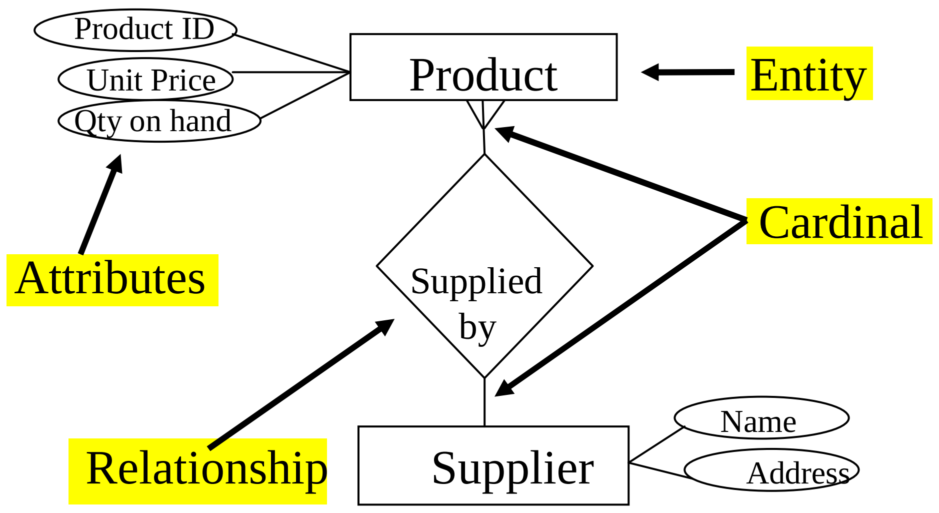


FIGURE 1: An ERD showing a relationship between products and suppliers.

Data Model Structure and Constraints:

Constructs are used to define the database structure. Constructs typically include elements (and their data types) as well as groups of elements (e.g. entity, record, table), and relationships among such groups. Constraints specify some restrictions on valid data; these constraints must be enforced at all times.

Data Model Operations:

These operations are used for specifying database retrievals and updates by referring to the constructs of the data model. Operations on the data model may include basic model operations (e.g. generic insert, delete, and update) and user-defined operations (e.g. compute_student_gpa, update_ inventory).

Database Model

This is a theory or specification describing how a database is structured and used. A database model is a collection of logical constructs used to represent the database's data structure as well as the data relationship(s) found within that structure. Several such models have been suggested which include:

- **Flat model:** This may not strictly qualify as a data model. The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another.

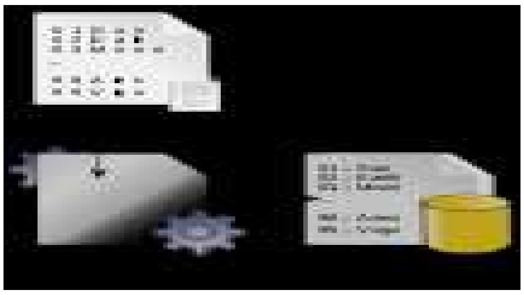


Figure 2 **Flat model**

- **Hierarchical model:** In this model data is organized into a tree-like structure, implying a single upward link in each record to describe the nesting, and a sort field to keep the records in a particular order in each same-level list.

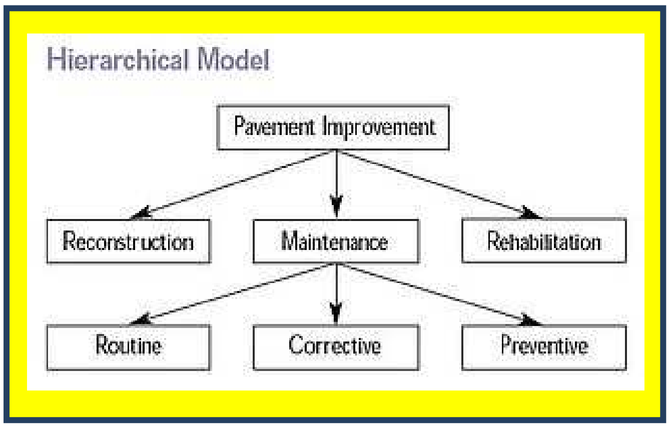


Figure 3 **Hierarchical model**

- **Network model:** This model organizes data using two fundamental constructs, called records and sets. Records contain fields, and sets define one-to-many relationships between records: one owner, many members.

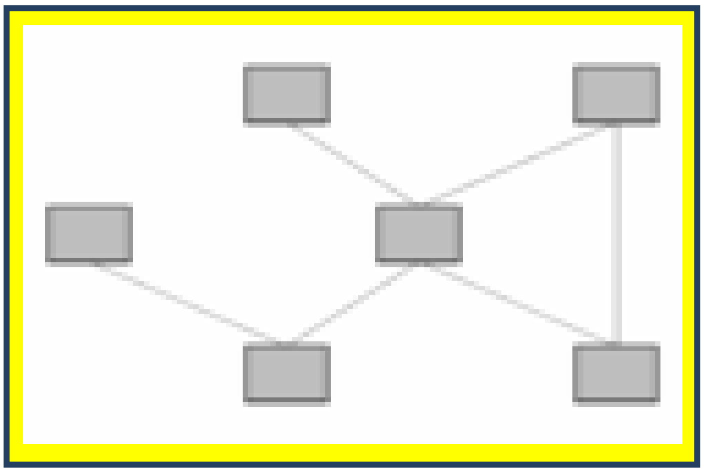


Figure 4 **Network model**

- **Relational model:** This is a database model based on first-order predicate logic. Its core idea is to describe a database as a collection of predicates over a finite set of predicate variables, describing constraints on the possible values and combinations of values.

Figure 5 **Relational model**

- **Object-relational model:** This is similar to a relational database model, but objects, classes and inheritance are directly supported in database schemas and in the query language.

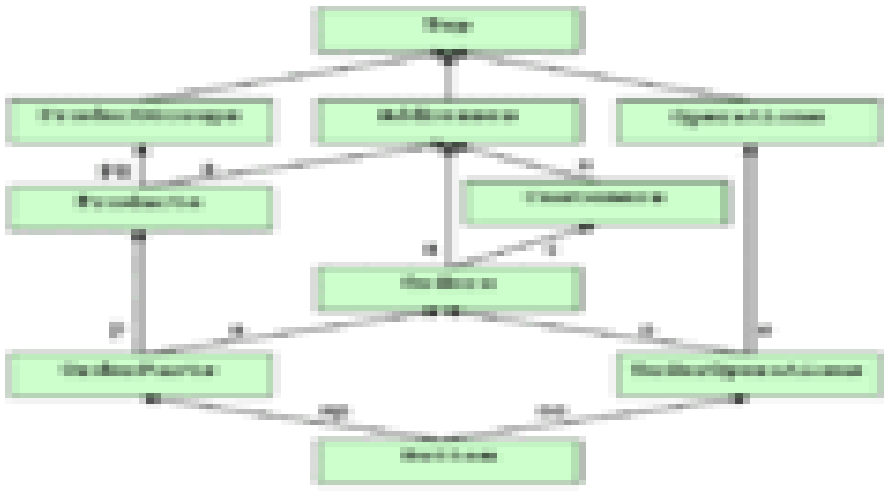
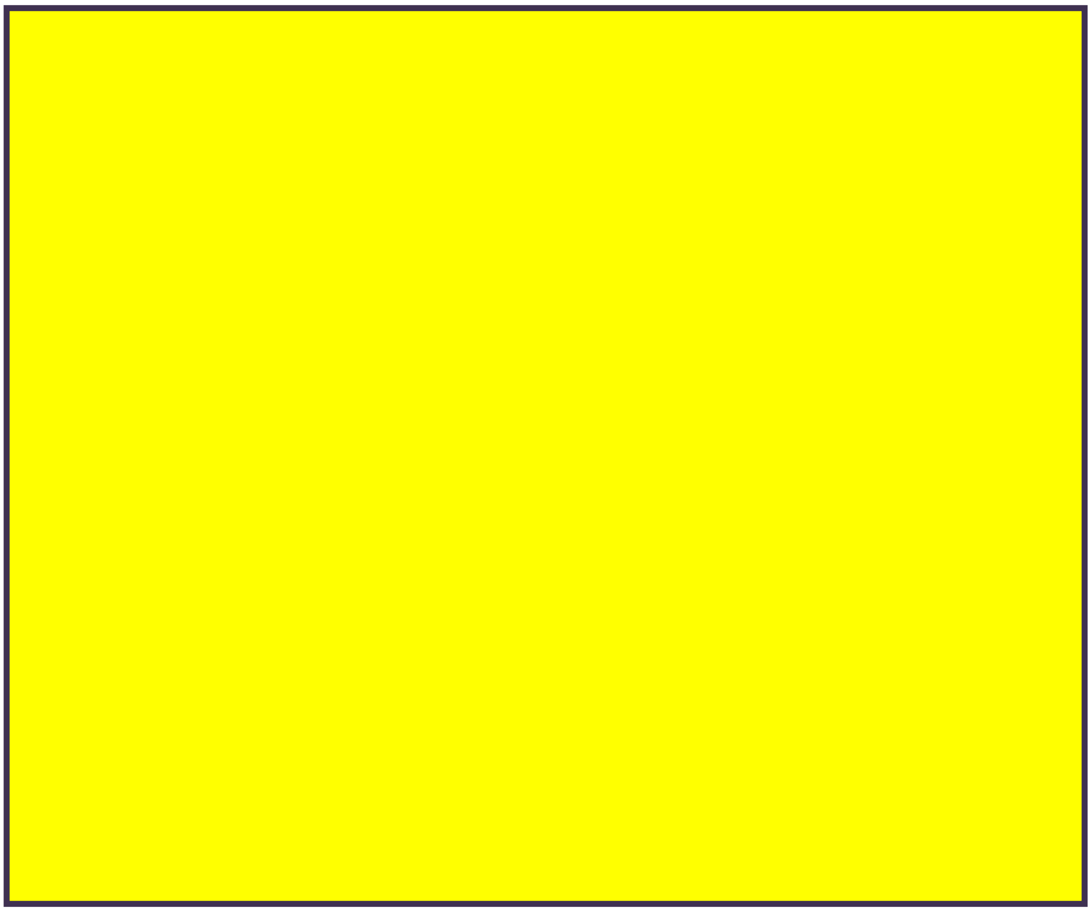


Figure 6 **Concept-oriented model**

- **Star schema**: This is the simplest style of data warehouse schema. The star schema consists of a few "fact tables" (possibly only one, justifying the name) referencing any number of "dimension tables". The star schema is considered an important special case of the snowflake schema.



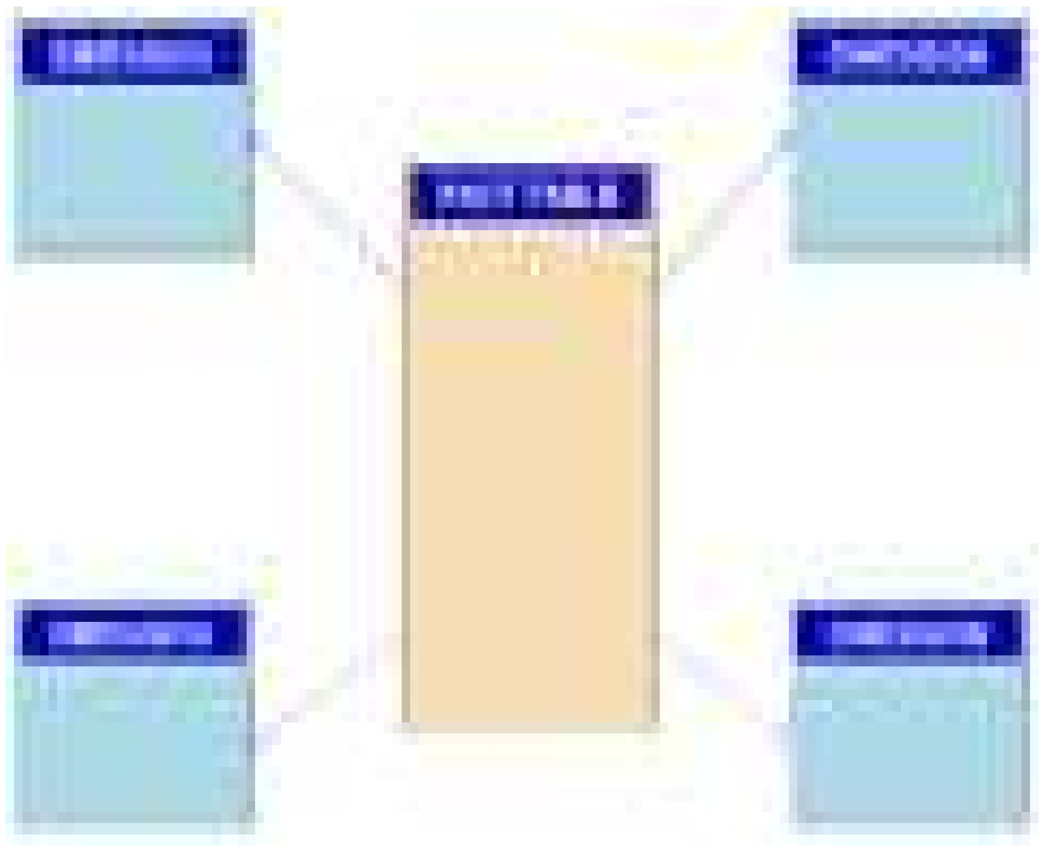


Figure 7 **Star schema**

Categories of Data Models:

Conceptual (high-level, semantic) data models: Provide concepts that are close to the way many users perceive data (also called entity-based or object-based data models).

Physical (low-level, internal) data models: Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals.

Implementation (representational) data models: Provide concepts that fall between the conceptual and physical models, used by many commercial

DBMS implementations (e.g. relational data models used in many commercial systems).

3.1 History of Data Models

One of the earliest pioneering works in modelling information systems was done by Young and Kent (1958), who argued for "a precise and abstract way of specifying the informational and time characteristics of a data processing problem". They wanted to create "a notation that should enable the analyst to organize the problem around any piece of hardware". Their work was a first effort to create an abstract specification and invariant basis for designing different alternative implementations using different hardware components.

The next step in IS modelling was taken by CODASYL, an IT industry consortium formed in 1959, who essentially aimed at the same thing as Young and Kent: the development of "a proper structure for machine independent problem definition language, at the system level of data processing". This led to the development of a specific IS information algebra.

In the 1960s data modelling gained more significance with the initiation of the management information system (MIS) concept. According to Leondes (2002), "during that time, the information system provided the data and information for management purposes. The first generation database system, called Integrated Data Store (IDS), was designed by Charles

Bachman at General Electric. Two famous database models, the network data model and the hierarchical data model, were proposed during this period of time". Towards the end of the 1960s Edgar F. Codd worked out his theories of data arrangement, and proposed the relational model for database management based on first-order predicate logic.

In the 1970s entity relationship modelling emerged as a new type of conceptual data modelling, originally proposed in 1976 by Peter Chen. Entity relationship models were being used in the first stage of information system design during the requirements analysis to describe information needs or the type of information that is to be stored in a database. This technique can describe any ontology, i.e., an overview and classification of concepts and their relationships, for a certain area of interest.

Also in the 1970s G.M. Nijssen developed "Natural Language Information Analysis Method" (NIAM) method, and developed this in the 1980s in cooperation with Terry Halpin into Object-Role Modelling (ORM). Further in the 1980s according to Jan L. Harrington (2000) "the development of the object-oriented paradigm brought about a fundamental change in the way we look at data and the procedures that operate on data. Traditionally, data and procedures have been stored separately: the data and their relationship in a database, the procedures in an application program. Object orientation, however, combined an entity's procedure with its data’’.

3.2 Schemas, Instances, and States

Schemas versus Instances:

Database Schema represents the **description** of a database, which includes descriptions of the database structure, data types, and the constraints on the database. **Schema Diagram** on the other hand, is an **illustrative** display of (most aspects of) a database schema, while **Schema Construct** is a **component** of the schema or an object within the schema, e.g., STUDENT, COURSE.

Database State is the actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database. It is also called database instance (or occurrence or snapshot). The term **instance** is also applied to individual database components, e.g. record instance, table instance, entity instance.

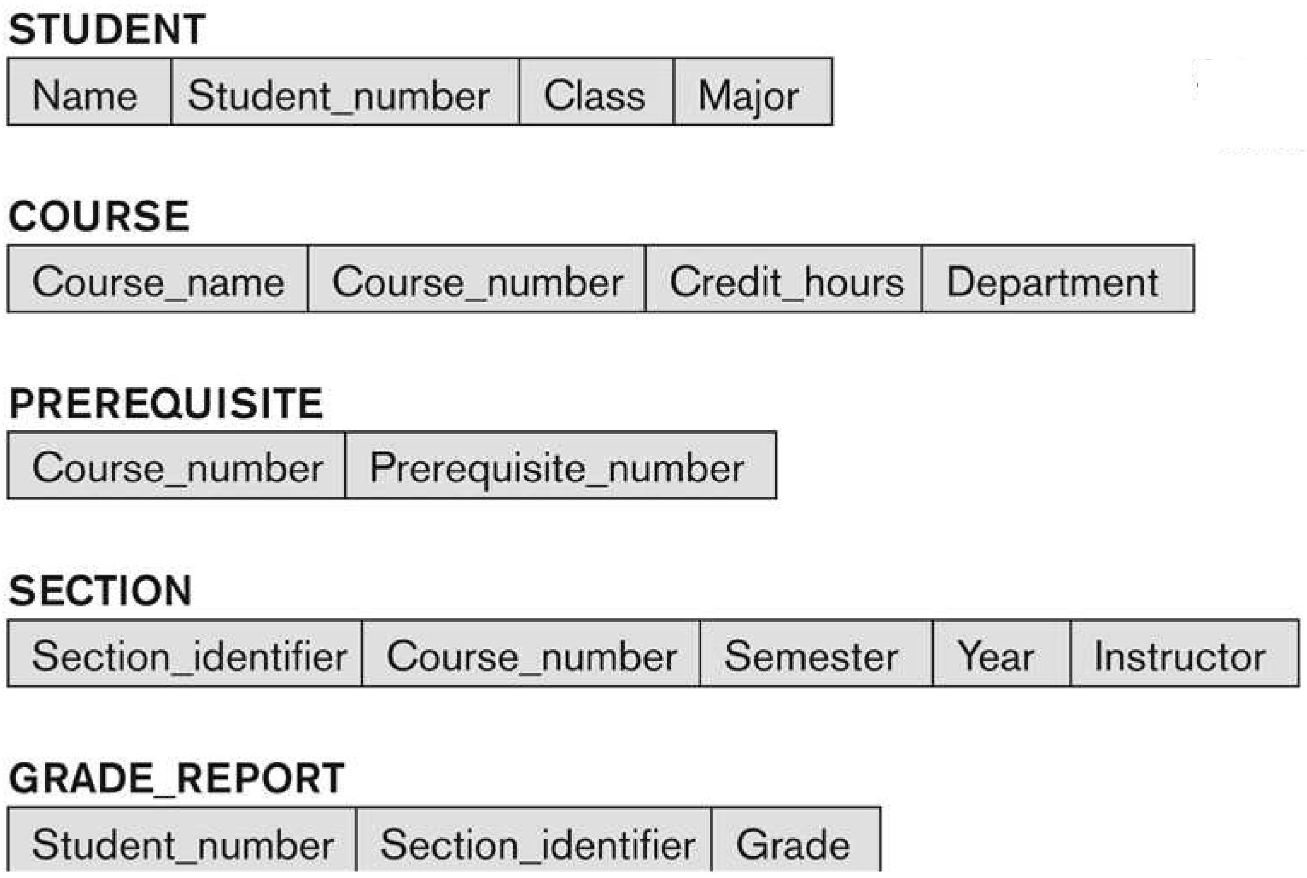
Database Schema versus Database State

Database State refers to the content of a database at a moment in time, which includes:

- ❖ **Initial Database State:** This refers to the database state when it is initially loaded into the system.
- ❖ **Valid State:** A state that satisfies the structure and constraints of the database.

The distinction to be made here is that, **database schema** changes very infrequently, whereas **database state** changes every time the database is updated. Then the **schema** is also called intension, as **state** is called extension.

Example of a Database Schema:



STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

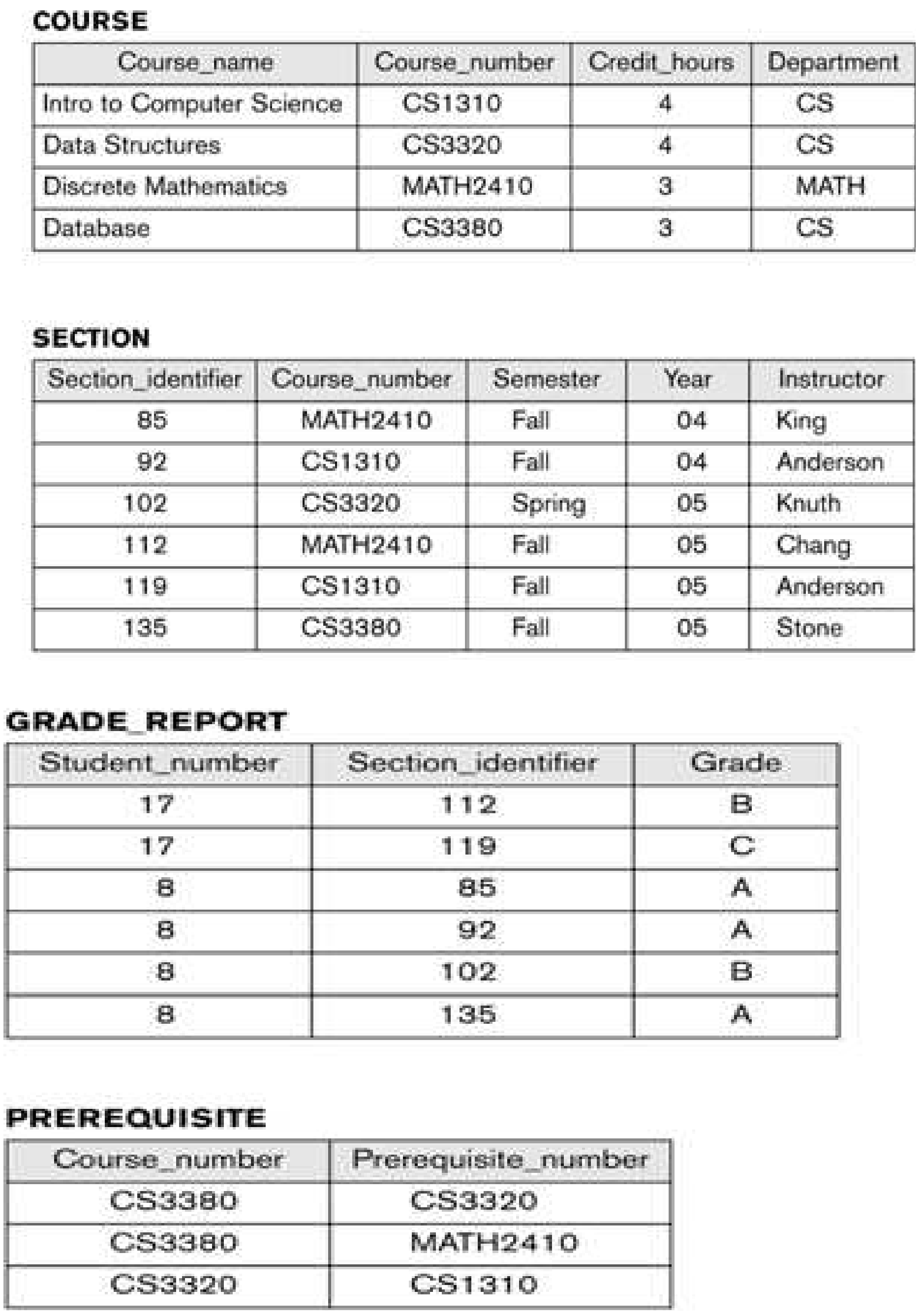
Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 8 Database Schema

Example of a database state:



COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Table 1: Database State

3.3 Data and The Three-Schema Architecture

Data architecture is the design of data for use in defining the target state and the subsequent planning needed to hit the target state. It is usually one of several architecture domains that form the pillars of an enterprise

architecture. It describes the data structures used by a business and its applications. These are descriptions of data in storage and data in motion; descriptions of data stores, data groups and data items; and mappings of those data artefacts to data qualities, applications, locations etc.
And essential to realizing the target state, Data architecture describes how data is processed, stored, and utilized in a given system. It provides criteria for data processing operations that make it possible to design data flows and also control the flow of data in the system.

Three-Schema Architecture defines DBMS schemas at three levels which include:

- **Internal schema:** This describes the physical storage structures and access paths (e.g indexes) of database, at the internal level of DBMS design. And it typically uses a physical data model.
- **Conceptual schema:** This describes the structure and constraints for the whole database, for a community of users, at the conceptual level. This uses a Conceptual or an Implementation data model.
- **External schemas**: This at the external level describes the various user views. And it usually uses the same data model as the conceptual schema.

Three-Schema Architecture is proposed to support DBMS characteristics of Program-data independence, and multiple views of the data. Although it not explicitly used in commercial DBMS products, it has been useful in explaining database system organization.

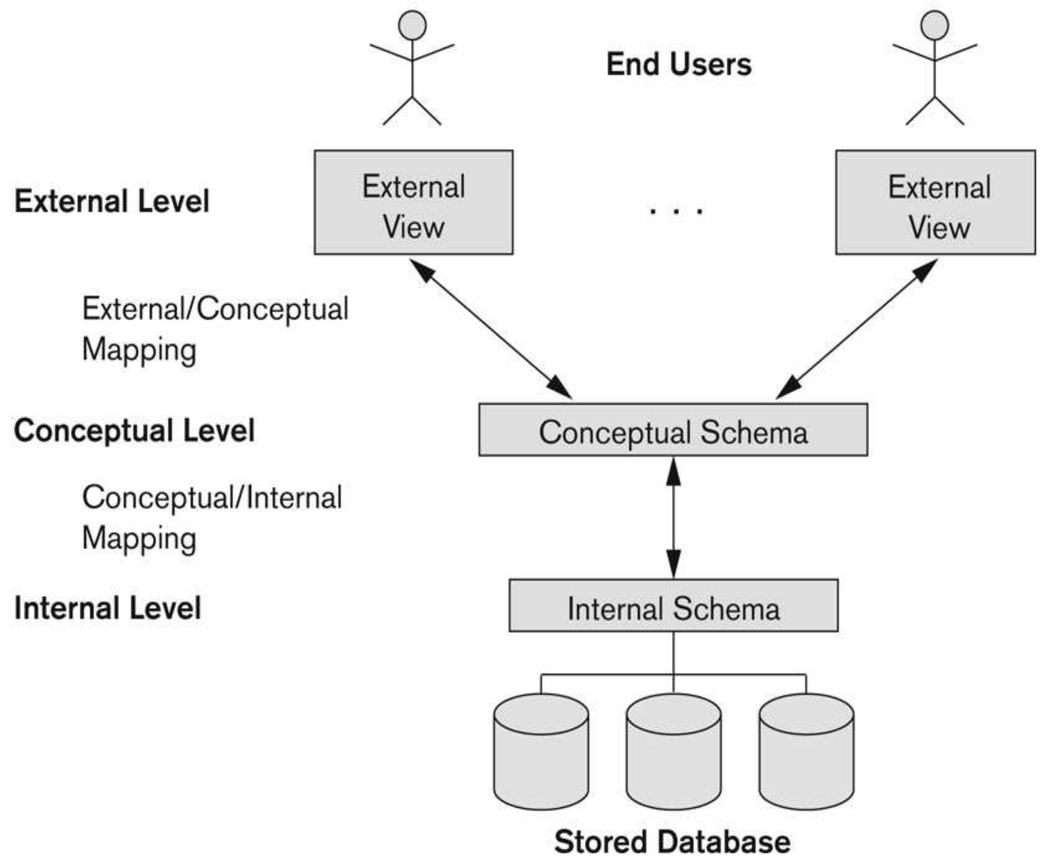


Figure 9: The Three-schema Architecture

Mappings among schema levels are also needed to transform requests and data. **Programs** refer to an external schema, and are mapped by the DBMS to the internal schema for execution. And data extracted from the internal DBMS level is reformatted to match the user’s external view (e.g. formatting the results of an SQL query for display in a Web page).

3.4 Data Independence

Data independence is the type of data transparency that matters for a centralized DBMS. It refers to the immunity of user applications to make changes in the definition and organization of data.

The physical structure of the data is referred to as "physical data description". Physical data independence deals with hiding the details of the storage structure from user applications. The application should not be involved with these issues since, conceptually; there is no difference in the operations carried out against the data. And that, the data independence and operation independence together gives the features of data abstraction.
There are two levels of data independence:
Logical Data Independence: This is the capacity to change the conceptual schema (logical) without having to change the external schemas (user views) and their associated application programs. For example, the addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.
Physical Data Independence: This is the capacity to change the internal schema (physical) without having to change the conceptual schema (logical) For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance, without having to change the conceptual or external schemas.
And with knowledge about the three-scheme architecture, the term data independence can be explained as follows: Each higher level of the data architecture is immune to changes of the next lower level of the architecture. When a schema at a lower level is changed, only the mappings between this schema and higher level schemas need to be changed in a DBMS that fully supports data independence. The higher-level schemas themselves are

unchanged. Hence, the application programs need not be changed since they refer to the external schemas.

3.5 DBMS Languages and Interfaces

DBMS Languages are of two categories which include:

- ❖ Data Definition Language (DDL)
- ❖ Data Manipulation Language (DML): This can either be

High-Level or Non-procedural Languages: These include the relational language SQL, which may be used in a standalone way or may be embedded in a programming language.
Low Level or Procedural Languages: These must be embedded in a programming language.

Data Definition Language (DDL): This is the language used by the database and database designers to specify the conceptual schema of a database. In many DBMSs, the DDL is also used to define internal and external schemas (views). In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas. SDL is typically realized via DBMS commands provided to the DBA and database designers.

Data Manipulation Language (DML): This is the language used to specify database retrievals and updates**.** DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java. A library of functions can also be provided to access the DBMS from a programming language. Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

For example, the SQL relational languages are “set”-oriented and specify what data to retrieve rather than how to retrieve it. It is also called **declarative** languages.

Low Level or Procedural Language: This is used to retrieve data, one record-at-a-time; constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

Stand-alone query language interfaces

- Example: Entering SQL queries at the DBMS Interactive SQL interface (e.g. SQL*Plus in ORACLE).Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
- Menu-based, forms-based, graphics-based, etc.

DBMS Programming Language Interfaces

Programmer interfaces for embedding DML in a programming languages:
Embedded Approach e.g. embedded SQL (for C, C++, etc.), SQLJ(for Java)

Procedure Call Approach e.g. JDBC for Java,

ODBC for other programming languages

Database Programming Language Approach:

e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components.

User-Friendly DBMS Interfaces

Menu-based, popular for browsing on the web.

Forms-based, designed for naïve users.

Graphics-based:(Point and Click, Drag and Drop, etc.).

Natural language: requests in written English.

Combinations of the above: For example, both menus and forms used extensively in Web database interfaces.

Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA: Creating user accounts, granting authorizations
  Setting system parameters Changing schemas or access paths.

3.6 The Database System Environment

The DBMS is an important component of the database environment. This environment also includes different types of hardware and software, people who perform different functions within the environment, procedures designed to accomplish desired activities, and data. The data constitute the database's central component through which information is generated. So, the main purpose of the database environment is to help an organization to perform its mission and to achieve its goals.

It is worthy of note that, the database administrator writes and enforces the procedures and standards that are then used by designers, analysts, programmers, and end users. The end users use the application programs created by analysts and programmers. In turn, the application programs make use of the DBMS, which manages the data. Note also that the database designer and the database administrator perform their jobs through the interface – that would be shown on the computer. The interface is the gateway to the database, which resides within the hardware. Finally, the System administrator manages the entire system.

Components of DB environment:

- ❖ **CASE tools :** Automated tools to design DBs & apps. programs.
- ❖ **Repository :** Centralised knowledge base containing all data definitions, screen & report formats & definitions**.**
- ❖ **DBMS :** Commercial S/W used to create, maintain & provide controlled access to the DB & the repository (definitions of data).
- ❖ **DB :** A shared collection of data, designed to meet the information needs of users in an organisation (occurrences of data)**.**
- ❖ **Application programs :** Computer programs that are used to create & maintain DB & provide information to users**.**

- **User Interface:** Languages, menus, and other facilities by which users interact with various system components, such as CASE tools, application programs, the DBMS and the repository.
- **Data administrators:** Responsible for overall information resources of an organization .
- **System developers:** System analysts, programmers who design new application programs.
- **End users:** add, delete & modify data in DBs & receive information from it throughout the organization .

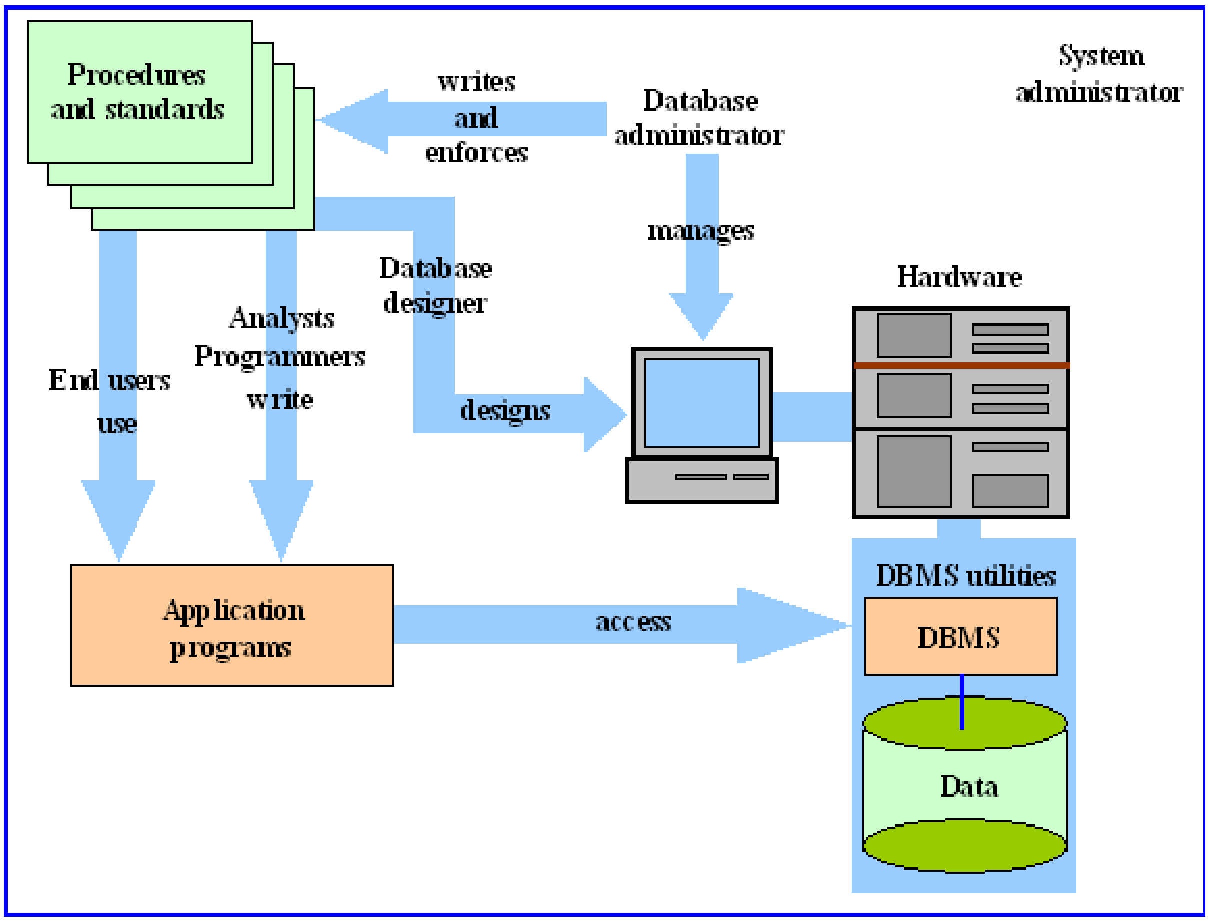


Figure 10: The Database Environment Components

3.7 Database System Utilities and Tools

Database System Utilities: These are used to perform certain functions such as:

- Loading data stored in files into a database. Includes data conversion tools.
- Backing up the database periodically on tape.
- Reorganizing database file structures.
- Report generation utilities.
- Performance monitoring utilities.
- Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools: These include data dictionary / repository, used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.

Active data dictionary is accessed by DBMS software and users/DBA.

Passive data dictionary is accessed by users/DBA only.

Application Development Environments and CASE (computer-aided software engineering) tools:

Examples: PowerBuilder (Sybase) , JBuilder (Borland), JDeveloper 10G (Oracle), Databrid, DTM ODBC Manager, mysql data syncronization, myDBR, AeroSQL, Data-tier Applications etc.

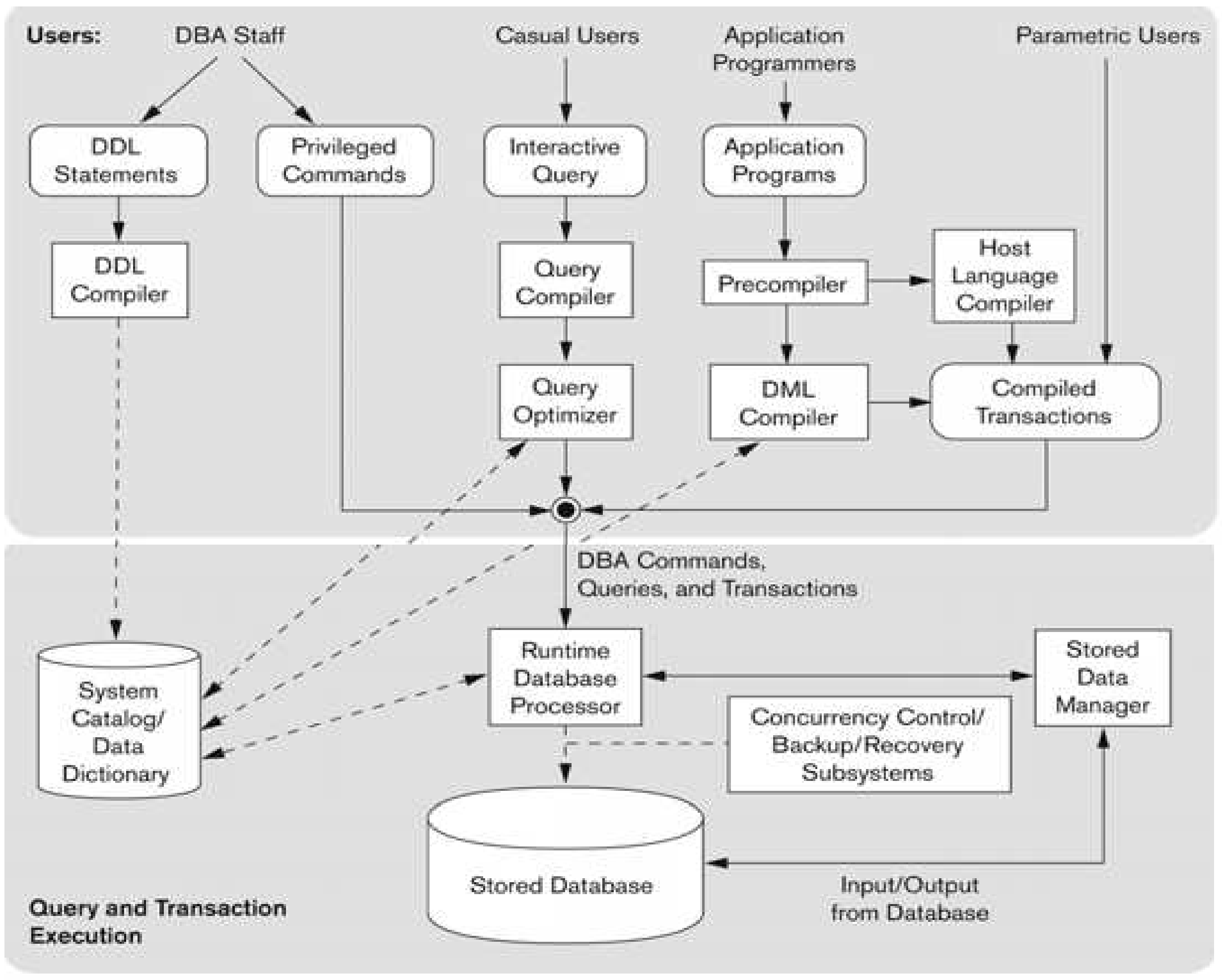


Figure 11: Typical DBMS Component Module

3.8 Centralized and Client-Server Architectures

Centralized DBMS: Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software. User can still connect through a remote terminal – however, all processing is done at centralized site.

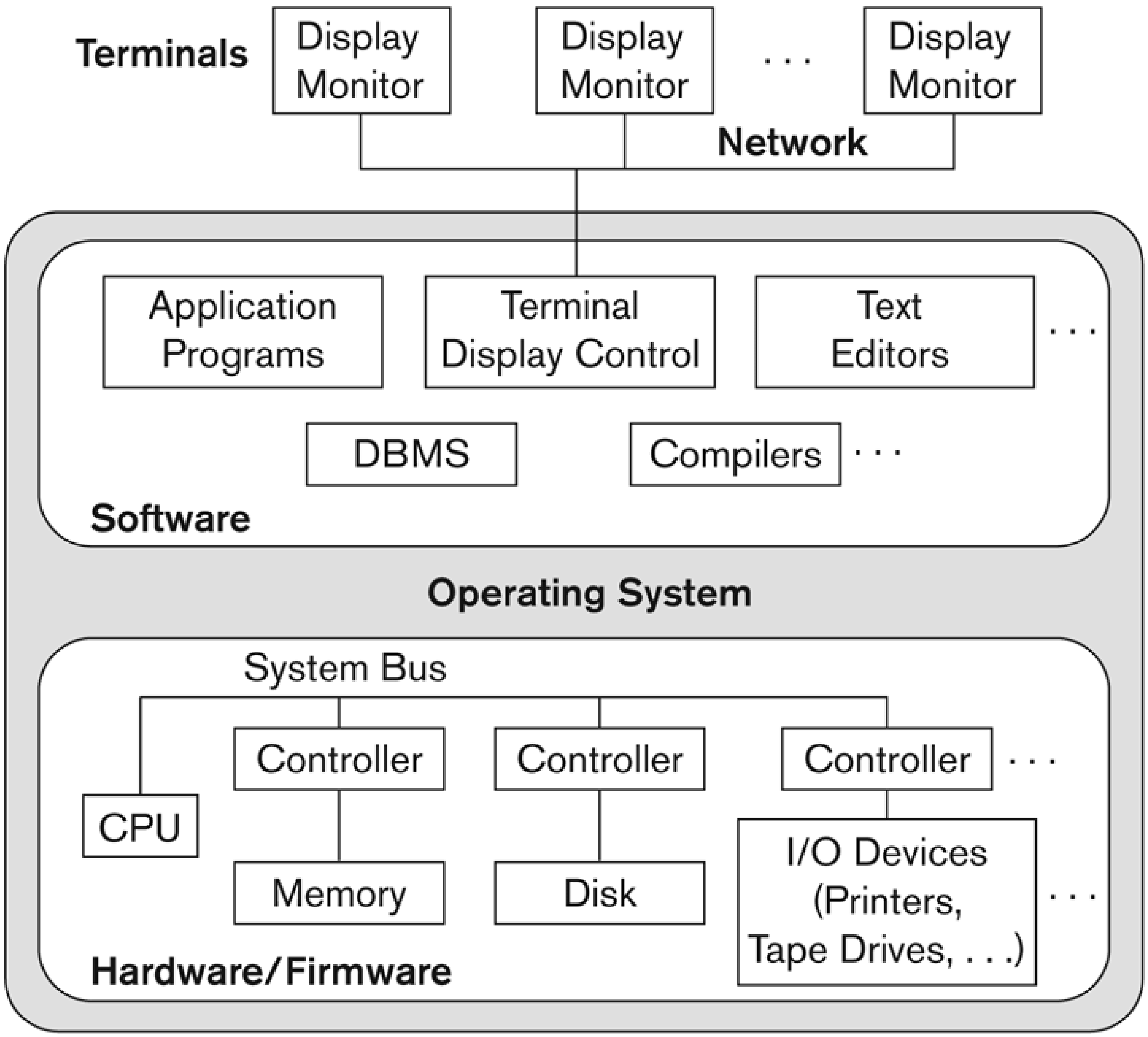
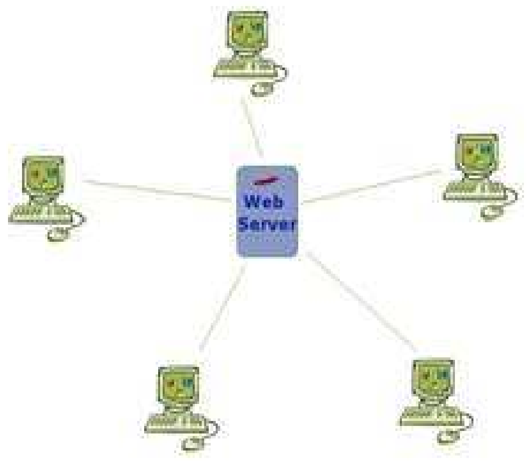


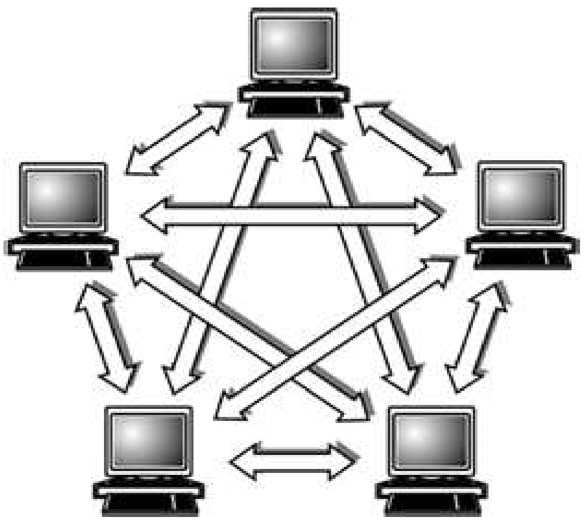
Figure 12: A Physical Centralized Architecture

Client-server: This is a software architecture model consisting of two parts, client systems and server systems, both communicate over a computer network or on the same computer. A client-server application is a distributed system consisting of both client and server software. The client process always initiates a connection to the server, while the server process always waits for requests from any client.

When both the client process and server process are running on the same computer, this is called a single seat setup but in peer-to-peer, each host or application instance can simultaneously act as both a client and a server (unlike centralized servers of the client-server model) and because each has equivalent responsibilities and status. Peer-to-peer architectures are often abbreviated using the acronym P2P.



A one **server** based network.



A **peer-to-peer** based network.

Figure 13: Client Server Networks

Basic 2-tier Client-Server Architectures

Specialized Servers with Specialized functions

- Print server
- File server
- DBMS server
- Web server

 Email server

Clients can access the specialized servers as needed

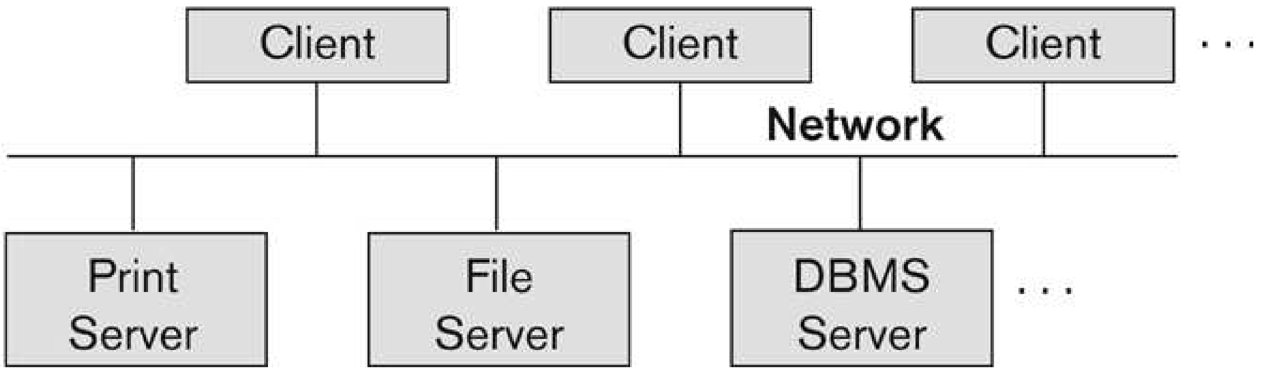


Figure 14 : Logical two-tier client server architecture

Two Tier Client-Server Architecture:

This is a client program that may connect to several DBMSs, sometimes called the data sources. In general, data sources can be files or other non-DBMS software that manages data. Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Three Tier Client-Server Architecture:

This is very common for Web applications**, I**ntermediate Layer called Application Server or Web Server. Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server. Acts like a conduit for sending partially processed data between the database server and the client.

Three-tier Architecture Can Enhance Security:

- Database server only accessible via middle tier
- Clients cannot directly access database server

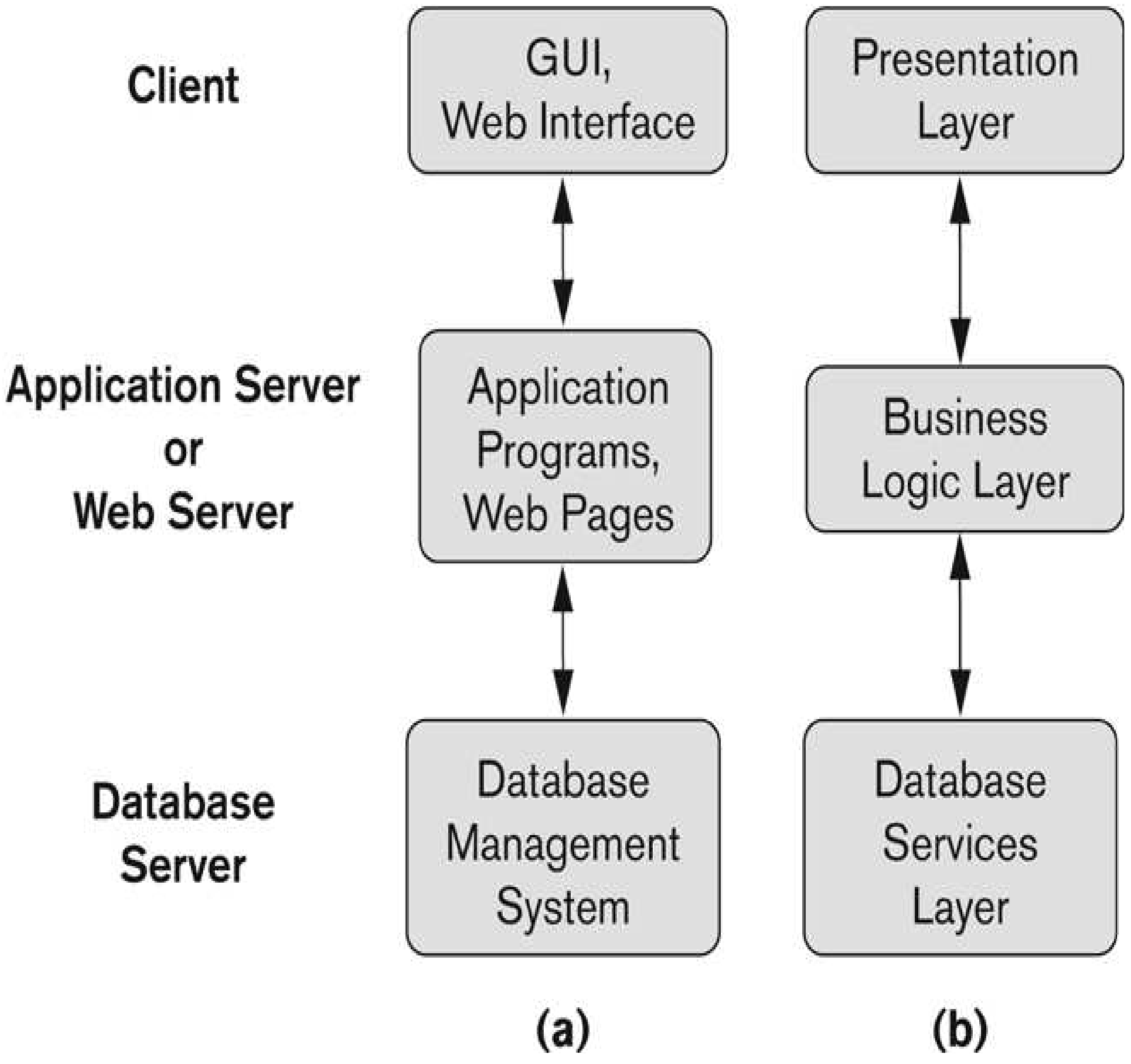


Figure 15: Three-tier client-server architecture

Clients: The **client–server model** of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
It provides appropriate interfaces through a client software module to access and utilize the various server resources. Clients may be diskless machines or PCs or Workstations with disks with only the client software installed. It is connected to the servers via some form of a network (LAN: local area network, wireless network, etc.)

Clients characteristics

- Always initiates requests to servers.
- Waits for replies.
- Receives replies.
- Usually connects to a small number of servers at one time.
- Usually interacts directly with end-users using any user interface such as graphical user interface.

DBMS Server provides database query and transaction services to the clients. Relational DBMS servers are often called SQL servers, query servers, or transaction servers. Applications running on clients utilize an **Application Program Interface** (**API**) to access server databases via standard interface such as:

- ODBC: Open Database Connectivity standard
- JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDB

Server characteristics

- Always wait for a request from one of the clients.
- Serve clients requests then replies with requested data to the clients.
- A server may communicate with other servers in order to serve a client request.

3.9 Classification of DBMSs

Based on the data model used:

❖ Traditional: Relational, Network, Hierarchical.

❖ Emerging: Object-oriented, Object-relational.

 Other classifications

❖ Single-user (typically used with personal computers) vs. multi-user (most DBMSs).

❖ Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

4.0 Conclusion

With the introduction to concepts of database system, students must have acquired the basic sense of the central importance of databases in today's information systems environment. And with this, they should be able to function very well in such environment.

5.0 Summary

In this unit we have learnt that:

- ❖ Data model is a set of concepts used to describe the structures of a database, and the operations for manipulating these structures, as well as certain constraints that the database should obey.
- ❖ Database models include flat, hierarchical network, relational**,** concept-oriented and star-schema.
- ❖ Data architecture is the design of data for use in defining the target state and the subsequent planning needed to hit the target state.
- ❖ Database system environment includes different types of hardware and software, people who perform different functions within the environment, procedures designed to accomplish desired activities, and data.

6.0 Tutor Marked Assignment

1. (a) Define data and database model

   (b) What do you understand by data independence and its levels

2. (a) Explain database environment and its components

7.0 Further Reading and Other Resources

Paul R. Smith & Richard Sarfaty (1993). Creating a strategic plan for configuration management using Computer Aided Software Engineering

(CASE) tools. Paper For 1993 National DOE/Contractors and Facilities CAD/CAE User's Group.

Data Modeling Made Simple 2nd Edition", Steve Hoberman, Technics Publications, LLC 2009

Michael R. McCaleb (1999). "A Conceptual Data Model of Datum Systems". National Institute of Standards and Technology. August 1999.

Matthew West and Julian Fowler (1999). Developing High Quality Data Models. The European Process Industries STEP Technical Liaison Executive (EPISTLE).

American National Standards Institute. 1975. *ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report*. FDT (Bulletin of ACM SIGMOD) 7:2.

Data Warehouse tool DWE - The most easy to use Data Warehouse tool. BI.dwexplorer.com/datawarehousetool

Riversand MDM Multi-entity Master Data Mgmt. Global, enterprise-class solutions www.riversand.com

"Distributed Application Architecture". Sun Microsystem. http://java.sun.com/developer/Books/jdbc/ch07. p

Module 1 Fundamentals of Database Systems

Unit 3 Data Modelling Using the Entity-Relationship Model

1.0 Introduction

2.0 Objective

3.0 Using High-Level Conceptual Data Models for Database Design

3.1 A Sample Database Application

3.2 Entity Types, Entity Sets, Attributes, and keys

3.3 Relationships, Relationship Types, Roles, and Structural Constraints

3.4 Weak Entity Types

3.5 Refining the ER Design for the Company Database

3.6 ER Diagrams, Naming Conventions, and Design Issues

4.0 Conclusion

5.0 Summary

6.0 Tutor Marked Assignment

7.0 Further Reading and Other Resources

1.0 Introduction

Traditionally, the design and testing of application programs have been considered to be more in the realm of the software engineering domain than in the database domain. As database design methodologies include more of the concepts for specifying operations on database objects, and as software engineering methodologies specify in more detail the structure of the databases that software programs will use and access, it is clear that these activities are strongly related. Conceptual modelling is a very important phase in designing a successful database application.

Generally, the term **database application** refers to a particular database and the associated programs that implement the database queries and updates. In this unit, the traditional approach of concentrating on the database structures and constraints during database design is strictly followed. The modelling concepts of the **Entity-Relationship**(**ER**) **model** is herewith presented, which is a popular high-level conceptual data model.

This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts. The basic data-structuring concepts and constraints of the ER model as well as their use in the design of conceptual schemas for database applications are described in this unit. Diagrammatic notation associated with the ER model, known as **ER diagrams** is also presented.

2.0 Objective

At the end of this unit, you should be able to:

- ❖ appreciate the features of Entity Types, Entity Sets, Attributes, and keys of database management system (DBMS)
- ❖ Use High-Level Conceptual Data Models for Database Design
- ❖ State the major concepts Relationships, Relationship Types, Roles, and Structural Constraints
- ❖ Draw the major components of ER Diagrams, Naming Conventions, and Design Issues in data management system.

3.0 Using High-Level Conceptual Data Models for Database Design

Figure below shows a simplified description of the database design process. The first step shown is **requirements collection and analysis.** During this step, the database designers interview prospective database users to understand and document their **data requirements.** The result of this step is a concisely written set of users' requirements. These requirem-ents should be specified in as detailed and complete a form as possible.

In parallel with specifying the data requirements, it is useful to specify the known **functional requirements** of the application. These consist of the user-defined **operations** (or **transactions**) that will be applied to the database, including both retrievals and updates. In software design, it is common to use data flow diagrams, sequence diagrams, scenarios, and other techniques for specifying functional requirements. We will not discuss any of these techniques here because they are usually described in detail in software engineering texts.

Once all the requirements have been collected and analysed, the next step is to create a **conceptual schema** for the database, using a high-level conceptual data model. This step is called **conceptual design.** The concept-tual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model. Because these concepts do not include implementation details, they are usually easier to understand and can be used to communicate with non-technical users.
The high-level conceptual schema can also be used as a reference to ensure that all users' data requirements are met and that the requirements do not conflict. This approach enables the database designers to concentrate on specifying the properties of the data, without being concerned with storage details. Consequently, it is easier for them to come up with a good conceptual database design.
During or after the conceptual schema design, the basic data model operations can be used to specify the high-level user operations identified during functional analysis. This also serves to confirm that the conceptual schema meets all the identified functional requirements. Modifications to

the conceptual schema can be introduced if some functional requirements cannot be specified using the initial schema.

The next step in database design is the actual implementation of the database, using a commercial DBMS. Most current commercial DBMSs use an implementation data model, such as the relational or the object-relational database model—so the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called **logical design** or **data model mapping,** and its result is a database schema in the implementation data model of the DBMS.

The last step is the **physical design** phase, during which the internal storage structures, indexes, access paths, and file organizations for the database files are specified. In parallel with these activities, application programs are designed and implemented as database transactions corresponding to the high-level transaction specifications.

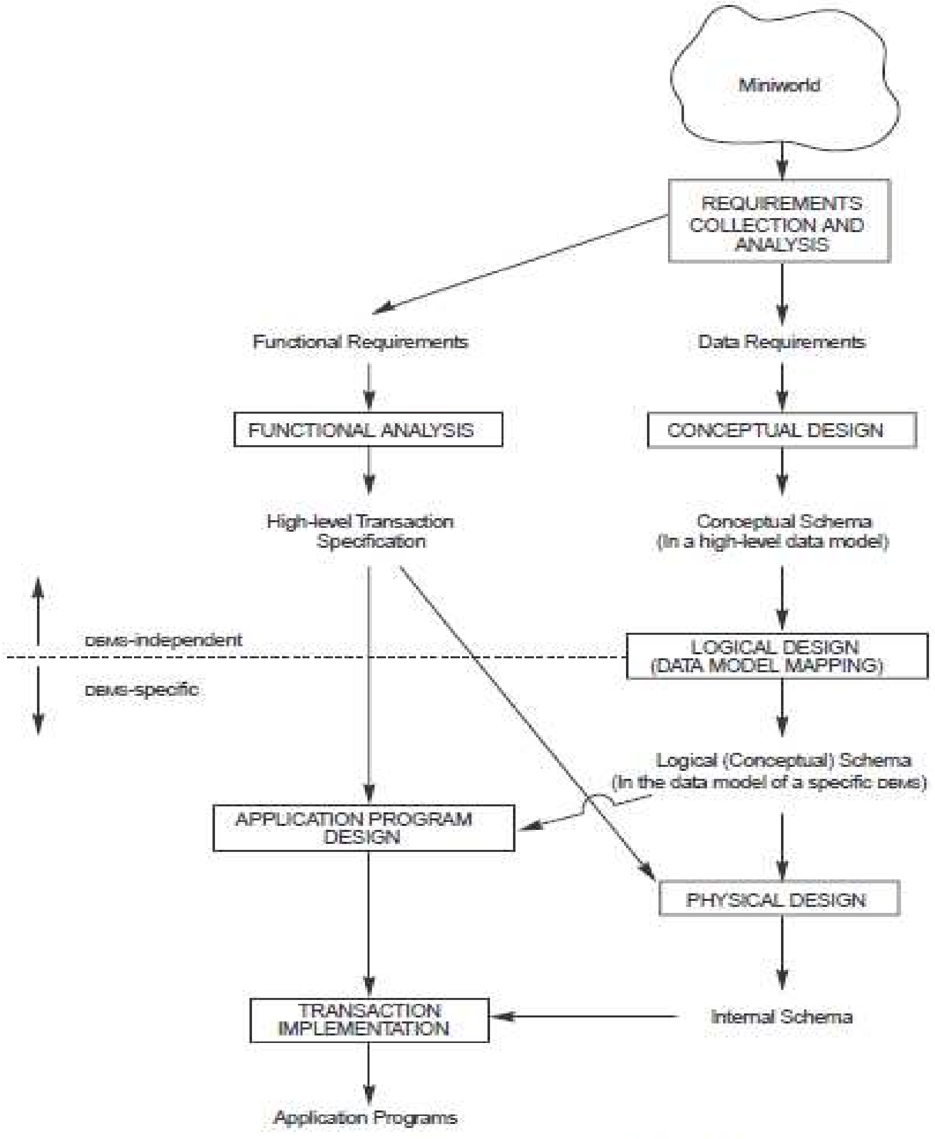

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 16: A Simplified Diagram of Phases of database Design

3.1 A Sample Database Application

In this section we describe an example database application, called COMPANY that serves to illustrate the basic ER model concepts and their use in schema design. We list the data requirements for the database here, and then create its conceptual schema step by step as the modelling concepts of the ER model are introduced. The COMPANY database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provided the following description of the “miniworld”—the part of the company to be represented in the database:

1. The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
2. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
3. We store each employee’s name, social security number, address, salary, sex, and birth date. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
4. We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent’s first name, sex, birth date, and relationship to the employee.

Figure 3.2 shows how the schema for this database application can be displayed by means of the graphical notation known as **ER diagrams.** We describe the step-by-step process of deriving this schema from the stated requirements—and explain the ER diagrammatic notation—as we introduce the ER model concepts in the following section.

3.2 Entity Types, Entity Sets, Attributes, and keys

The ER model describes data as entities, relationships, and attributes:

Entities and Their Attributes: The basic object that the ER model represents is an **entity,** which is a "thing" in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for example, a company, a job, or a university course). Each entity has **attributes**—the particular properties that describe it. For example, an employee entity may be described by the employee's name, age, address, salary, and job. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.

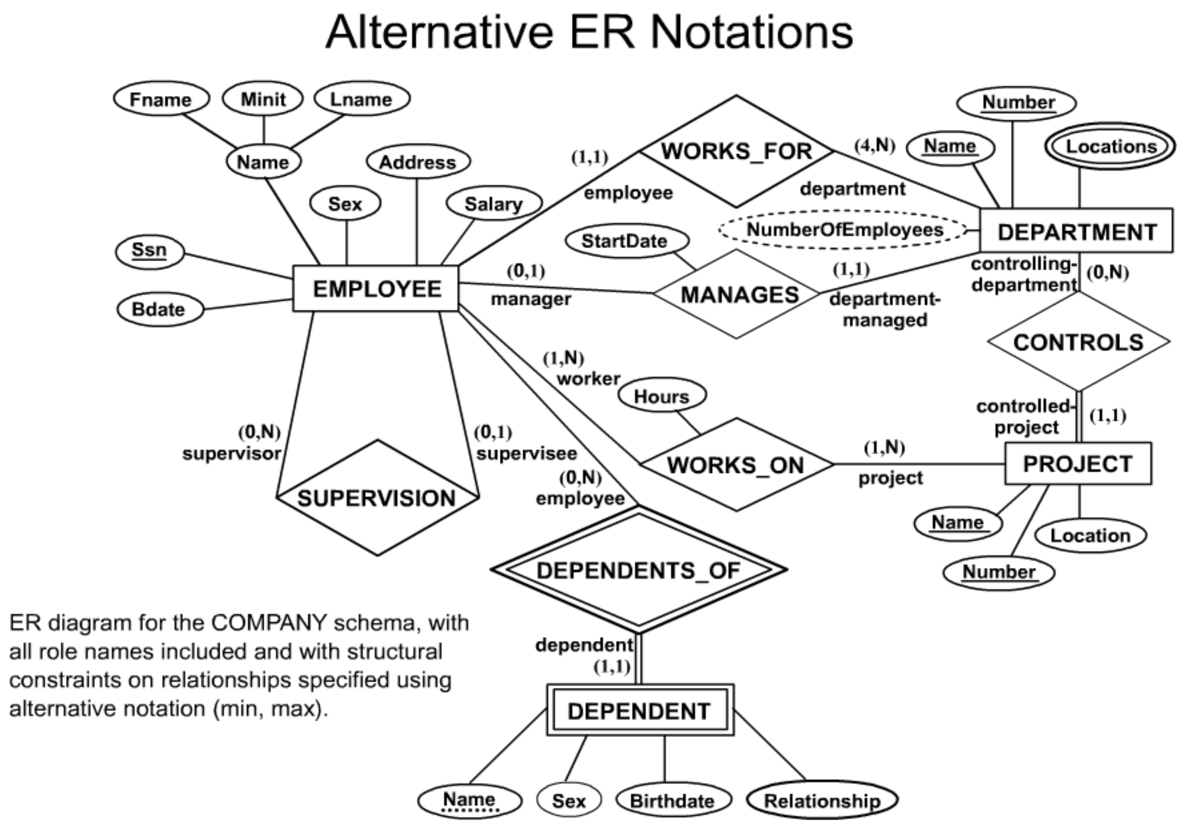


Figure 17: Alternative ER Notation

Figure 3.3 shows two entities and the values of their attributes. The employee entity *e*1 has four attributes: Name, Address, Age, and HomePhone; their values are “John Smith,” “2311 Kirby, Houston, Texas 77001,” “55,” and “713-749-2630,” respectively. The company entity *c*1 has three attributes: Name, Headquarters, and President; their values are “Sunco Oil,” “Houston,” and “John Smith,” respectively.

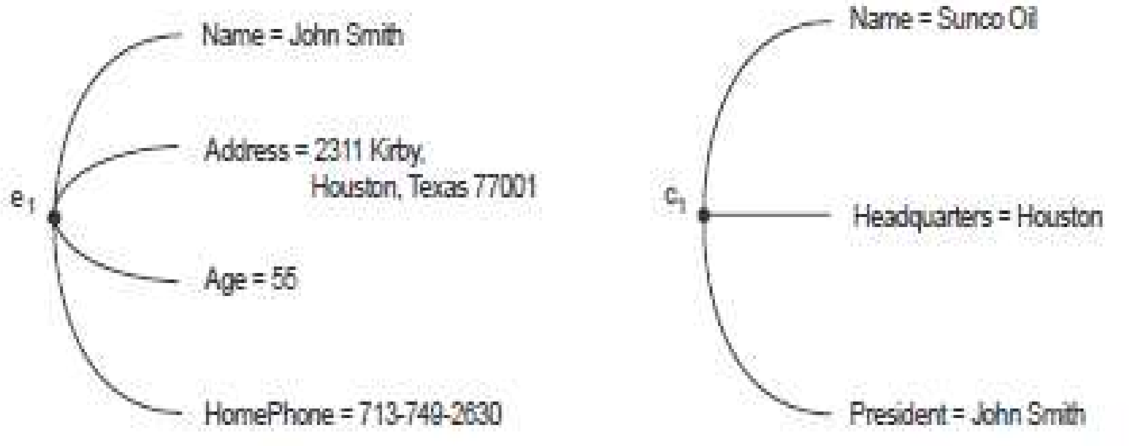


Figure 18: Two entities, employee e_1 and company c_1, and their attributes

Several types of attributes occur in the ER model:
In ER model we have the following attributes: simple versus composite, single-valued versus multi-valued, and stored versus derived.

Composite versus Simple (Atomic) Attributes: Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. For example, the Address attribute of the employee entity shown in Figure 3.3 can be subdivided into StreetAddress, City, State, and Zip,3 with the values “2311 Kirby,”“Houston,” “Texas,” and “77001.” Attributes that are not divisible are called **simple** or **atomic attributes.** Composite attributes can form a hierarchy; for example, StreetAddress can be further subdivided into three simple attributes: Number, Street, and ApartmentNumber, as shown in Figure 3.4. The value of a composite attribute is the concatenation of the values of its constituent simple attributes.

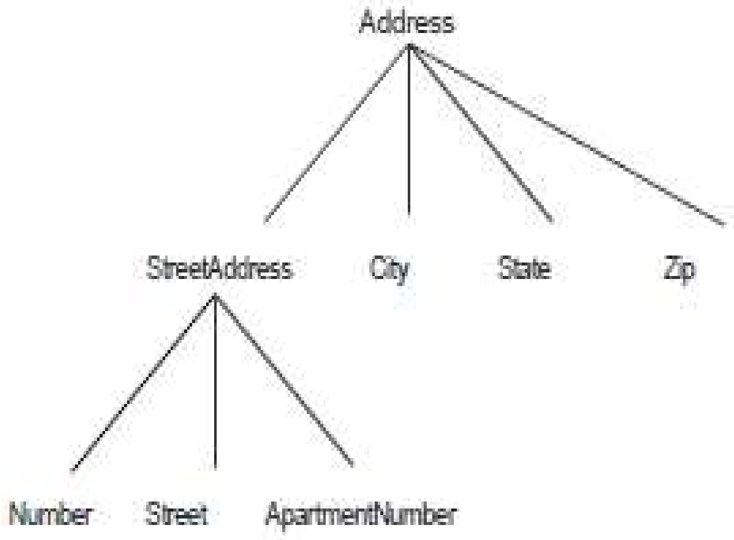


Figure 19: A hierarchy of composite attributes

Single-Valued versus Multivalued Attributes: Most attributes have a single value for a particular entity; such attributes are called **single-valued.** For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity—for example, a Colors attribute for a car, or a CollegeDegrees attribute for a person. Cars with one color have a single value, whereas two-tone cars have two values for Colors. Similarly, one person may not have a college degree, another person may have one, and a third person may have two or more degrees; therefore,

different persons can have different *numbers of values* for the CollegeDegrees attribute. Such attributes are called **multivalued.**
Stored versus Derived Attributes. In some cases, two (or more) attribute values are related—for example, the Age and BirthDate attributes of a person. For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's BirthDate. The Age attribute is hence called a **derived attribute** and is said to be **derivable from** the BirthDate attribute, which is called a **stored attribute.**
In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare. For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}.

Entity Types, Entity Sets, and Keys

Entity Types and Entity Sets: A database usually contains groups of entities that are similar. For example, a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has *its own value(s)* for each attribute. An **entity type** defines a *collection* (or *set*) of entities that have the same attributes. Each entity type in the database is described by its name and attributes. **Figure** shows two entity types, named EMPLOYEE and COMPANY, and a list of attributes for each.
A few individual entities of each type are also illustrated, along with the values of their attributes. The collection of all entities of a particular entity type in the database at any point in time is called an **entity set;** the entity set

is usually referred to using the same name as the entity type. For example, EMPLOYEE refers to both a *type of entity* as well as the current *set of all employee entities* in the database.
An entity type describes the **schema** or **intension** for a *set of entities* that share the same structure. The collection of entities of a particular entity type is grouped into an entity set, which is also called the **extension** of the entity type.

Key Attributes of an Entity Type: An important constraint on the entities of an entity type is the **key** or **uniqueness constraint** on attributes. An entity type usually has an attribute whose values are distinct for each individual entity in the entity set. Such an attribute is called a **key attribute,** and its values can be used to identify each entity.

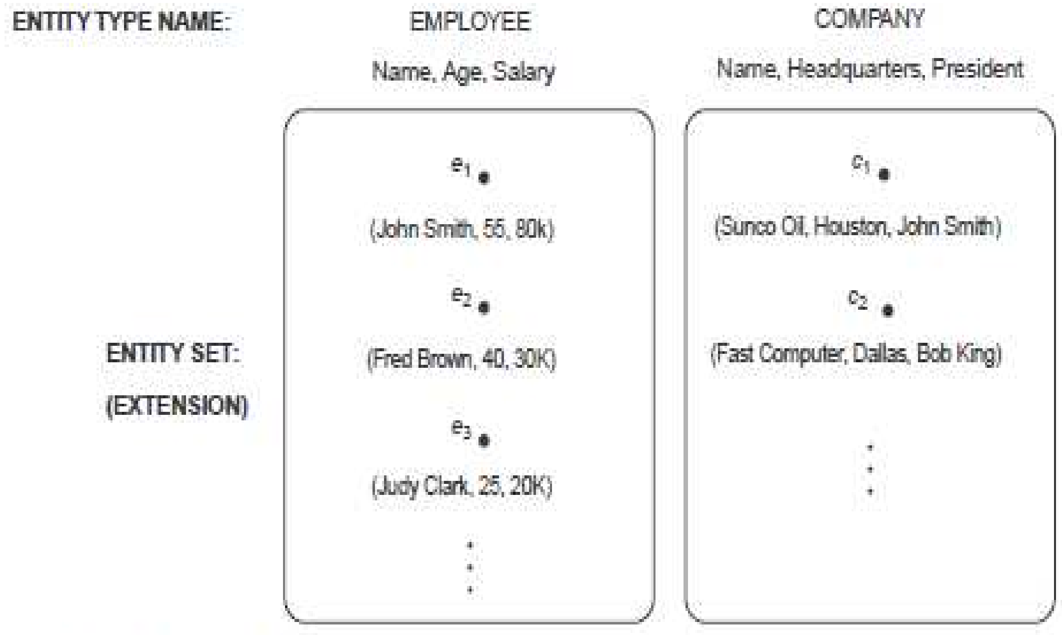
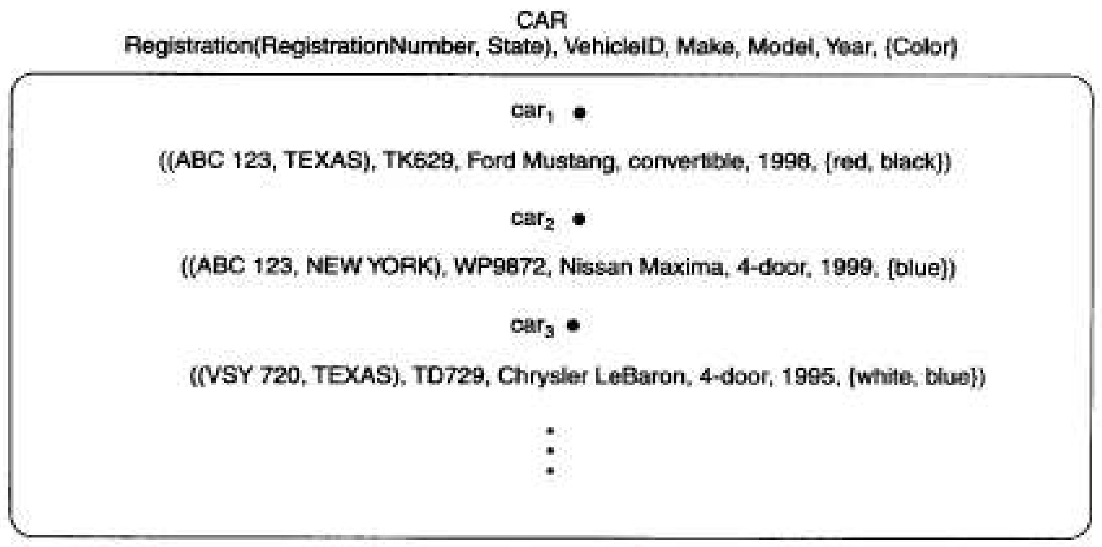


Figure 20: Two entity types, EMPLOYEE and COMPANY, and some member entities of each


CAR
Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, {Color}

car_1 •

((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1998, {red, black})

car_2 •

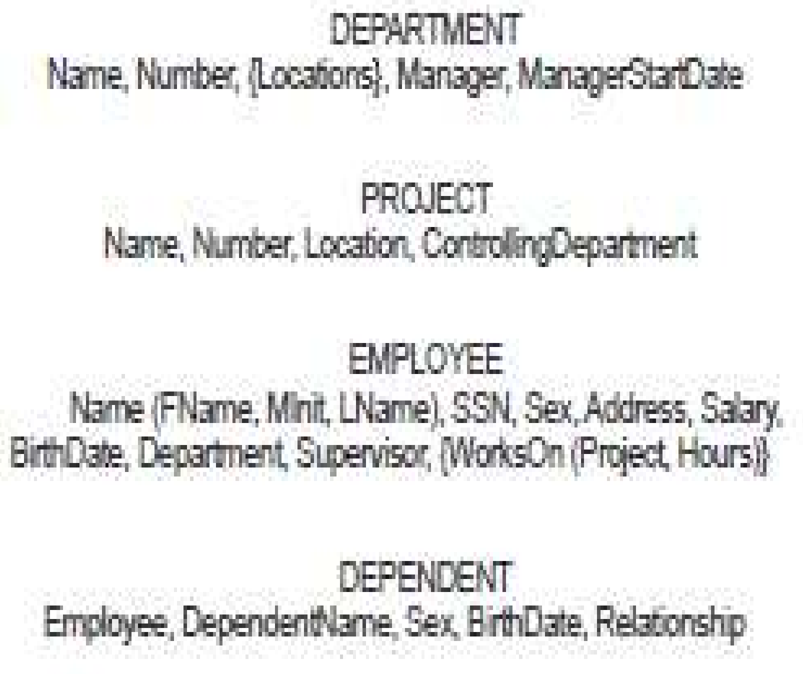
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 1999, {blue})

car_3 •

((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 1995, {white, blue})

⋮

Figure 21: The CAR entity type with two key attributes, Registration and VehicleID


DEPARTMENT
Name, Number, {Locations}, Manager, ManagerStartDate

PROJECT
Name, Number, Location, ControllingDepartment

EMPLOYEE
Name (FName, MInit, LName), SSN, Sex, Address, Salary, BirthDate, Department, Supervisor, {WorksOn (Project, Hours)}

DEPENDENT
Employee, DependentName, Sex, BirthDate, Relationship

Figure 22: Preliminary design of entity types for the COMPANY database

.

3.3 Relationships, Relationship Types, Roles, and Structural Constraints

In **Figure 17** above, there are several *implicit relationships* among the various entity types. In fact, whenever an attribute of one entity type refers to another entity type, some relationship exists. For example, the attribute Manager of DEPARTMENT refers to an employee who manages the department; the attribute Controlling Department of PROJECT refers to the department that controls the project; the attribute Supervisor of EMPLOYEE refers to another employee (the one who supervises this employee); the attribute Department of EMPLOYEE refers to the department for which the employee works; and so on. In the ER model, these references should not be represented as attributes but as **relationships,** which are discussed in this section.

A relationship relates two or more distinct entities with a specific meaning.
For example: *EMPLOYEE John Smith works on the ProductX PROJECT or*
EMPLOYEE Franklin Wong manages the Research DEPARTMENT.

A relationship type *R* among *n* entity types *E*1, *E*2, . . . , *En* defines a set of associations or a **relationship set**—among entities from these entity types. Relationships of the same type are grouped or typed into a relationship type.

For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs *participate. The degree of a relationship type is the number of participating entity types. Both MANAGES and WORKS_ON are binary relationships.*

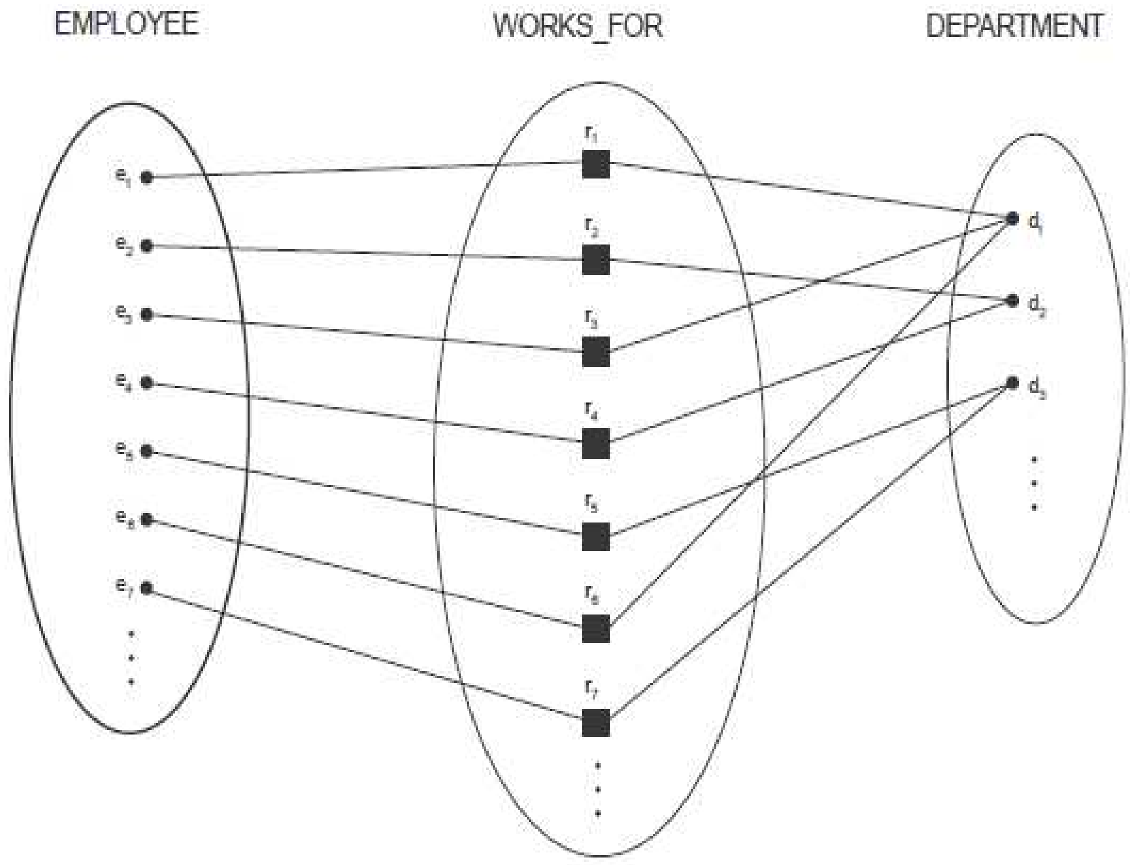


Figure 23: Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT

Role Names and Recursive Relationships: Each entity type that participates in a relationship type plays a particular **role** in the relationship. The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means. For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of *employee* or *worker* and DEPARTMENT plays the role of *department* or *employer*. Role names are not technically necessary in relationship types where all the participating entity types are distinct, since each participating entity type name can be used as the role name.

However, in some cases the *same* entity type participates more than once in a relationship type in *different roles*. In such cases the role name becomes essential for distinguishing the meaning of each participation. Such relationship types are called **recursive relationships.**

CONSTRAINTS ON RELATIONSHIP TYPES

Relationship Types (also known as ratio constraints) usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. These constraints are determined from the miniworld situation that the relationships represent. For example, in **Figure 22**, if the company has a rule that each employee must work for exactly one department, then we would like to describe this constraint in the schema. We can distinguish two main types of relationship constraints: *cardinality ratio* and *participation:*

Cardinality Ratios for Binary Relationships. The **cardinality ratio** for a binary relationship specifies the *maximum* number of relationship instances that an entity can participate in. For example, in the WORKS_FOR binary relationship type, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department. The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N. Cardinality ratios for binary relationships are represented on ER diagrams by displaying 1, M, and N on the diamonds as shown in **Figure 17** above.

Maximum Cardinality can be in the form of:

- One-to-one (1:1)
- One-to-many (1:N) or Many-to-one (N:1)
- Many-to-many

The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the *minimum* number of relationship instances that each entity can participate in, and is sometimes called the **minimum cardinality constraint.**

Minimum Cardinality (also called participation constraint or existence dependency constraints) can also be in the form of:

- zero (optional participation, not existence-dependent)
- one or more (mandatory, existence-dependent)

3.4 Weak Entity Types

Entity types that do not have key attributes of their own are called **weak entity types.** In contrast, **regular entity types** that do have a key attribute are called **strong entity types.** Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values. We call this other entity type the **identifying** or **owner entity type,** and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type. A weak entity type always has a *total participation* *constraint* (existence dependency) with respect to its identifying relationship, because a weak entity cannot be identified without an owner entity. However, not every existence dependency results in a weak entity type. For example, a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (LicenseNumber) and hence is not a weak entity.

In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines (**see Figure 17**). The partial key attribute is underlined with a dashed or dotted line.

Weak entity types can sometimes be represented as complex (composite, multi-valued) attributes.

3.5 Refining the ER Design for the Company Database

We can now refine the database design of **Figure 22,** by changing the attributes that represent relationships into relationship types. The cardinality ratio and participation constraint of each relationship type are determined from the requirements listed earlier. If some cardinality ratio or dependency cannot be determined from the requirements, the users must be questioned further to determine these structural constraints. In our example, we specify the following relationship types:

1. MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT

participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation.14 The attribute StartDate is assigned to this relationship type.
2. WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
3. CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.
4. SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participat-ions are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.
5. WORKS_ON, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.

3.6 ER Diagram and Naming Conventions

Figure 3.2 above displays the COMPANY **ER database schema** as an **ER diagram.** We now review the full ER diagram notation.

General Database Naming Conventions

The following conventions apply to all of the elements of a database:

- All names used throughout the database should be lowercase only. This will eliminate errors related to case-sensitivity.

- Separate name parts by underlines, never by spaces. This way, you improve the readability of each name (e.g. product_name instead of productname). You will not have to use parentheses or quotes to enclose names using spaces as well. The use of spaces in a database name is allowed only on some systems, while the underline is an alphanumeric character, allowed on any platform. Thus, your database will become platform independent.
- Do not use numbers in the names (e.g. product_attribute1). This is proof of poor design, indicating a badly divided table structure. If you need a many-to-many relation, the best way to achieve it is by using a separate linking table. See how here. Moreover, using numbers to differentiate between two columns that store similar information might be an indication that you need an extra table, storing that information. For instance, having a location1 column in a manufacturers table and a location2 column in a distributors table could be solved by creating a separate table that stores all locations, and that is referenced by both the manufacturers and distributors tables via foreign keys.
- Do not use the dot (.) as a separator in names. This way you will avoid problems when trying to perform queries, as the dot is used to identify a field in a specific column. In SQL language, manufacturer_man.address_man means the address column from the table that stores information about manufacturers.
- Do not use any of the reserved words as names of database elements. Each database language uses some words as names for internal

functions, or as part of the SQL syntax. For instance, using order as the name of a table that stores product orders from an online shop is bad practice, because order is also used in SQL language to sort records (ascending or descending).
For a complete list of the reserved words that you should not use when naming the database elements, consult your specific database software manual. See the list of reserved words for MySQL here.

- When naming the elements, do not use long or awkward names. Keep them as simple as you can, while maintaining a clear meaning. It's also a good idea to use names which are close to the natural language: description_prd is certainly a better name for a column that stores product descriptions than dscr_pr or some generic name as field.

Database names

Each database must have a name of its own, which should also follow some conventions:

- Use the project name as the name of the database.
- Prefix the database name with the owner name, separated by an underscore. The owner might be a person (e.g. the project manager) or the application for which the database is created. For example, acme catalog can be a good name for the database storing
  the product catalog of the ACME company.

Table names

Tables are some of the most common elements used in an application, as they store the columns, and as such are mentioned in each query. Therefore, the following conventions should apply to tables:

- Table names contain the name of the entity that is being defined, followed by a three letter acronym of that name (e.g. category_ctg). Optionally, you can use the same prefix for all tables in the same database. For example, acme_product_prd, acme_manufacturer_prd, acme_category_prd can be tables from the acme_catalog database.
- Prefix tables that define the same larger entity with a 2 or 3-letter acronym that identifies it. For example, e.g. hr_applicant_app, hr_job_job and hr_resume_rsm are
  all tables that belong to the Human Resources Department of a large corporation database.
- Do not use generic prefixes, such as tbl_, or db_, as they are redundant and useless.
- Use short, unambiguous names for each table, restricted to one word, if possible. This way tables can be distinguished easily.
- Use singular for table names. This way, you avoid errors due to the pluralization of English nouns in the process of database development. For instance, activity becomes activities, box becomes boxes, person becomes people or persons, while data remains data.
- Use clear names. Do not overdo it using abbreviations and acronyms. While using a shorter name might help the developers, it makes the meaning less clear to other members of the team. Using clear names makes the design self-explanatory.

- Prefix lookup tables with the name of the table they relate to. This helps group related tables together (e.g. product_type, product_status), and also helps prevent name conflicts between generic lookup tables for different entities. You can have more than one generic lookup table for an existing master table, but which address different properties of the elements in the table.

Column names

Columns are attributes of an entity, describing its properties. Therefore, the name they carry should be natural, and as meaningful as possible. The following conventions are recommended:

- All keys are used for indexing and identifying records. Therefore, it's a good practice to put the id particle in their name. This way, you'll know the field is used as a key.
- The primary key is used to uniquely identify each record. That is why its name should be made up of the id particle, followed by the table name acronym. For instance, for the table product_prd, the primary key is id_prd.
- The foreign key name should be composed by the id particle, followed by the acronym of the referred table, and then by the acronym of the table it belongs to. For example, the idctg_prd foreign key belongs to the products table (product_prd), but it refers to the categories table (category_ctg). This way, the table being referenced is obvious from the key name.

- Each column name should be followed by the 3-letter table acronym. This way, each column has a unique name across the database. Without the table acronym, you would end up having two columns called "name", one storing the product name and the other the manufacturer name. Instead, name_prd and name_man can easily be distinguished.
- Date columns should use the “date_” prefix, and the boolean type columns should use the “is_” prefix. For instance, date birth stores the birth date of a person, while is confirmed could indicate the order status for a product in a shop, using true/false values (or 0/1).

❖ 4.0 Conclusion

With the introduction of High-Level Conceptual Data Models for Database Design. We have explained all the features of Entity Types, Entity Sets, Attributes, and keys of database management system (DBMS) in relationships to the major concepts Relationships and Structural Constraints. Students must have acquired the basic concepts of the conceptual importance of data models in today’s designing systems. And with this, they should be able to function very well in such environment.

5.0 Summary

In this unit we have learnt that:

- ❖ The features of Entity Types, Entity Sets, Attributes, and keys of database management system (DBMS) as a set of concepts used to describe the structures of a database management and the operations for manipulating these structures, as well as certain constraints that the database should obey.
- ❖ The use of High-Level Conceptual Data Models for Database Design
- ❖ The major concepts of Relationships, Relationship Types, Roles, and Structural Constraints in database management systems (DBMS)
- ❖ How to draw the major components of ER Diagrams, Naming Conventions and Design Issues in data management system

6.0 Tutor Marked Assignment

- ❖ Explain the use of High-Level Conceptual Data Models for Database Design is all about.
- ❖ Differentiate between the major concepts of Relationship Types and Structural Constraints in database management systems (DBMS)
- ❖ Draw the major components of ER Diagrams, Naming Conventions and Design Issues in database management system

7.0 Further Reading and Other Resources

Stephen M. Richard (1999). Geologic Concept Modelling. U.S. Geological Survey Open-File Report 99-386.

Joachim Rossberg and Rickard Redler (2005). Pro Scalable .NET 2.0 Application Designs.. Page 27

Terry Halpin (2001). "Object-Role Modelling: an overview"

The ORM Foundation home page

Terry Halpin (2001), Object-Role Modelling: an overview

Terry Halpin (2005), ORM2 On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops, eds R. Meersman, Z. Tari, P. Herrero et al., Cyprus. Springer LNCS 3762, pp 676–87.

Halpin, Terry (1989), Conceptual Schema and Relational Database Design, Sydney: Prentice Hall, ISBN 978-0131672635

Rossi, Matti; Siau, Keng (April 2001), Information Modelling in the New Millennium, IGI Global, ISBN 978-1878289773

The Entity Relationship Model: Toward a Unified View of Data" for entity-relationship modelling.

A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in

Douque and Nijssen (eds.), Data Base Description, North-Holland, 1975, ISBN 0-7204-2833-5.

Paul Beynon-Davies (2004). Database Systems. Houndmills, Basingstoke, UK: Palgrave

"Gliffy March 2007 NewsLetter", March 1, 2007, accessed January 13, 2011

Richard Barker (1990). CASE Method: Tasks and Deliverables. Wokingham, England: Addison-Wesley.

Paul Beynon-Davies (2004). Database Systems. Houndmills, Basingstoke, UK: Palgrave

Module 2 The Statistical database system

Unit 1 Statistical Database Concepts

1.0 Introduction

2.0 Objective

3.0 Basic Definition

3.1 Concept of Public Policies on Statistical Data

3.2 The Design of a Statistical Database

3.3 Features and Requirements of a Statistical Database

3.4 Statistical Database Modelling

3.5 Some Statistical Data Models

4.0 Conclusion

5.0 Summary

6.0 Tutor Marked Assignment

7.0 Further Reading and Other Resources

1.0 Introduction

A statistical database contains confidential information about individuals or events. These databases are mainly used to generate statistical information about the stored information. But Statistical databases have some non-standard characteristics which cannot be well supported by commercially available database management systems. This concerns the underlying data structures, the various abstraction levels of the data, and the type of operations on the data and the kind of processing requirements. The current research is aiming at an appropriate conceptual modelling, an efficient representation of the data and a powerful and user-friendly query processing on each data level.

2.0 Objectives

At the end of this unit, you should be able to:

- ❖ Appreciate the Features of Statistical database system (DBMS)
- ❖ Define Statistical database system.
- ❖ Use High-Level Concept of Statistics data in policies formulation.
- ❖ State some major concepts of Statistical database Models.
- ❖ Design the major components of Statistical database and Modelling.

3.0 Basic Definition

Databases that are mainly used for statistical analysis are called **statistical databases (SDB).** And a statistical database management system

(SDBMS) may be defined as a database management system that provides capabilities to model, store, and manipulate data in a manner suitable for the needs of SDB users. It is also a database management system that provides capabilities to apply statistical data analysis techniques that range from simple summary statistics to advanced procedures.
In addition**,** a statistical database (SDB) is one that provides data of a statistical nature, such as counts and averages. The term *statistical database* is used in two contexts:

- **Pure statistical database:** This type of database only stores statistical data. An example is a census database. Typically, access control for a pure SDB is straightforward: Certain users are authorized to access the entire database.
- **Ordinary database with statistical access:** This type of database contains individual entries; this is the type of database discussed so far in this chapter. The database supports a population of non-statistical users who are allowed access to selected portions of the database using DAC, RBAC, or MAC. In addition, the database supports a set of statistical users who are only permitted statistical queries. For these latter users, aggregate statistics based on the underlying raw data are generated in response to a user query, or may be pre-calculated and stored as part of the database.

3.1 Concept of Public Policies on Statistical Data

A statistical database contains confidential information about individuals

or events. These databases are mainly used to generate statistical information about the stored information. Such databases accept only statistical queries , which involve statistical functions such as SUM, AVG, COUNT, MIN, MAX , and so on. However, users are not allowed to retrieve information about a particular individual.
Consider a relation BankEmp with the attributes ECode, EName, Sex, State, Salary, Branch , and Designation . Using statistical queries, one can retrieve the number of clerks, maximum salary, average salary of clerks, and so on. However, users are not allowed to retrieve the salary of a particular employee.

Data Confidentiality and Disclosure

Confidentiality should mean that, the dissemination of data in a manner that would allow public identification of the respondent or would in any way be harmful to him is prohibited and that the data are immune from legal process. Confidentiality differs from privacy because it applies to business as well as individuals. Privacy is an individual right whereas confidentiality often applies to data on organizations and firms.
Statistical disclosure occurs when released statistical data (either tabular or individual records) reveal confidential information about an individual respondent.

Disclosure relates to inappropriate attribution of information to a data subject, whether an individual or an organization. Disclosure occurs when a data subject is identified from a released file (**identity disclosure**), sensitive information about a data subject is revealed through the released file (**attribute disclosure**), or the released data make it possible to determine the value of some characteristic of an individual more accurately than otherwise would have been possible (**inferential disclosure**).

Note that each type of disclosure can occur in connection with the release of either tables or microdata. The definitions and implications of these three kinds of disclosure are discussed below: **Identity disclosure** occurs if a third party can identify a subject or respondent from the released data. Revealing that an individual is a respondent or subject of a data collection may or may not violate confidentiality requirements.

For tabulations, revealing identity is generally not disclosure, unless the identification leads to divulging confidential information (attribute disclosure) about those who are identified.

For microdata, identification is generally regarded as disclosure, because microdata records are usually so detailed that identification will automatically reveal additional attribute information that was not used in identifying the record. Hence disclosure limitation methods applied to microdata files limit or modify information that might be used to identify specific respondents or data subjects.

Attribute disclosure occurs when confidential information about a data subject is revealed and can be attributed to the subject. Attribute disclosure occurs when confidential information about a person or firm’s business operations is revealed or may be closely estimated. Thus, attribute

disclosure comprises identification of the subject and divulging confidential information pertaining to the subject.
Attribute disclosure is the primary concern of most statistical agencies in deciding whether to release tabular data. Disclosure limitation methods applied to tables assure that respondent data are published only as part of an aggregate with a sufficient number of other respondents to disguise the attributes of a single respondent.

Inferential disclosure, occurs when individual information can be inferred with high confidence from statistical properties of the released data. For example, the data may show a high correlation between income and purchase price of home. As purchase price of home is typically public information, a third party might use this information to infer the income of a data subject. There are two main reasons that some statistical agencies are not concerned with inferential disclosure in tabular or micro data. First a major purpose of statistical data is to enable users to infer and understand relationships between variables.

Tables, Microdata, and On-Line Query Systems

The choice of statistical disclosure limitation methods depends on the nature of the data products whose confidentiality must be protected. Most statistical data are released in the form of tables, microdata files, or through on-line query systems. Tables can be further divided into two categories: tables of frequency (count) data and tables of magnitude data. For either

category, data can be presented in the form of numbers, proportions or percentages.
A microdata file consists of individual records, each containing values of variables for a single person, business establishment or other unit. Some microdata files include direct identifiers, such as name, address or Social Security number. Removing any of these identifiers is an obvious first step in preparing for the release of a file for which the confidentiality of individual information must be protected.
Historically, disclosure limitation methods for tables were applied directly to the tables. Methods include redesign of tables, suppression, controlled and random rounding. More recent methods have focused on protecting the **microdata** underlying the tables using some of the microdata protection techniques. In this way all tables produced from the protected microdata are also protected. This may be done whether there is an intention to release the microdata or not. It is a particularly useful way to protect tables produced from on-line query systems.

Restricted Data and Restricted Access

The confidentiality of individual information can be protected by restricting the amount of information provided or by adjusting the data in released tables and microdata files (**restricted data**) or by imposing conditions on access to the data products (**restricted access**), or by some combination of these.

Tables of Magnitude Data Versus Tables of Frequency Data

The selection of a statistical disclosure limitation technique for data presented in tables (**tabular data**) depends on whether the data represent frequencies or magnitudes. Tables of **frequency count data** present the number of units of analysis in a cell. Equivalently the data may be presented as a percent by dividing the count by the total number presented in the table (or the total in a row or column) and multiplying by 100. Tables of **magnitude data** present the aggregate of a "quantity of interest" that applies to units of analysis in the cell. Equivalently the data may be presented as an average by dividing the aggregate by the number of units in the cell. To distinguish formally between **frequency count data** and **magnitude data**, the "quantity of interest" must measure something other than membership in the cell. Thus, tables of the number of establishments within the manufacturing sector by SIC group and by county-within-state are frequency count tables, whereas tables presenting total value of shipments for the same cells are tables of magnitude data.

3. 2 The Design of a Statistical Database (Micro-, Macro and Metadata Modelling)

A database is called a statistical database (SDB) if it contains data of three kinds:

Microdata as primary or basis data on individuals, objects or events representing sampled, census or collected data.

Macrodata as grouped or aggregated data (summarized data) which are cross-classified by a set of categorical attributes(variables). The summary attribute represents counts(frequencies), means, indices or other statistics characterizing a set(population) of individuals, objects or events.

Metadata describing the micro- and macrodata on the semantic, structural, statistical and physical level in such a way that they can be stored, transfor-med retrieved and transmitted in a reasonable way. It covers the whole data life cycle, i.e. the data collecting from the data source, the data storing, the data processing and retrieval, and the data disseminating within the electronic data interchange (EDI). As a matter of fact, the metadata itself must be easily accessible similiar to the micro- and macrodata. We close the introduction with some examples of typical retrieval operations (queries) on the data of the above kind. We make use of a pseudo-code notation.In some cases the system response (i^) is given.

Microdata

```
list name, age, sex
from labourcensusemployees
where industry = 'whole industry' and year = 1980
```

Macrodata

```
list number (employees), average (employees.income)
from labourcensus
where industry - 'whole industry' and year - 1980
cross-classified by age^group and sex
```

Metadata

household *All the people belong to a household who live there together and have a joint budget Each person who has an own*
budget forms her own household,
summary-attribute (employees)
income categoryattribute (employees)
domain (industry)

3.3 Features and Requirements of a Statistical Database

The data structures offered by conventional database management systems are rather inconvenient for storing and retrieving statistical data. Moreover, the functionality of the systems is inappropriate and inadequate for interpreting, validating and analyzing such kind of data. The main reasons are the followings:

- Set- (tupel- or row-)oriented as well as ordered structures must berepresented.
- Vectors, matrices, nested arrays besides non-fixed formatted data types like documents, images, and plots must be stored.
- Almost static (historical) micro- and macrodata with very slow update rates exist.
- GBytes of data are to be stored some of which are sparse.
- Very complex semantic integrity constraints are defined on the micro- and macrodata, espescially in order to map their links.
- Micro- and macrodata without the corresponding metadata are useless.
- The locality (^clusterings) of data according to attributes (columns) or records (rows) is context-dependent.
- The querying and the processing of statistical data can cause »long transactions«.
- The creation of macrodata by grouping, aggregation etc. implies the bookkeeping of metadata.
- The frontier (interface) between retrieval and statistical analysis is not crisp.
- The retrieval of even »anonymous« data must care about privacy rules.

The above special features of a statistical database (STDB) give raise to the following requirements for the design of a statistical database:

- The conceptual data model must include the micro-, macro- and metadata level.
- The data-structures for micro-, macro- and metadata must be efficient.

- The set of operators should be complete which is defined on the micro-, macro- and meta data level.
- The user-interfaces for the different user-groups must be user-friendly.
- Privacy handling mechanism must be incorporated.

3.4 Statistical Data Modelling

In this section, we propose to use the general location model to model database and use model learnt to generate synthetic database. We will examine in detail how to extract statistics and rules to estimate parameters of the general location model and how to resolve the potential disclosure of confidential information in data generation using model learnt.

The General Location Model

Let $A_1;A_2; \_ \_ \_ ;A_p$ denote a set of categorical attributes and $Z_1;Z_2; \_ \_ \_ ;Z_q$ a set of numerical ones in a table with n entries. Suppose A_j takes possible domain values 1, 2, $\_ \_ \_$; d_j , the categorical data **W** can be summarized by a contingency table with total number of cells equal to $D = \pi^p{}_{j=1} {}^d{}_j$. Let $\mathbf{x} = \{x_d : d = 1; 2; \_ \_ \_ ;D\}$ denote the number of entries in each cell. Clearly $\sum^D{}_{d=1} {}^{xd} = n$.

Database Modelling through the General Location Model

Our approach is to derive an approximate statistical model from the characteristics (e.g., constraints, statistics, rules, and data summary) of the real databases and generate a synthetic data set using model learned. A major advantage of our system over [1] is that, in addition to constraints, we

extract more complex characteristics (e.g., statistics and rules) from data catalogue and data and use them to build statistical model. As we discussed in introduction, even if one synthetic database satisfies all constraints, it does not mean it can fulfil users' testing requirement as it may have different data distribution than production database.
Our intuition is that, for database applications, if two databases are approximately the same from a statistical viewpoint, then the performance of the application on the two databases should also be approximately the same 1. Furthermore, our system includes one disclosure analysis component which helps users remove those characteristics which may be used by attackers to derive confidential information in production database.

As all information used to generate synthetic data in our system are contained in characteristics extracted, our disclosure analysis component can analyze and preclude the potential disclosure of confidential information at the characteristics (i.e., statistics and rules) level instead of data level.

Model Learning

The characteristics of production databases can be extracted from three parts: DDL, Data Dictionary, and Data. In order to ensure that the data is close looking or statistically similar to real data, or at least from the point of view of application testing, we need to have the statistical descriptions, *S*, and non-deterministic rules, *NR*, of real data in production databases. These two sets describe the statistical distributions or patterns of underlying data and may affect the size of relations derived as a result of the evaluations of queries the application will need to execute. Hence, they are imperative for the statistical nature of the data that determines the query performance of database application.

Extracting characteristics from data dictionary Data dictionary consists of read only base tables that store information about the database. When users execute an SQL command (e.g., CREATE TABLE, CREATE INDEX, or CREATE SEQUENCE) to create an object, all of the information about column names, column size, default values, constraints, index names, sequence starting values, and other information are stored in the form of metadata to the data dictionary. Most commercial DBMSs also collect statistical information regarding the distribution of values in a column to be created. This statistical information can be used by the query processor to determine the optimal strategy for evaluating a query. As the data in a column changes, index and column statistics can become out-of-dated and cause the query optimizer to make less-than-optimal decisions on how to process a query. SQL server automatically updates this statistical information periodically as the data in the tables changes. In our system, we have one component which simply accesses tables in data dictionary and fetch characteristics related.

Extracting characteristics from data The statistics information about columns extracted directly from data dictionary is usually with high granularity which may be insufficient to derive relatively accurate model. In practice it is usually true that userscan collect more statistics at low granularity from original data or a sample of real data themselves.

```
SELECT     Zip, Race, Age, Gender, COUNT(*),
            AVG(Balance), AVG(Income), AVG(InterestPaid),
            VAR POP(Balance), VAR POP(Income), VAR POP(InterestPaid),
            COVAR POP(Balance,Income), COVAR POP(Balance,InterestPaid),
            COVAR POP(Income,InterestPaid)
 FROM       Mortgage
GROUP BY Zip, Race, Age, Gender
HAVING    COUNT(*) > 5
```

Example 1: extracting statistics using SQL

Example 1above presents one SQL command to extract statistics at the finest granularity level. It returns all information needed to derive parameters of general location model. For example, the value from aggregate function COUNT(*) is the estimate, x_d, of the number of tuples in each cell while the values from aggregate functions (i.e., AVG, VAR POP, COVAR POP) are the estimates of mean vectors and covariance matrices respectively of the multi-variate normal distribution of each cell. It is worth pointing out all aggregate functions can be used with GROUP BY clause with CUBE or ROLLUP option if we want to extract statistics at all possible granularities. In practice, it may be infeasible to extract statistics at the finest level because statistics at the finest level may contain too much information and may be exploited by attackers to derive some confidential information about production databases.

As our goal is to generate synthetic data for database application testing, we may extract statistics which are only related to queries in workload of database software. For the workload which contains two queries shown in Figure 2, it is clear that the distribution of underlying data at some high level (instead of at the finest level) is sufficient to capture the relation with the execution time of queries in workload. For example, the approximate distribution on *Zip;Race* would satisfy the performance requirements of Q1 in workload. In this case, we may only extract statistics necessary for query performance of queries. Example 3 presents two SQL commands to extract statistics for two queries shown in Example 2.

```
Q1: SELECT AVG(Income), AVG(InterestPaid) FROM Mortgage WHERE Zip = z AND Race = r
Q2: SELECT AVG(Income) FROM Mortgage WHERE Age = a AND Zip = z
```

Example 2. Workload example of Mortgage database

```
Statistics 1: SELECT      Zip, Race, COUNT(*), AVG(Balance), AVG(InterestPaid)
              FROM        Mortgage
              GROUP BY    Zip, Race
              HAVING      COUNT(*) > 5
Statistics 2: SELECT      Zip, Age, COUNT(*), AVG(Income)
               FROM       Mortgage
              GROUP BY     Zip, Age
              HAVING       COUNT(*) > 5
```

Example 3. SQLs of extracting statistics for workload

Extracting characteristics from rule sets To derive the deterministic rule set *R*, we take advantage of the database schema, which describes the domains, the relations, and the constraints the database designer has explicitly specified. Some information (function dependencies, correlations, hierarchies etc.) can be derived from database integrity constraints such as foreign keys, check conditions, assertions, and triggers. Furthermore, users may apply some data mining tools to extract non-deterministic rules *NR* from production database. The non-deterministic rule set *NR* helps describe the statistical distributions or patterns of underlying data and may affect the size of relations derived as a result of the evaluations of queries the application will need to execute. Formally, each rule in *R* and *NR* can be represented as a declarative rule and is generally of the form:

IF <*premise*> THEN <*conclusion*> [with support *s* and confidence *c*]

The rules may include exact, strong, and probabilistic rules based on the support and confidence. We note here that complex predicates and external function references may be contained in both the condition and action parts of the rule. Anyone with subject matter expertise will be able to understand the business logic of the data and can develop the appropriate conditions and actions, which will then form the rule set.

Rule 1: IF Zip = 28223, Race = Asian, and Age in (25, 40) THEN Balance is in (20k,30k) with support s = 900 and confidence c= 90 %.

Rule 2: IF Zip = 28262 THEN Race = White with support s = 5000 and confidence c = 80 %

Example 4: The non-deterministic rules for Mortgage dataset

The above example 4 shows two non-deterministic rules for Mortgage database. We can interpret Rule 1 as there are 1000 customers with Zip = 28223, Race = Asian, and Age in (25, 40) and 90 % of them with Balance in the range of (20k, 30k). It is straightforward to see these rules can be mapped to statistics of general location model at some granularity. For example, the number of data entries in cell (28223, Asian, 25- 40, All) is

1000 and we can derive average balance of data entries in this cell from the clause Balance in (20k,30k) with confidence c= 90 %.

Balance	>1000	All	All	All		All	>1000	
Statistics 1	28223	Asian	All	All	2800	23000		75000
	28223	Black	All	All	3100	35000		89000
	.	.	.	.	.	.	.	.
	28262	White	All	All	2500	123000		112000
Statistics 2	28223	All	20	All	300		56000	
	28223	All	21	All	570		38000	
.	.	.	.	.	.	.	.	.
	28262	All	40	All	210		73000	
Rule 1	28223	Asian	25-40	All	900	(20k,30k)		
Rule 2	28262	All	All	All	5000			
	28262	White	All	All	4000			

Table 2: The table of rule sets

Fitting model using characteristics It is easy to see all characteristics (i.e., *S*, *R*, *NR*) extracted from production database can be mapped to constraints of parameters of general location model at the finest level. Table 2 shows parameter constraints derived from examples (e.g., constraint Balance > 1000, two statistics from Example 3, and two rules from Example 4.) we discussed previously.

Given those constraints, we can apply linear programming techniques to derive parameters of general location model at the finest level. However, it is infeasible to apply linear programming techniques directly in practice due to high complexity (the number of variables is linear of the number of cells *D* while the number of constraints is large). As we know, the problem of estimating the cell entries at the finest granularity subject to some linear constraints is known to be NP-hard. In our system, we combine some heuristics to derive parameters from high level constraints. For example, from Rule 1, we can initialize the number of tuples in those 30 cells (28223, Asian, Age, Gender) where Age is in (25,40) and Gender in *(*Male, Female) as 30 (900/15*x*2) when we assume the tuples are uniformly distributed among Age and Gender in cell (28223, Asian, Age, Gender).

3.5 Some Statistical Data Models

According to *Ullman (1988)* a data model is a notation for describing data and a set of operations used to manipulate the data. Many proposals for such models have been published where most of them use a graph-theoretic approach. The most prominent models are presented in the next sub-chapters. A common feature of these models is the fact that most of them deal only with macro- and metadata. The macrodata are linked to a specific topic (context) or a field of interest Several statistical populations are part of such a field of interest. A specific population is considered as a set or class of units (instances) each of which are characterized by a set of properties (attributes) which are related to each other. The attributes can be classified *(Shoshani, 1982)* according to their role. Attributes used to categorize or to index data are called category attributes and the attributes expressing summary properties are called summary attributes. The value of a category attribute is sometimes called »category« instead of »category value«.

3.5.1 SUBJECT

Chan, Shoshani (1981) consider a single multi-dimensional table which is modelled as a root tree (V, E) where the set V of nodes includes C-nodes (cluster-nodes): a cluster is a set of subordinate categories, e.g. the node »Year« has subordinates 1980, 1985, 1990. X-nodes (cross-product nodes): a cross product is the Cartesian product of sets of categories, e.g. domain (Sex) **x** domain (Year); note that the root node is of this type and refers to a complete table.

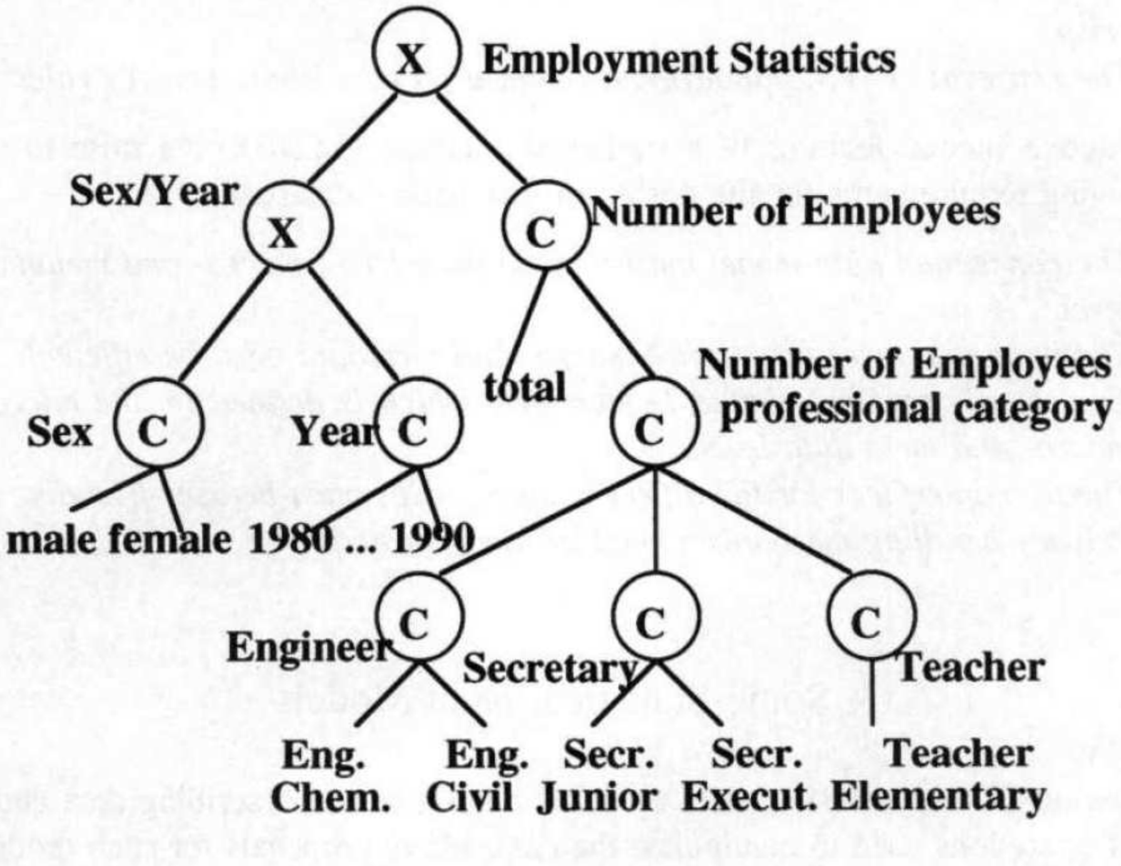


Figure 24: A Subject Tree

The left subtree represents the stub of a statistical table and is modelled by Cand X-nodes.The right subtree models the heading of the table and consists of C- and X-nodes. The different summary attributes are clustered in a C-node, indicating that the subordinate nodes are pointing to data columns in the multidimensional table. Besides of being marked with either a »C« or a »X« , the nodes are labeled according to the role they play in a context, i.e. the name of the table, the names of category attributes, the corresponding subordinate categories and the names of the summary variables are given.

The set E of edges reflects the linking of nodes of type (X, X), (X, C), (C, C), and (C,X). There are some special features of this data model:
- The tree is representing more the physical structure of a statistical table (i.e. stab and heading of a table are modelled as subtrees).
- The tree is asymmetric in its subtrees. The right subtree models the heading of the table and refers columnwise to the data.

The following set of SUBJECT operations are available:
- **Browsing** is achieved by traversing the graph
- **Search** is provided to locate the file nodes directly using specific keywords
- **Examine** locating of nodes that contain specific keywords
- **Include** allowing for the specification of predicate conditions for queries
- **Aggregation** aggregating a selected set of terminal nodes

- **Display** displaying the result of a query in a table form
- **Document** displaying the text document associated with a node

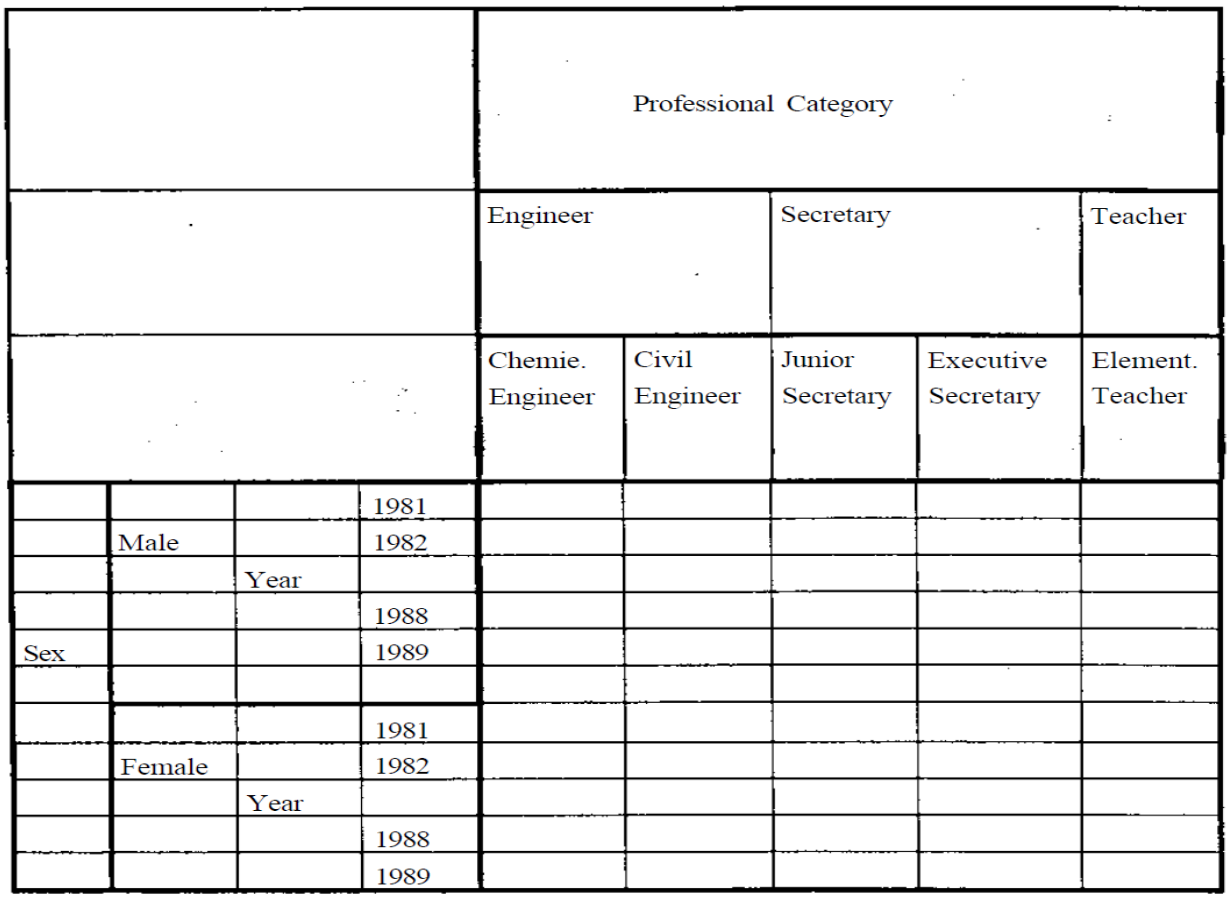
3.5.2 NF2-table structures

Ozsoyoglu and Yuan (1987) proposed a non-first-normal form (NF2) of a nested relationship type. This form is called a multi-dimensional or multi-way table in statistics, cf Nelder (1974). It is a more natural representation of a complex data-structure because it is not »flattened out« like a normalized relationship. It is characterized by a crossing and nesting of the categorical attributes which together make up the stub and heading of the table. Each cell contains the corresponding value of the summary attribute linked to the relationship type.

For example, consider the table »Professional Position in California« published in Ferri, Pisano, Rafanelli (1992) and printed in a slightly modified form below. It shows the absolute frequency distribution on the category attributes sex, year, professional category and qualification. The attributes sex and year are nested while sex and professional category are crossed.

3.5.3 STORM (Statistical Object Representation Model)

Rafanelli, Shoshani(1990) proposed a graph-oriented data model which is an enhanced version of the *SUBJECT* model.



				Professional Category				
				Engineer		Secretary		Teacher
				Chemie. Engineer	Civil Engineer	Junior Secretary	Executive Secretary	Element. Teacher
			1981					
	Male		1982					
		Year						
			1988					
Sex			1989					
			1981					
	Female		1982					
		Year						
			1988					
			1989					

Table 3 Number of professional positions in California (in 1000)

A complex data structure is used which describes a statistical object(StO). It is defined by the quadruple

$$\mathbf{StO = (N, Ca, S, f)}$$

where
N is the name of the statistical object
Ca is a (finite) set of category attributes
S is a (single) summary attribute
f links Ca and S by a root tree.

A *STORM* model is a directed, acyclic graph(DAG) consisting of several topic nodes (T-nodes). A T-node is either a root node, or a node with at least one proceeding T-node and/or at least one succeeding S-node. A S-node is the root node of subtree corresponding to a single StO.

Allowing for:

T-nodes .... topic nodes

S-nodes summary attribute nodes

X-nodes cross product nodes

C-nodes cluster nodes

a StO is represented by a *STORM* tree and a *STORM* model by a DAG.
A specific feature of *STORM* is to model non-balanced (»non-symmetric«) and heterogeneous statistical objects, cf. Rafanelli (1991). Non-symmetry arises when one category attribute is classified in a classification hierarchy (nomenclature), in which the number of levels is different depending on the category attribute referred to. For example, state, county and city form a hierarchy. Evidently, there exist states having cities but missing counties. Non-mogeneity arises when the instances of a category attribute are, in turn, classified with regard to different criteria. For example, the category attribute'Professional Category ' has categories 'Engineer', 'Secretary' and 'Teacher'. While 'Teacher' may have subordinate categories like elementary, grammar or high school teacher, 'Engineer' may be sub-classified according to the diploma degree etc.

3.5.4 CSM (Conceptual Statistical Model)

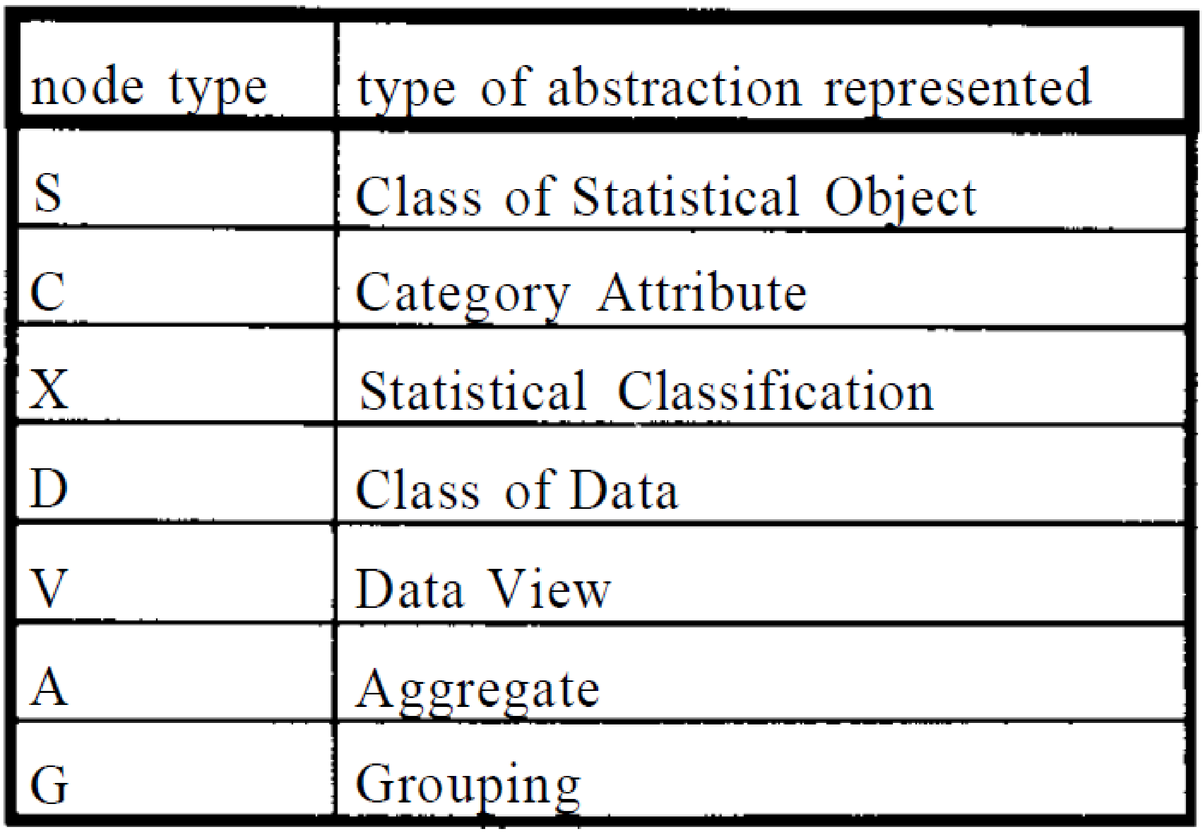
The data models presented so far were concerned with macro- and metadata. Di Battista and Batini (1988) introduced a model which covers all kind of data, however, using different paradigms.

1. Microdata or elementary data are represented by an ER-model

2. Macrodata or summary data are represented by a graph-based model similar to the *SUBJECT* data-model.

3. Metadata are embedded in the ER-model and in the nodes of model graph representing the macrodata.

The conceptual schema of the macrodata is modelled by a labeled and marked DAG. Its nodes have the following semantic (Tab. 4). The corresponding proauction rules used for designing a feasible DAG are of the type (if a node is oftype A then it has as parents nodes of type C and/or A)



node type	type of abstraction represented
S	Class of Statistical Object
C	Category Attribute
X	Statistical Classification
D	Class of Data
V	Data View
A	Aggregate
G	Grouping

Table 4: The node types of the CSM data model

3.5.5 SDM4S (Statistical Data Model based on a 4 Schema Concept)

This data model is devoted to macro- and metadata. The conceptual approach is strictly object-oriented. It has been developed and improved over a couple of years. Similar ideas have been developed in order to improve the semantics of data modelling. The concepts of an object graph with formal definitions in an (infologica) language called INFOL, of a metaobject graph with type, series and occurrence layers and of the so-called alfa-beta-gamma-tau-analysis have been introduced, which represent metadata from a substantial, regional and temporal point of view. The main features of this model are the following:

The model considers only macro- and metadata.

- The macrodata are stored physically in a relational database.
- The metadata are embedded in a statistical data dictionary (StDD). Its logical structure is described by an object-oriented (frame-oriented) system with 4 levels.

There are three classes of objects forming the data-dictionary frames.

- Statistical object
- Category value (domain)
- Summary value (domain).

The hierarchical structure is mapped by arcs of the type

- a_kind_of representing a superclass-subclass relationship,
- is_a representing a class-instance relationship,
- a_part_of representing a whole-part relationship.

Note, that the definition of the »is-a« relationship is rather unusual. The main manipulating facilities are:

- An editor for insert, update and delete operations.
- A browser uses the links between frames to explore the StDD and to list its items.

The statistical data-dictionary of the SDM4S is represented by a frame or object-oriented system. It consists of 3 types of classes (frames) and has 4 levels of abstraction.

(1) **The data model level** (root level of the frames):

There exist 3 root levels :

- Statistical objects categorized by

• the categorical attributes with category values

• and the summary attributes with summary values

- Category Values (Domain)

- Summary Values (Domain)

(2) **The Conceptual Level:**

On this level the data is described which is conceptually obtainable in the realm (the real object world) of a database regardless of its availability.

Ex.:

Persons are categorized by sex, age and summarized by the population size, i.e. by counting the number of persons classified according to their sex and age group.
Persons, Employees represent statistical objects on the conceptual level Age and Sex Category, Population Size represent conceptual domains

Example 5: The Conceptual Level

(3) **DB Schema Level:**

On this level one describes which part of the conceptual data is available as actual and stored data in the SDB. There may be gaps due to sparse data. The distinction between conceptually obtainable and actual available data makes it possible to give second best answers to a query. For example, a response could be »Census data on persons not available on a annual basis but on a 10 years basis!«.

Ex.:
Persons for Census are categorized by
sex = male/female
age = 5 years age group and summarized with
population = number with unit = 1000.
Persons for Census represent an actual statistical object
5 years age group represent an actual domain

Example 6: DB schema Level

4) Instance Level

On this level the individual metadata and individual values are described which are linked to the objects on the instance level.

Ex.:
The metadata corresponding to a cell of a multi-dimensional table are kept here together with the name
of the statistical object and the values of the categorical and summary attributes.

Example 7: Instance Level

3.5.6 Modelling metadata using an eER-diagram

A comprehensive formal way to model metadata is to use an extended entity relationship model (eERD). An eER diagram is a graph which consists of **the nodes of type (rectangular)** representing entity types or classes of meta-objects which have similiar characteristics.

The nodes of type (rhombs) representing types of sub- or supordinating as well as a temporal preordering **the arcs** reflecting structural and semantic relationships between the metadata. The arcs are marked with the maximum complexity number.In order to simplify the notation the eER diagram is drawn without any characteristics, i.e. attributes of the entity types of the metadata. Of course, adetailed structure can be visualized by a stepwise refinement of each subset of entity types or classes. This is demonstrated below (Fig. 4).

The ER-diagram shown below (Fig. 5) is visualizing die attributes of the (Meta) entity types (attribute), (statistical object) and (file). It should be considered as an example for a further refinement of an eER-diagram. it reflectsthe semantic, statistical and storage views on the metadata.

The operations needed to create, update and explore the metadata are:

- loading, editing a metadata graph

- scrolling

- browsing

- searching

- zooming

- listing, printing, storing.

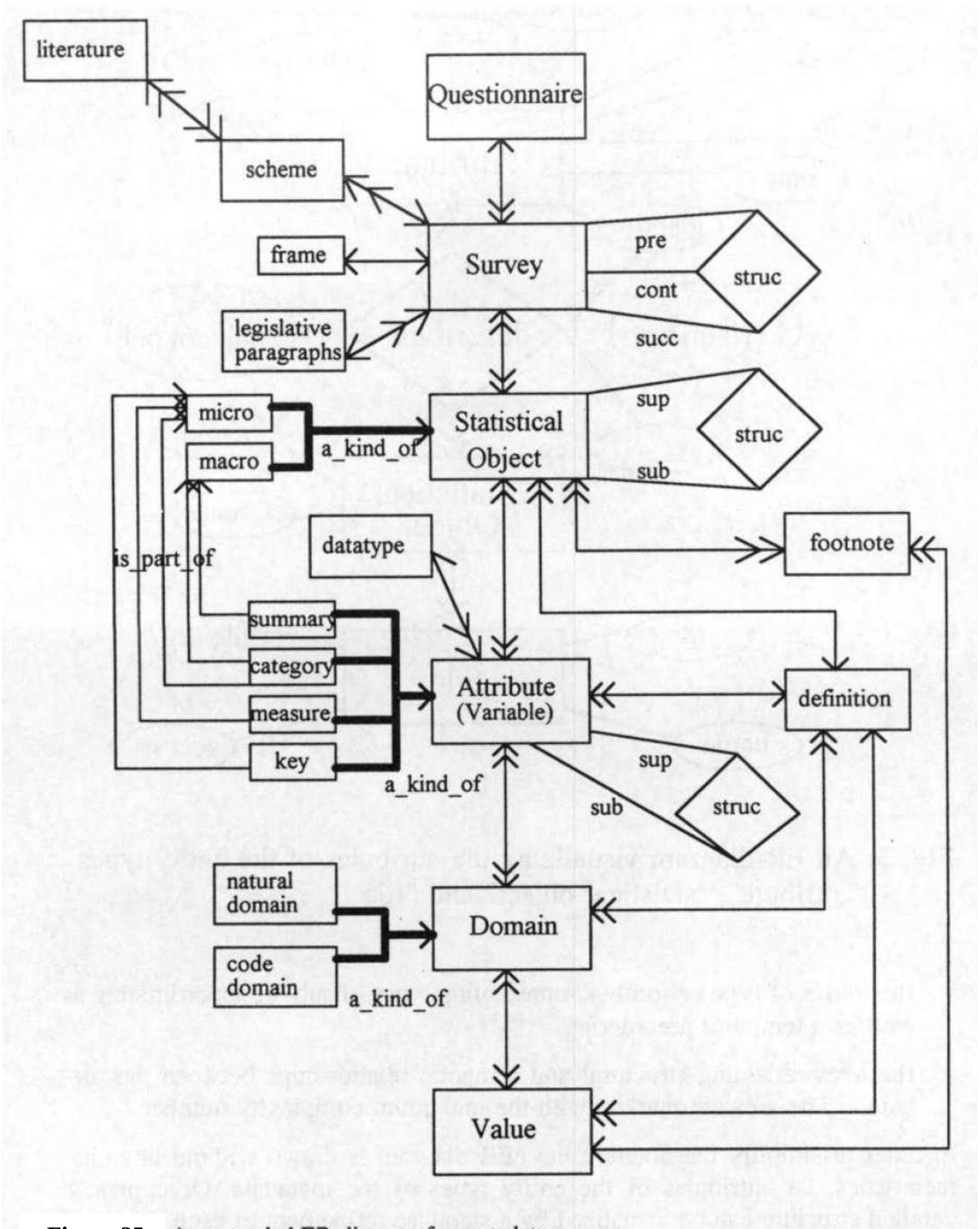


Figure 25 An extended ER-diagram for metadata

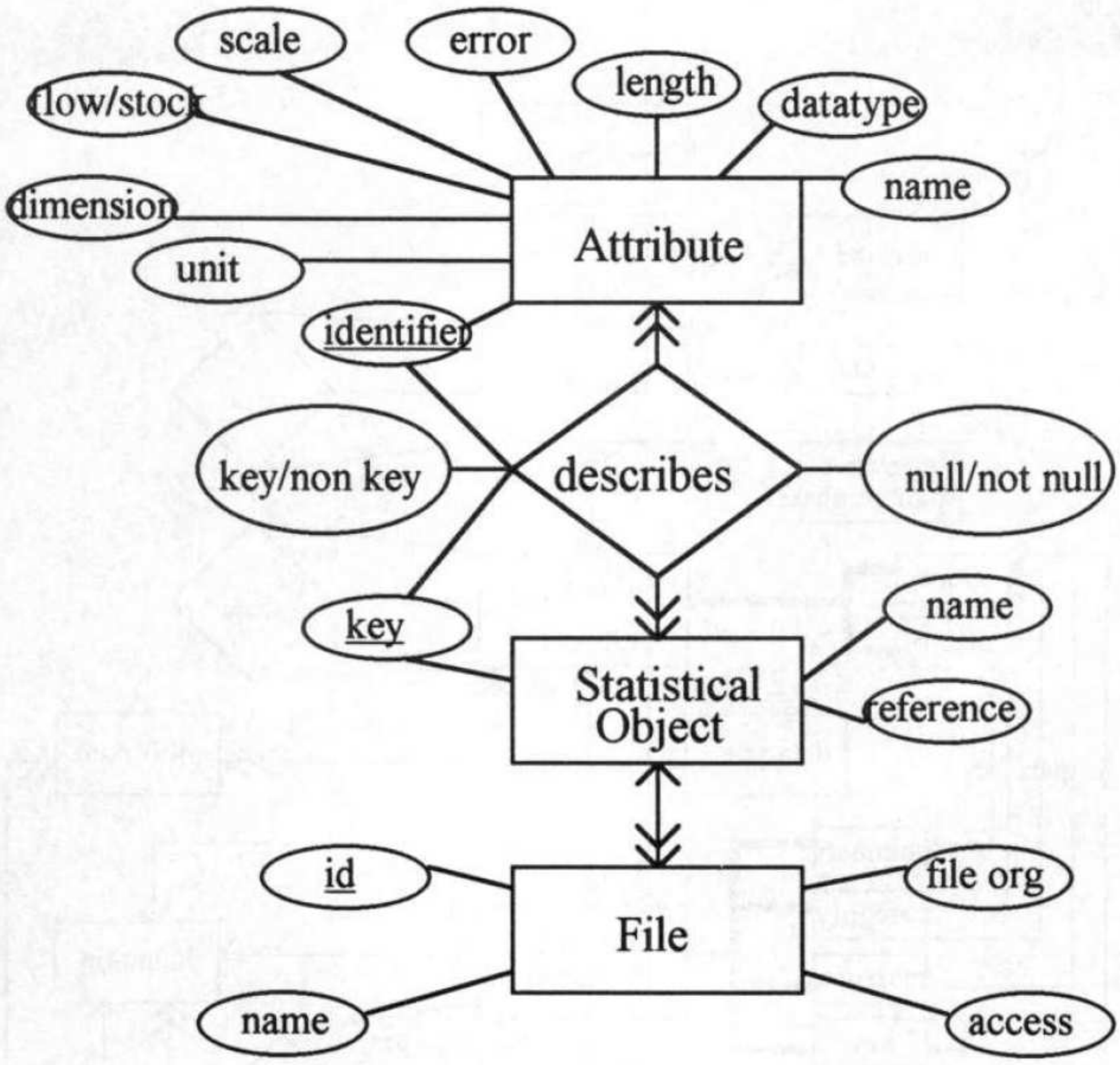


Fig. 26 An ER-diagram visualizing the attributes of the entity types "attribute", "staistical object" and "file"

4.0 Conclusion

With the introduction to basic concepts Statistics database system, students must have acquired the basic sense of the central importance of statistical databases in today’s information systems environment. And with this, they should be able to design and model various Statistical database for real life situations.

5.0 Summary

In this unit we have learnt that:

- ❖ Basic features and different definitions of Statistical database system and the operations for manipulating these structures, as well as certain constraints that the database should obey.
- ❖ Various concepts of Statistical database policies and models were extensively discussed.
- ❖ We also design Statistics database models for use in defining the target state and the subsequent planning needed to hit the target state..

6.0 Tutor Marked Assignment

- ❖ 1. (a) Define Statistical database system

  (b) Design a model to solve one real life problem using a

  Statistical database approach.

- ❖ 2. (a) Explain the basic concepts in Statistical database system.

7.0 Further Reading and Other Resources

Duncan et. al., 1993, *Private Lives and Public Policies*,. p. 157

“Restricted Access Procedures” by the Confidentiality and Data Access Committee (April
2002) at http://www.fcsm.gov/committees/cdac/cdacra9.doc.

http://www.fcsm.gov/committees/cdac/resources.html

Di Battista, G. and Batini, C.(1988), Design of Statistical Databases: A Methodology for the Conceptual Step, Inform. Systems, Vol.13, No. 4, pp. 407-422.
Van den Berg, G.M., de Feber, E., and de Greef, P. (1992), Analysing StatisticalData Processing, in Eurostat, New Technologies and Techniques for Statistics,Luxembourg, pp 102-111.

Chan, P., and Shoshani, A. (1981), SUBJECT: A Directory Driven System for Large Statistical Databases, in Proc. of the LBL Workshop on Statistical Database Management, Lawrence Berkeley Lab, Berkeley, CA.

Cubitt, R. et al.(eds.) (1986), Proceedings of the Third International Workshop on Statistical and Scientific Database Management, Eurostat, Luxembourg. Eurostat (1992), New Technologies and Techniques for Statistics, Luxembourg.

Ferri, F., Pisano, M.T., and Rafanelli, M.(1992), A Object Oriented Visual Definition Language for Statistical Data, in New Technologies and Techniquesfor Statistics, Proc. of the conference, Bonn, pp.320-339.

Hinterberger, H., French, J.C.(eds.) (1992), Proceedings of the Sixth InternationalWorkshop on Statistical and Scientific Database Management, Departmentof Informatik, ETH Zürich, Zürich.

Lenz, H.-J.(1993), M3-Database Design, Micro-, Macro- and Metadata Modelling, in F. Faulbaum (ed.), SoftStat '93 Advances in Statistical Software

D. Chays, S. Dan, P. Frankl, F. Vokolos, and E. Weyuker. A framework for testing database applications. In *Proceedings of the ISSTA*. Portland, Oregon, 2000.

A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. *Statistical Journal of the United Nations ECE*, 18:363–371, 2001.

J.L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman Hall, 1997.

Module 2 The Statistical Database System

Unit 2 Statistical Data Analysis, Mining and Decision Tree

1.0 Introduction

2.0 Objective

3.0 Statistical analysis of Oracle performance data using R

3.1 Data Mining and the Semantic Conference Organizer

3.2 Data Mining and Decision Tree

4.0 Conclusion

5.0 Summary

6.0 Tutor Marked Assignment

7.0 Further Reading and Other Resources

1.0 Introduction

A Statistical data analysis of Database can be carried out using Basic R-Statistical function with little laborious configuration. This analysis can be made possible with the use of Statistical tables and graphs for statistic inferences. In the same manner, the organization of a technical meeting, workshop, or conference involving submitted abstracts or full-text documents can be quite an onerous task. To gain a sense of what topic each submission addresses may require more than just a quick glimpse at the title or abstract. The use of automated indexing and text mining revolutionises the manner and speed of information assessment and organization.

2.0 Objective

At the end of this unit, you should be able to:

- ❖ Appreciate the features of Statistical analysis in Oracle performance data using R database management system (RDBMS)
- ❖ Use High-Level Conceptual Data mining and the Semantic conference organizer Database system.
- ❖ State the major concepts of Data mining and Decision Tree.
- ❖ Distinguishes between the major components of Data mining and Semantic conferencing.

3.0 Statistical analysis of Oracle performance data using R

R is without doubt the Open Source tool of choice for statistical analysis, it contains a huge variety of statistical analysis techniques – rivalled only by hugely expensive commercial products such as SAS and SPSS.

Installing R:

R can be install in linux as a standard package: **On Windows, one** may wish **to use the Revolution R binaries.** At times, there may trouble installing the 32-bit binaries on the system as they conflicted with the 64-bit JDBC. The easiest way to setup a connection to Oracle to install the RJDBC package is hereby stated:

```
[oracle@GuysOEL ~]$ R

R version 2.12.1 (2010-12-16)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-redhat-linux-gnu (64-bit)

<snip>

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("RJDBC")
```

Getting data from Oracle into R

Once R is installed, it’s pretty simple to get data out of Oracle and into R.

Here’s a very short snippet that grabs data from the V$SQL table:

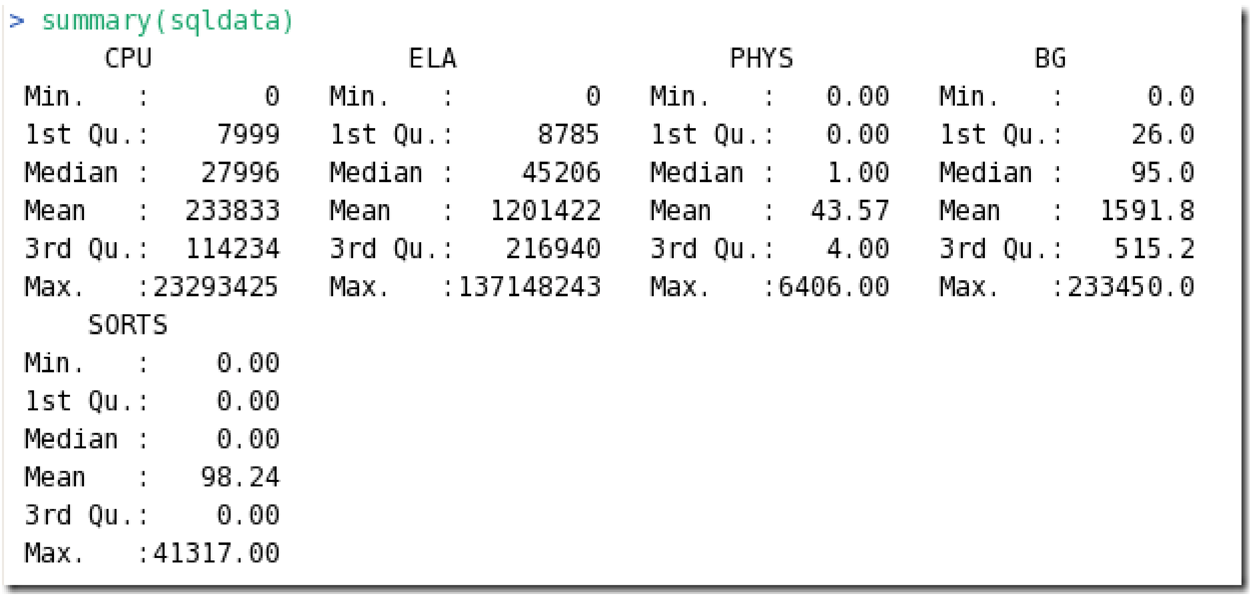
```
 1:  library(RJDBC)
 2:
 3:  drv <- JDBC("oracle.jdbc.driver.OracleDriver",
 4:                "/ora11/home/jdbc/lib/ojdbc6.jar")
 5:
 6:  conn <- dbConnect(drv,"jdbc:oracle:thin:@hostname:1521: service","username","password")
 7:  sqldata<-dbGetQuery(conn, "SELECT cpu_time cpu,elapsed_time ela,disk_reads phys,
 8:                          buffer_gets bg,sorts sorts
 9:                       FROM V$SQL ")
10:  summary(sqldata)
```

Let’s look at that line by line:

Line	Comments
1	The library command loads the RJDBC module, which will provide connectivity to Oracle.
3	We create a driver object for the Oracle JDBC driver. The second argument is the location of the Oracle JDBC jar file, almost always $ORACLE_HOME/jdbc/lib/ojdbc6.jar.
6	Connect to the Oracle database using standard JDBC connections strings
7	Create an R dataset from the result set of a query. In this case, we are loading the contents of the V$SQL table.
10	The R “summary” package provides simple descriptive statistics for each variable in the provided dataset.

Basic R statistical functions

R has hundreds of statistical functions, in the above example we used "summary", which prints descriptive statistics. The output is shown below; mean, medians, percentiles, etc:



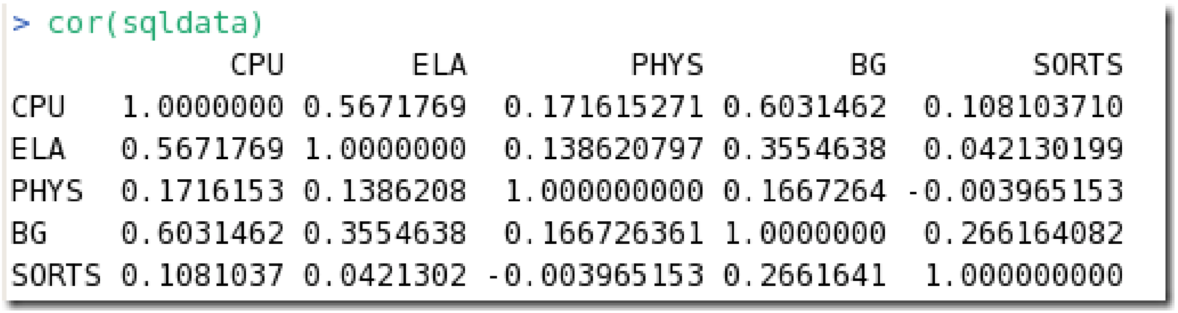
```
> summary(sqldata)
      CPU                  ELA                   PHYS                 BG
 Min.   :       0   Min.   :        0   Min.   :   0.00   Min.   :     0.0
 1st Qu.:    7999   1st Qu.:     8785   1st Qu.:   0.00   1st Qu.:    26.0
 Median :   27996   Median :    45206   Median :   1.00   Median :    95.0
 Mean   :  233833   Mean   :  1201422   Mean   :  43.57   Mean   :  1591.8
 3rd Qu.:  114234   3rd Qu.:   216940   3rd Qu.:   4.00   3rd Qu.:   515.2
 Max.   :23293425   Max.   :137148243   Max.   :6406.00   Max.   :233450.0
     SORTS
 Min.   :    0.00
 1st Qu.:    0.00
 Median :    0.00
 Mean   :   98.24
 3rd Qu.:    0.00
 Max.   :41317.00
```

Figure 27.

Correlation

Statistical correlation reveals the association between two numeric variables. If two variables always increase or decrease together the correlation is 1; if two variables are absolutely random with respect of each other then the correlation tends towards 0.

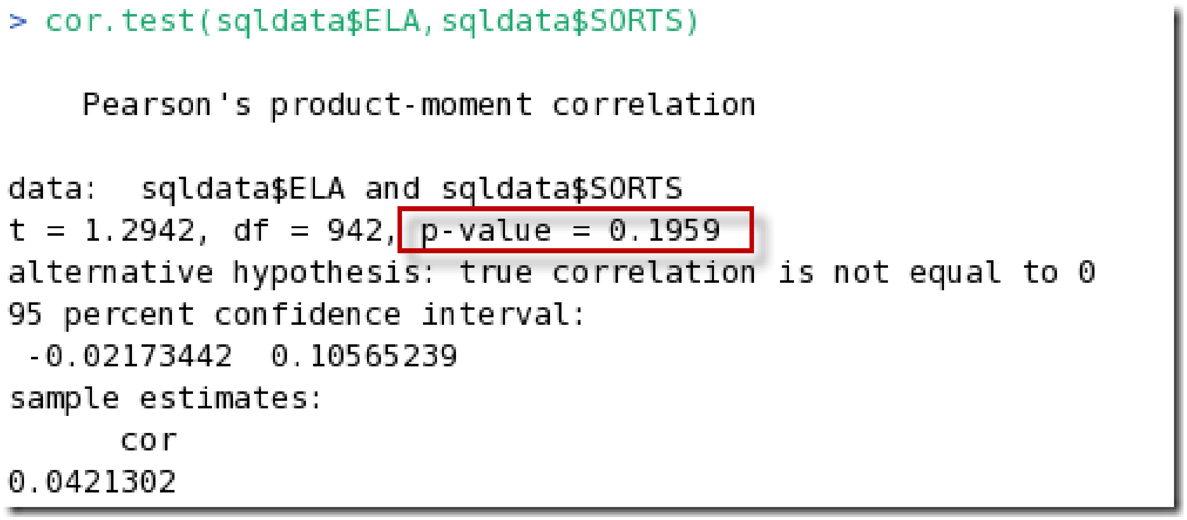
correlation **prints the correlation between every variable in the data set:**



```
> cor(sqldata)
            CPU       ELA         PHYS        BG        SORTS
CPU   1.0000000 0.5671769  0.171615271 0.6031462  0.108103710
ELA   0.5671769 1.0000000  0.138620797 0.3554638  0.042130199
PHYS  0.1716153 0.1386208  1.000000000 0.1667264 -0.003965153
BG    0.6031462 0.3554638  0.166726361 1.0000000  0.266164082
SORTS 0.1081037 0.0421302 -0.003965153 0.2661641  1.000000000
```

Figure 28.

Correlation test calculates the correlation coefficient and prints out the statistical significance of the correlation, which allows you to determine if there is a significant relationship between the two variables. So does the number of sorts affect response time? Let's find out:

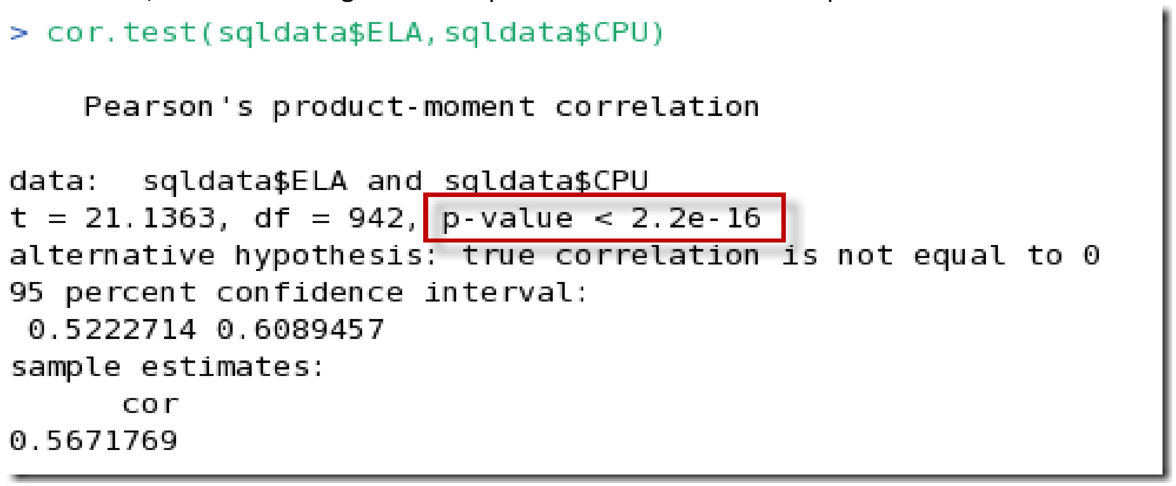


```
> cor.test(sqldata$ELA,sqldata$SORTS)

	Pearson's product-moment correlation

data:  sqldata$ELA and sqldata$SORTS
t = 1.2942, df = 942, p-value = 0.1959
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.02173442  0.10565239
sample estimates:
      cor
0.0421302
```

Figure 29

The p-value is 0.19 which indicates no significant relationship – p values of no more than 0.05 (one chance in 20) are usually requires before we assume statistical significance.

On the other hand, there is a strong relationship between CPU time and Elapsed time:



```
> cor.test(sqldata$ELA,sqldata$CPU)

    Pearson's product-moment correlation

data:  sqldata$ELA and sqldata$CPU
t = 21.1363, df = 942, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5222714 0.6089457
sample estimates:
      cor
0.5671769
```

Figure 30

Plotting

plot prints a scattergram chart. Here's the output from plot(sqldata$ELA,sqldata$CPU):

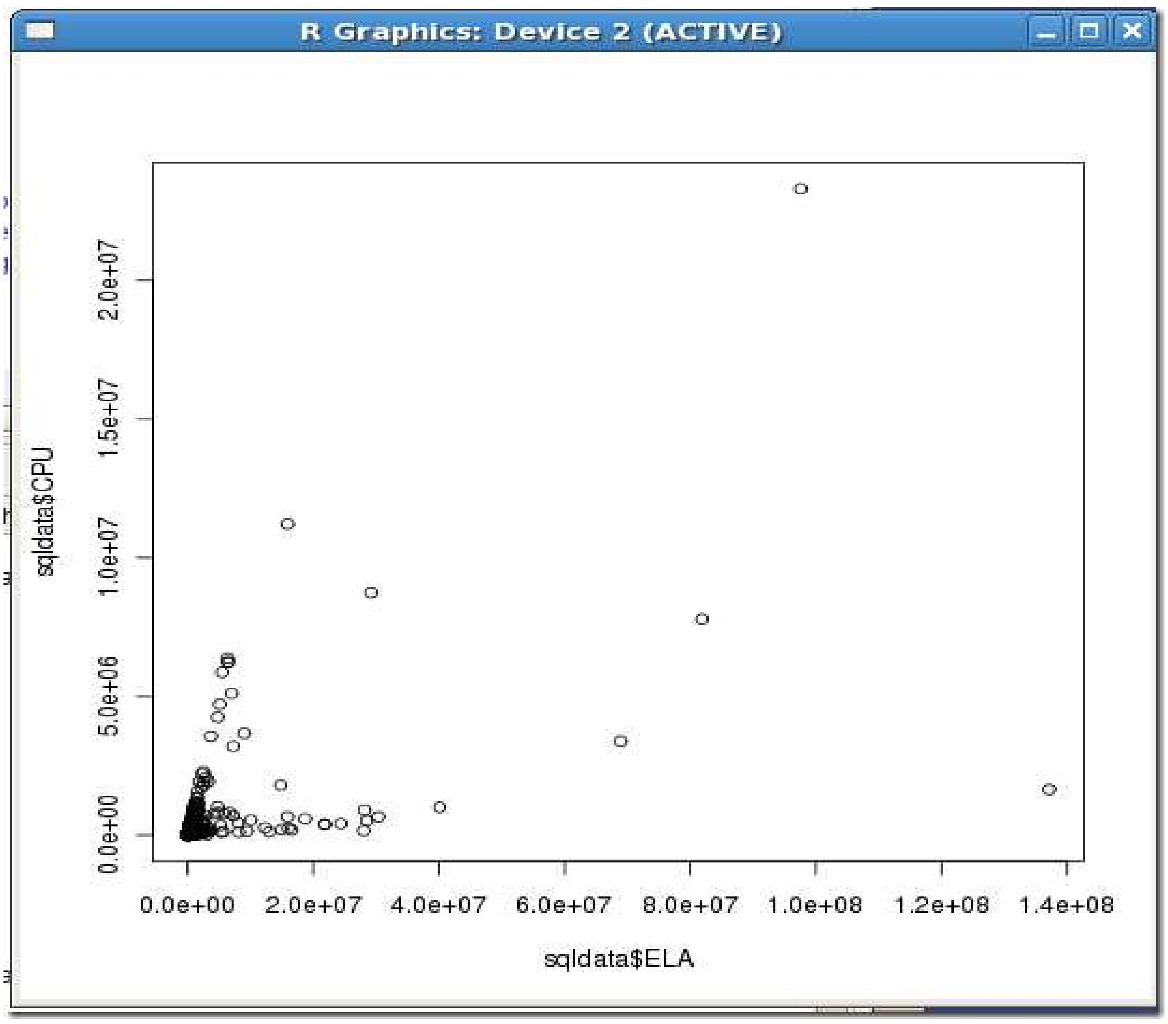
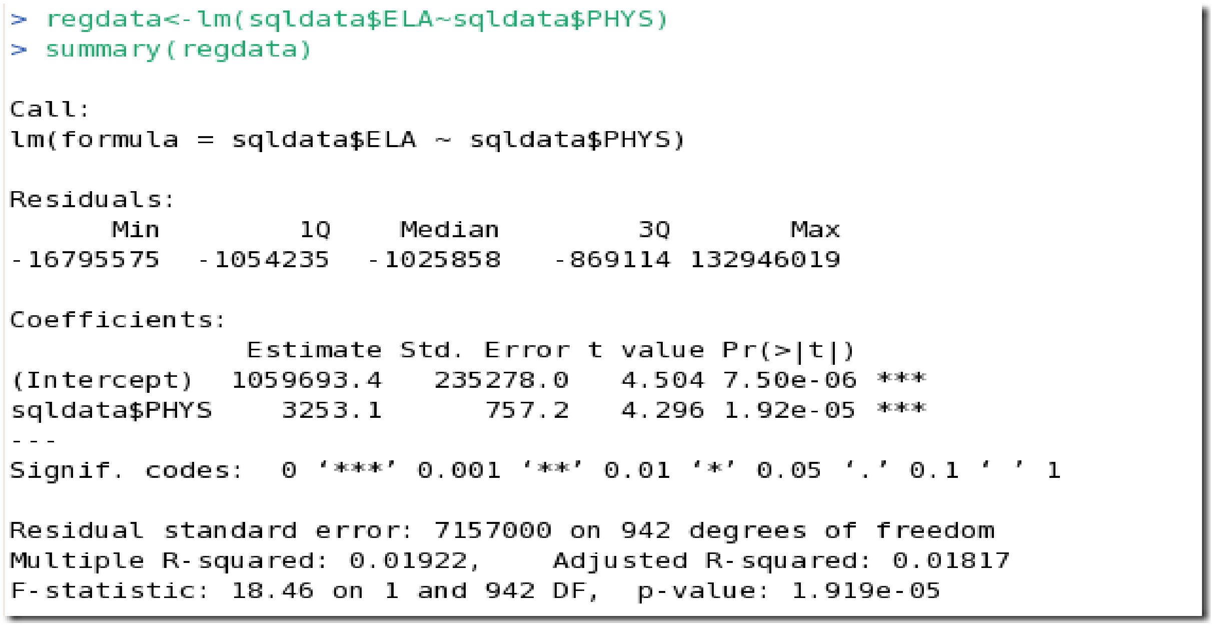


Figure 31

Regression

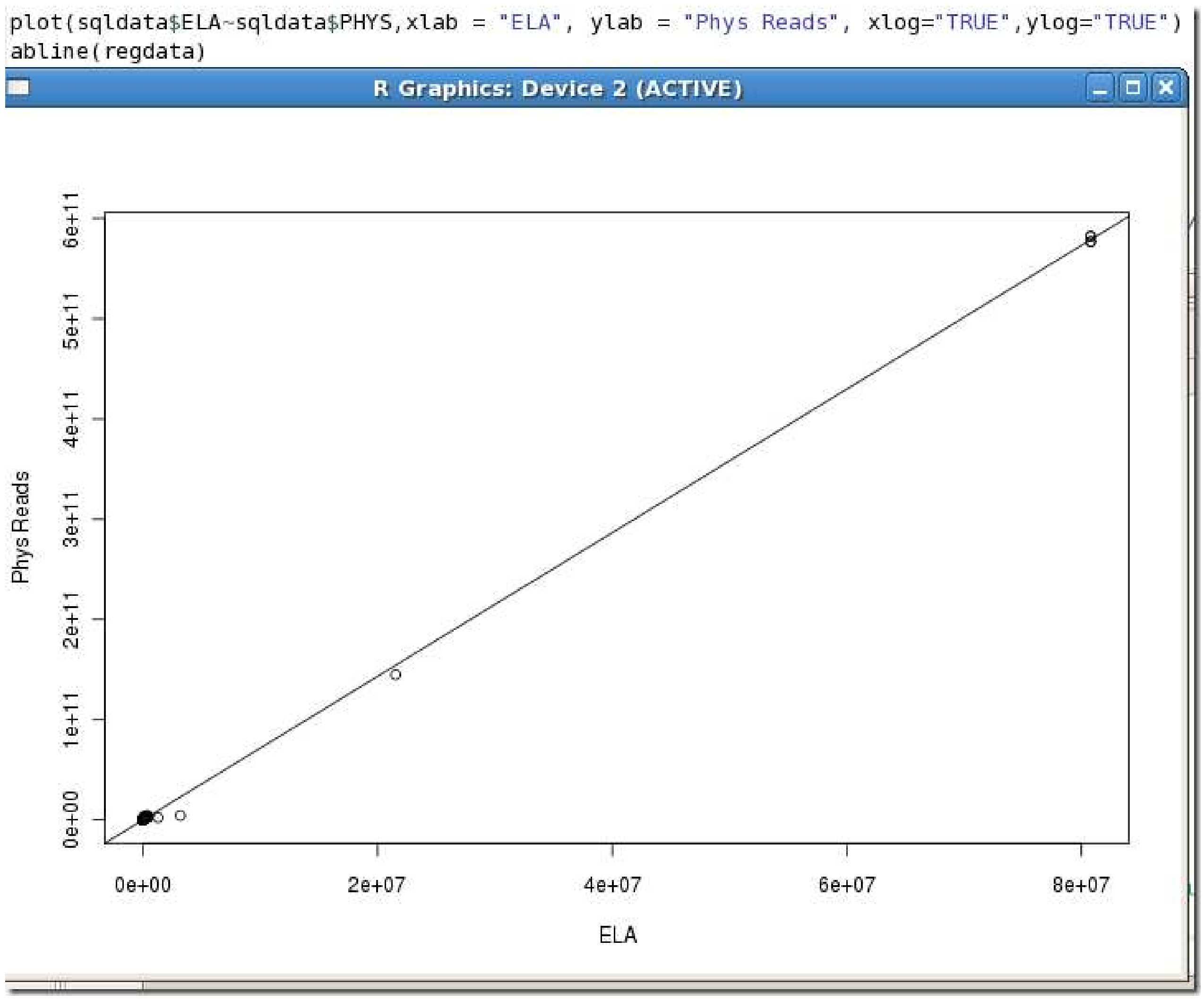
Regression is used to draw “lines of best fit” between variables. In the simplest case, we use the “lm” package to create a linear regression model between two variables (which we call “regdata” in the example). The summary function prints a summary of the analysis:



```
> regdata<-lm(sqldata$ELA~sqldata$PHYS)
> summary(regdata)

Call:
lm(formula = sqldata$ELA ~ sqldata$PHYS)

Residuals:
      Min        1Q    Median        3Q       Max
-16795575  -1054235  -1025858   -869114 132946019

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1059693.4   235278.0   4.504 7.50e-06 ***
sqldata$PHYS    3253.1      757.2   4.296 1.92e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7157000 on 942 degrees of freedom
Multiple R-squared: 0.01922,    Adjusted R-squared: 0.01817
F-statistic: 18.46 on 1 and 942 DF,  p-value: 1.919e-05
```

Figure 32

This might seem a little mysterious if your statistics is a bit rusty, but the data above tells us that there is a significant relationship between elapsed time (ELA) and physical reads (PHYS) and gives us the gradient and Y axis intercept if we wanted to draw the relationship. We can get R to draw a graph, and plot the original data by using the plot at abline functions:



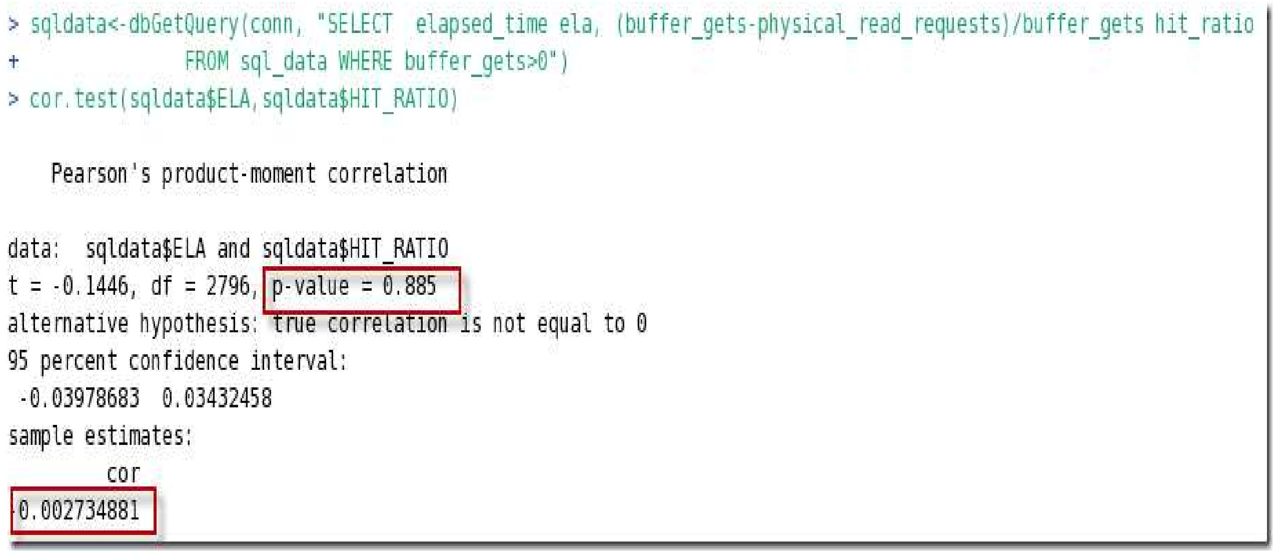
```
plot(sqldata$ELA~sqldata$PHYS,xlab = "ELA", ylab = "Phys Reads", xlog="TRUE",ylog="TRUE")
abline(regdata)
```

Figure 33

Testing a hypothesis

One of the benefits of statistical analysis is you can test hypotheses about your data. For instance, what about we test the until recently widely held notion that the buffer cache hit rate is a good measure of performance. We might suppose if that were true that SQL statements with high buffer cache hit rates would show smaller elapsed times than those with low buffer cache hit rates. To be sure, there are certain hidden assumptions underlying that hypothesis, but for the sake of illustration let’s use R to see if our data supports the hypothesis.

Simple correlation is a fair test for this; all we need to do is see if there is a statistically significant correlation between hit rate and elapsed time. Here’s the analysis:



```
> sqldata<-dbGetQuery(conn, "SELECT  elapsed_time ela, (buffer_gets-physical_read_requests)/buffer_gets hit_ratio
+               FROM sql_data WHERE buffer_gets>0")
> cor.test(sqldata$ELA,sqldata$HIT_RATIO)

    Pearson's product-moment correlation

data:  sqldata$ELA and sqldata$HIT_RATIO
t = -0.1446, df = 2796, p-value = 0.885
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.03978683  0.03432458
sample estimates:
        cor
-0.002734881
```

Figure 34

The correlation is close to 0, and the statistical significance way higher than the widely accepted .05 threshold for statistical significance. Statements with high hit ratios do not show statistically significantly lower elapsed times that SQLs with low hit ratios.

3.1 Statistical Data Mining and the Semantic Conference Organizer

The organization of a technical meeting, workshop, or conference involving submitted abstracts or full-text documents can be quite an onerous task. To gain a sense of what topic each submission addresses may require more than just a quick glimpse at the title or abstract. The use of automated indexing and text mining can revolutionize the manner and speed of information assessment and organization. In this section, the use of Latent Semantic Indexing (LSI) for probing and labelling conference abstracts using an intuitive Web interface and client-server internal software design using grid-based middleware such as NetSolve, is demonstrated.

3.1.1 Background

Creating a conference manually can be a burdensome task. After all papers have been submitted, the human organizer must then group the papers into sessions. The session topics can be decided either before or after the organizer has a feel for the material covered in the papers. And since the average conference has around one hundred papers submitted to it, the organizer must shuffle these papers between topics trying to find a workable fit for the papers and the sessions to which they are assigned. Of course, one person trying to fit fifty to one hundred papers into about twenty sessions will lose context very quickly. Switching rapidly between sessions will cause confusion, and renaming sessions or assigning different topics may cause the entire conference to get reworked. Many times the human organizer will only work with document surrogates such as an

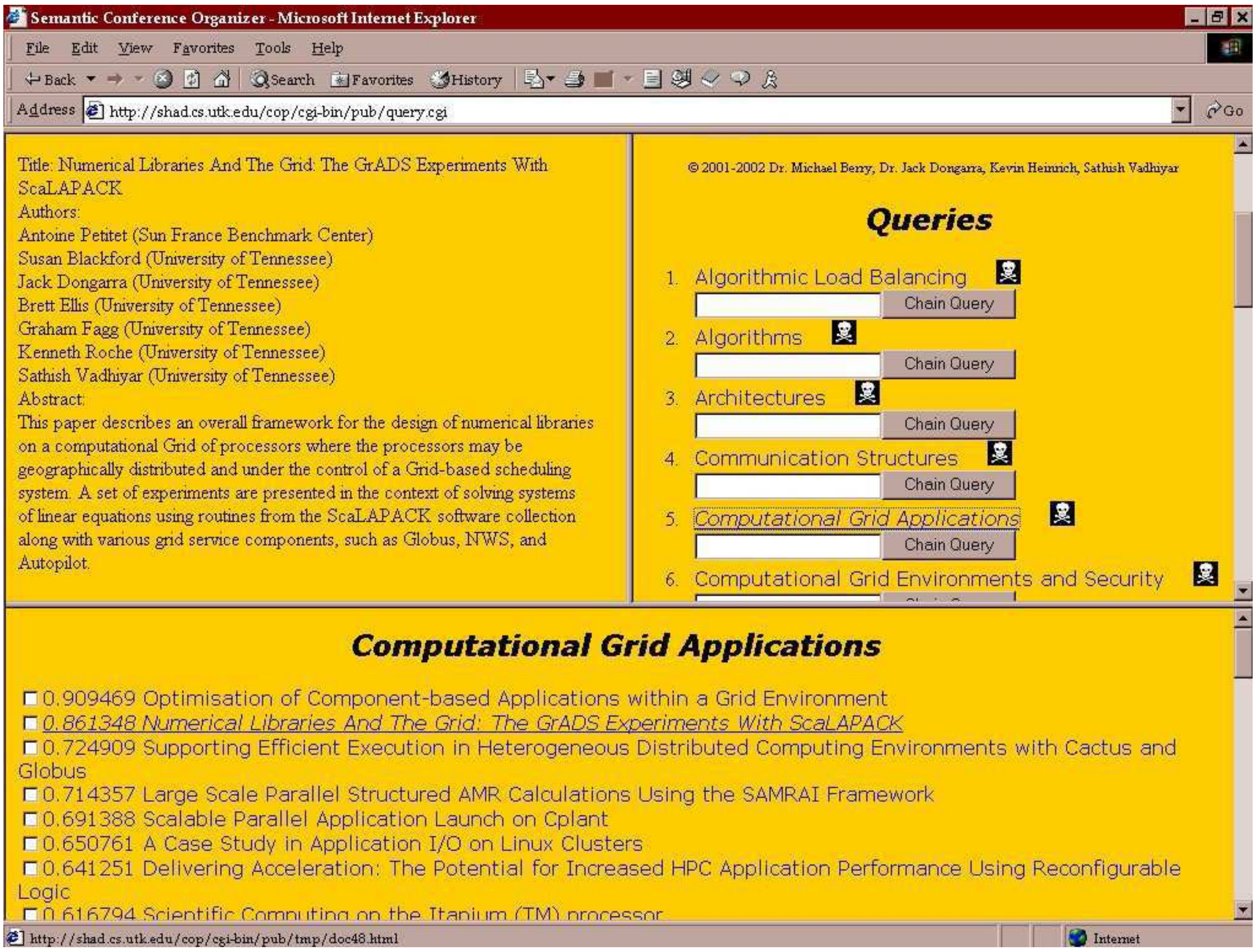
abstract or simply the paper title, so often papers will be misclassified due to summarization errors.
Also note that a significant amount of time must be spent reading and re-reading abstracts to remember what each paper's subject is. Manually creating a conference takes anywhere from a day to a week or longer. With such a combinatorial problem confronting the person who manually organizes the conference, the need for some sort of automated assistance is justified in hopes of reducing the hours spent in creating a conference.

3.1.2 Latent Semantic Indexing

In order for the Semantic Conference Organizer to be useful, it must replace the most time-consuming of tasks undertaken when creating a conference—reading. There are several techniques and algorithms used in the field of information retrieval that enable relevant documents to be retrieved to meet a specific need without requiring the user to read each document. The model used by the Semantic Conference Organizer is latent semantic indexing or LSI.

Once the document collection is received, it must be parsed into bare words called *tokens*. All punctuation and capitalization is ignored. In addition, articles and other common, non-distinguishing words are discarded. In effect, each document is viewed as a bag of words upon which operations can be performed. Once the bag of words has been formed, a term-by-document matrix is created where the entries of the matrix are the weighted frequencies associated with the corresponding term in the appropriate document.

The weight of a term within a document is a nonnegative value used to describe the correlation between that term and the corresponding document. A weight of zero indicates no correlation. In general, each weight is the product of a local and global component. A simplistic method of obtaining weights is to assign the local component as the frequency of the word within the document and the global component as the log of the proportion of total documents to the number of documents in which the term appears. Such a method is known as a tf-idf (term frequency, inverse-document frequency) weighting scheme. The aim of any scheme is to measure similarity within a document while at the same time measuring the dissimilarity of a document from the other documents within the collection.


Semantic Conference Organizer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://shad.cs.utk.edu/cop/cgi-bin/pub/query.cgi
Go
Chain Query
http://shad.cs.utk.edu/cop/cgi-bin/pub/tmp/doc48.html
Internet

Title: Numerical Libraries And The Grid: The GrADS Experiments With ScaLAPACK
Authors:
Antoine Petitet (Sun France Benchmark Center)
Susan Blackford (University of Tennessee)
Jack Dongarra (University of Tennessee)
Brett Ellis (University of Tennessee)
Graham Fagg (University of Tennessee)
Kenneth Roche (University of Tennessee)
Sathish Vadhiyar (University of Tennessee)
Abstract:
This paper describes an overall framework for the design of numerical libraries on a computational Grid of processors where the processors may be geographically distributed and under the control of a Grid-based scheduling system. A set of experiments are presented in the context of solving systems of linear equations using routines from the ScaLAPACK software collection along with various grid service components, such as Globus, NWS, and Autopilot.

© 2001-2002 Dr. Michael Berry, Dr. Jack Dongarra, Kevin Heinrich, Sathish Vadhiyar

Queries

1. Algorithmic Load Balancing
2. Algorithms
3. Architectures
4. Communication Structures
5. Computational Grid Applications
6. Computational Grid Environments and Security

Computational Grid Applications

☐ 0.909469 Optimisation of Component-based Applications within a Grid Environment
☐ 0.861348 Numerical Libraries And The Grid: The GrADS Experiments With ScaLAPACK
☐ 0.724909 Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus
☐ 0.714357 Large Scale Parallel Structured AMR Calculations Using the SAMRAI Framework
☐ 0.691388 Scalable Parallel Application Launch on Cplant
☐ 0.650761 A Case Study in Application I/O on Linux Clusters
☐ 0.641251 Delivering Acceleration: The Potential for Increased HPC Application Performance Using Reconfigurable Logic
☐ 0.616794 Scientific Computing on the Itanium (TM) processor

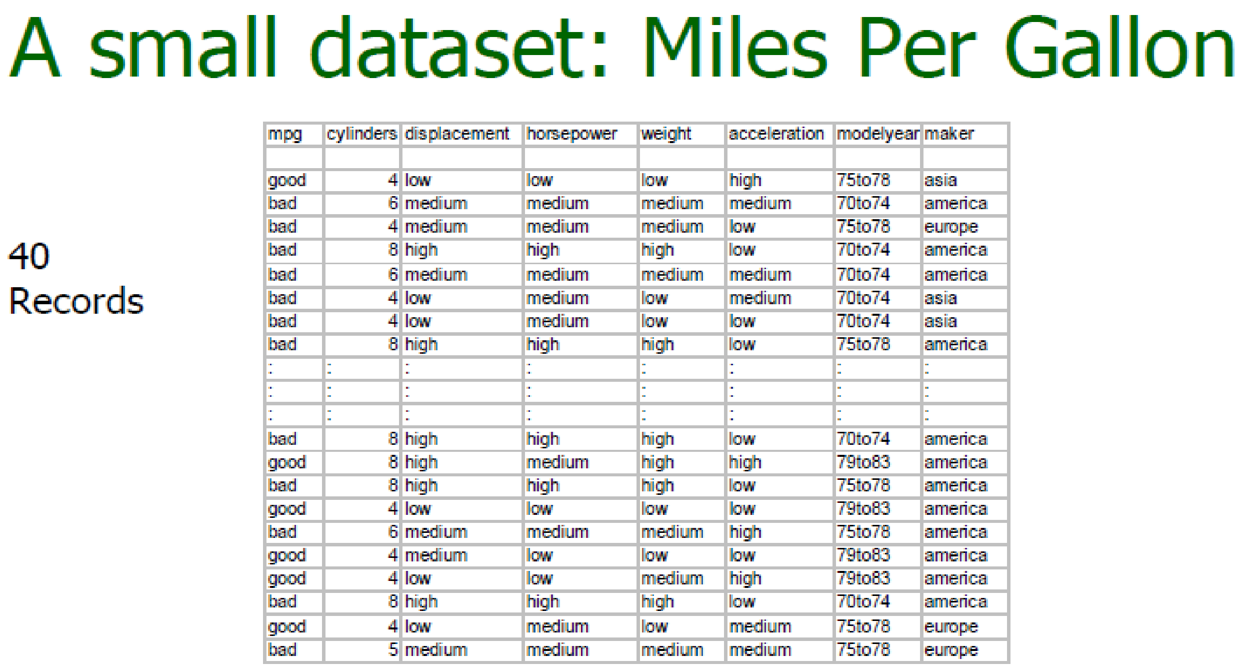
FIGURE 35. Sample layout of the Semantic Conference Organizer.

3.2 Data Mining and Decision Tree

since Data Mining is all about automating the process of searching for patterns in the data. It is necessary to known which patterns are interesting, which might be mere illusions and how can they be exploited? And the answer will turn out to be the engine that drives decision tree learning.

3.2.1 Learning Decision Trees

A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output, and to decide which attribute should be tested first, simply find the one with the highest information gain.

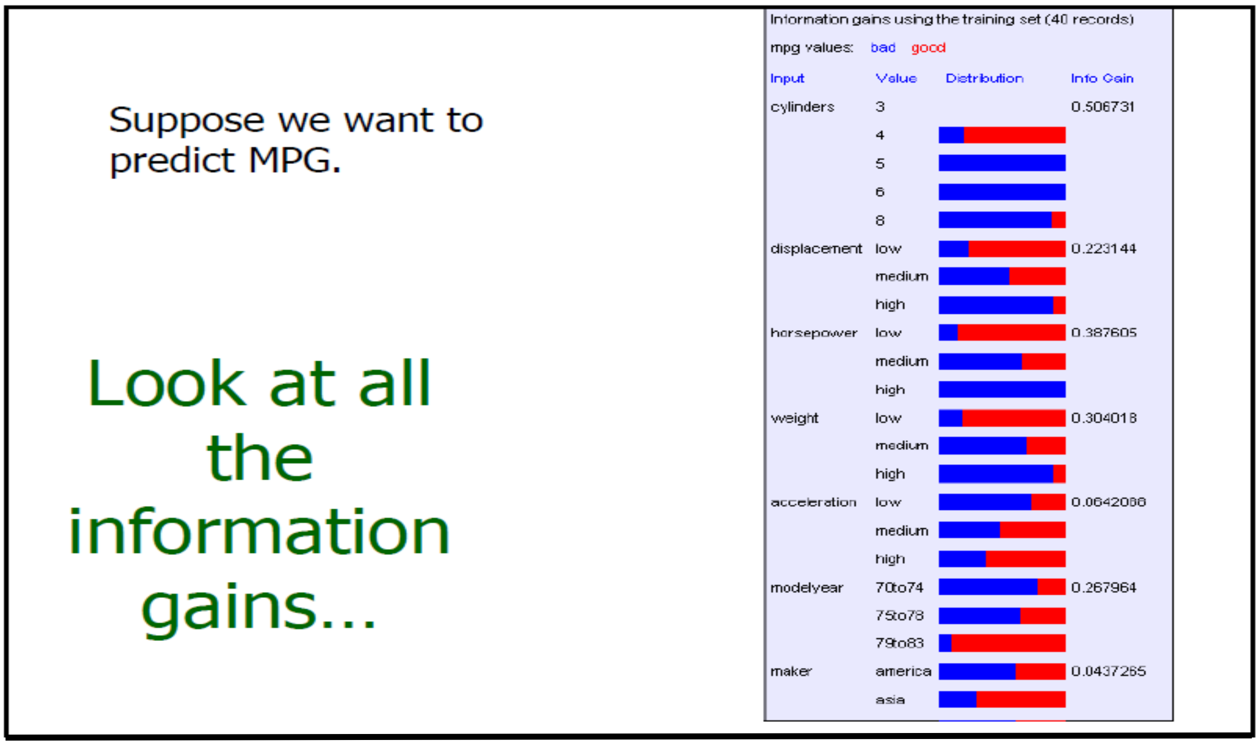


A small dataset: Miles Per Gallon

40 Records

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

Table 5: Table of Dataset



Suppose we want to predict MPG.

Look at all the information gains...

Information gains using the training set (40 records)

mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	3		0.506731
	4		
	5		
	6		
	8		
displacement	low		0.223144
	medium		
	high		
horsepower	low		0.387605
	medium		
	high		
weight	low		0.304018
	medium		
	high		
acceleration	low		0.0642086
	medium		
	high		
modelyear	70to74		0.267964
	75to78		
	79to83		
maker	america		0.0437265
	asia		

Figure 36

The final tree

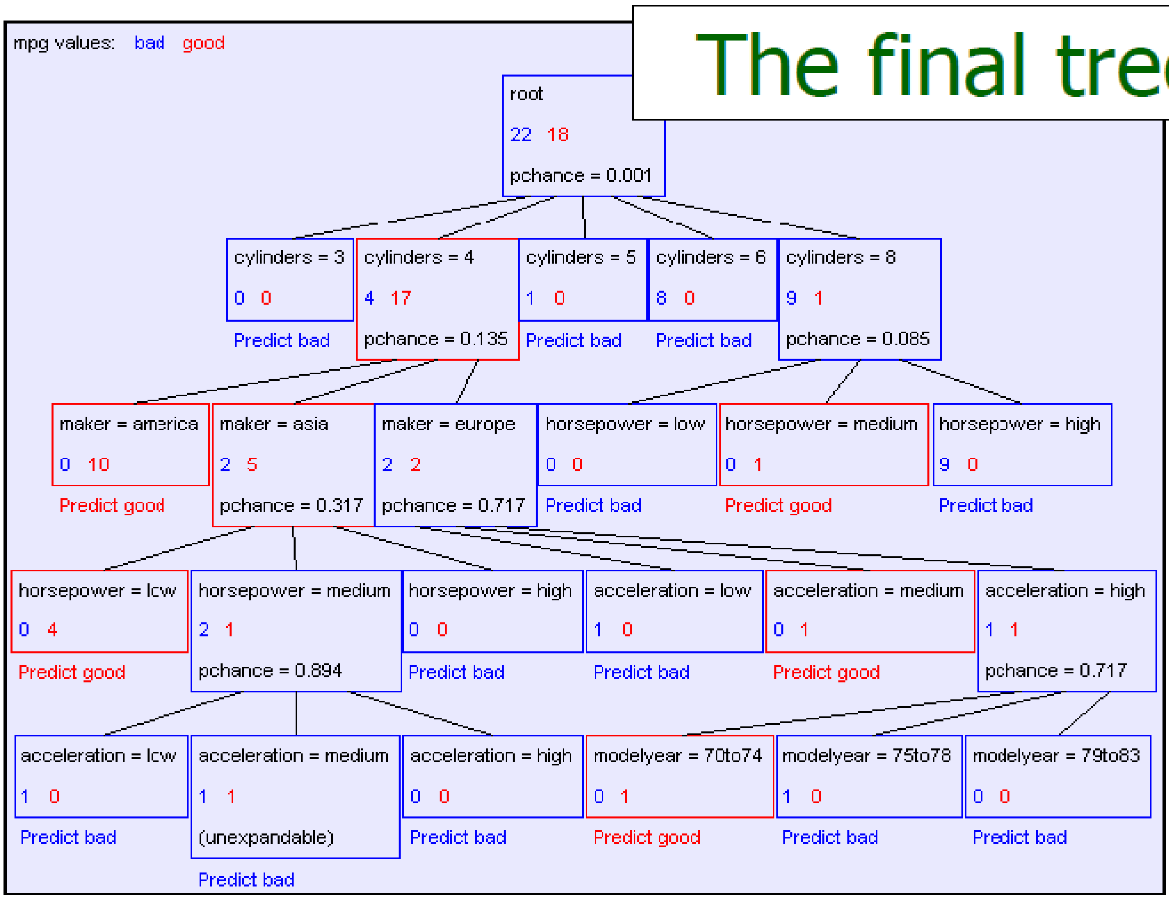

mpg values: bad good
root 22 18 pchance = 0.001
cylinders = 3 0 0 Predict bad
cylinders = 4 4 17 pchance = 0.135
cylinders = 5 1 0 Predict bad
cylinders = 6 8 0 Predict bad
cylinders = 8 9 1 pchance = 0.085
maker = america 0 10 Predict good
maker = asia 2 5 pchance = 0.317
maker = europe 2 2 pchance = 0.717
horsepower = low 0 0 Predict bad
horsepower = medium 0 1 Predict good
horsepower = high 9 0 Predict bad
horsepower = low 0 4 Predict good
horsepower = medium 2 1 pchance = 0.894
horsepower = high 0 0 Predict bad
acceleration = low 1 0 Predict bad
acceleration = medium 0 1 Predict good
acceleration = high 1 1 pchance = 0.717
acceleration = low 1 0 Predict bad
acceleration = medium 1 1 (unexpandable) Predict bad
acceleration = high 0 0 Predict bad
modelyear = 70to74 0 1 Predict good
modelyear = 75to78 1 0 Predict bad
modelyear = 79to83 0 0 Predict bad

Figure 37

Summary of Basic Decision Tree Building

BuildTree(DataSet,Output)

- If all output values are the same in DataSet, return a leaf node that says "predict this unique output"
- If all input values are the same, return a leaf node that says "predict the majority output"
- Else find attribute X with highest Info Gain
- Suppose X has nX distinct values (i.e. X has arity nX).
    - Create and return a non-leaf node with nX children.
    - The i'th child should be built by calling BuildTree(DSi,Output)

Where DSi built consists of all those records in DataSet for which X = ith distinct value of X.

4.0 Conclusion

There's tons of data in our Oracle databases that could benefit from statistical analysis – not the least the performance data in the dynamic performance views, ASH and AWR. We use statistical tests in Spotlight on Oracle to extrapolate performance into the future and to set some of the alarm thresholds. Using R, you have easy access to the most sophisticated statistical analysis techniques and as I hope I've shown, you can easily integrate R with Oracle data.

5.0 Summary

In this unit we have learnt that:

- How to use Statistical analysis tools in database system and collection of related information (data) in a structured way using 'R' method
- Statistical Database analysis is a tool of program that manages the database structure and that control shared access to the data in the database.
- The advantages include controlling redundancy, restricting unauthorized access, saving time etc, with flexibility, economies of scale, and potential for enforcing standards, are some of the implications of Statistics database approach.

6.0 Tutor Marked Assignment

1. (a) Differentiate between the following:

   (i) Data mining and information (ii) Semantic conferencing and database management.

   (b) Demonstrate a real life situation where you can use statistical analysis to solve human's problem.

2. (a) What are the characteristics of Semantic conferencing.

   (b) Mention the advantages of using Decision Tree management.

7.0 Further Reading and Other Resources 2010

Annual Subscription to STAN: OECD Structural Analysis Statistics Online Database on the OECD iLibrary:

Wiebren de Jonge, Compromising statistical databases responding to queries about means, ACM Transactions on Database Systems, Volume 8, Issue 1 (March 1983), Pages: 60 – 80

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science 41:391-407, 1990.

R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, Boston, MA, 1999.

M. Berry and M. Browne. Understanding Search Engines: Mathematical Modeling and Text Retrieval. SIAM, Philadelphia, PA, 1999.

M. Berry, Z. Drmaˇc, and E. Jessup. Matrices, Vector Spaces, and Information Retrieval. SIAM Review 41:335-362, 1999.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

C4.5 : Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan

Learning Classification Trees, Wray Buntine, Statistics and Computation (1992), Vol 2, pages 63-73

Kearns and Mansour, On the Boosting Ability of Top-Down Decision Tree Learning Algorithms, STOC: ACM Symposium on Theory of Computing, 1996“

Module 2 The Statistical Database System

Unit 3 Computer Security and Statistical Databases

1.0 Introduction

2.0 Objective

3.0 Overview of Statistical Database

3.1 Inference from a Statistical Database

3.2 Query Restriction

3.3 Partitioning

3.4 Query Denial and Information Leakage

3.5 Perturbation

4.0 Conclusion

5.0 Summary

6.0 Tutor Marked Assignment

7.0 Further Reading and Other Resources

1.0 Introduction

In this unit, it is imperative for us to take a look at the unique security issues that relate to statistical databases and know why the database administrator must prevent, or at least detect, the statistical user who attempts to gain individual information through one or a series of statistical queries.
The security problem of a statistical database is to limit the use of the database so that no sequence of statistical queries is sufficient to deduce confidential or private information.

Statistical databases often incorporate support for advanced statistical analysis techniques, such as correlations, which go beyond SQL. They also pose unique security concerns, which were the focus of much research, particularly in the late 1970s and early to mid 1980s. In a statistical database, it is often desired to allow query access only to aggregate data, not individual records.

And securing such a database is a difficult problem, since intelligent users can use a combination of aggregate queries to derive information about a single individual. Some common approaches are:

- only allowing aggregate queries (SUM, COUNT, AVG, STDEV, etc.)
- rather than returning exact values for sensitive data like income, only return which partition it belongs to (e.g. 35k-40k)
- return imprecise counts (e.g. rather than 141 records met query, only indicate 130-150 records met it.)
- don't allow overly selective WHERE clauses

- audit all users queries, so users using system incorrectly can be investigated
- use intelligent agents to detect automatically inappropriate system use

2.0 Objective

At the end of this unit, you should be able to:

- ❖ Students should be able to use the features of Query Restriction, Partitioning, Perturbation and information leakage of database management system (DBMS)
- ❖ Use High-Level Conceptual Query Denial and Information leakage.
- ❖ State the major concepts Perturbation and Query restriction.
- ❖ Draw the major components of Perturbation, Naming Conventions and Design Issues in data management system

3.0 Overview of Statistical Database

A statistical database (SDB) is one that provides data of a statistical nature, such as counts and averages. The term *statistical database* is used in two contexts:

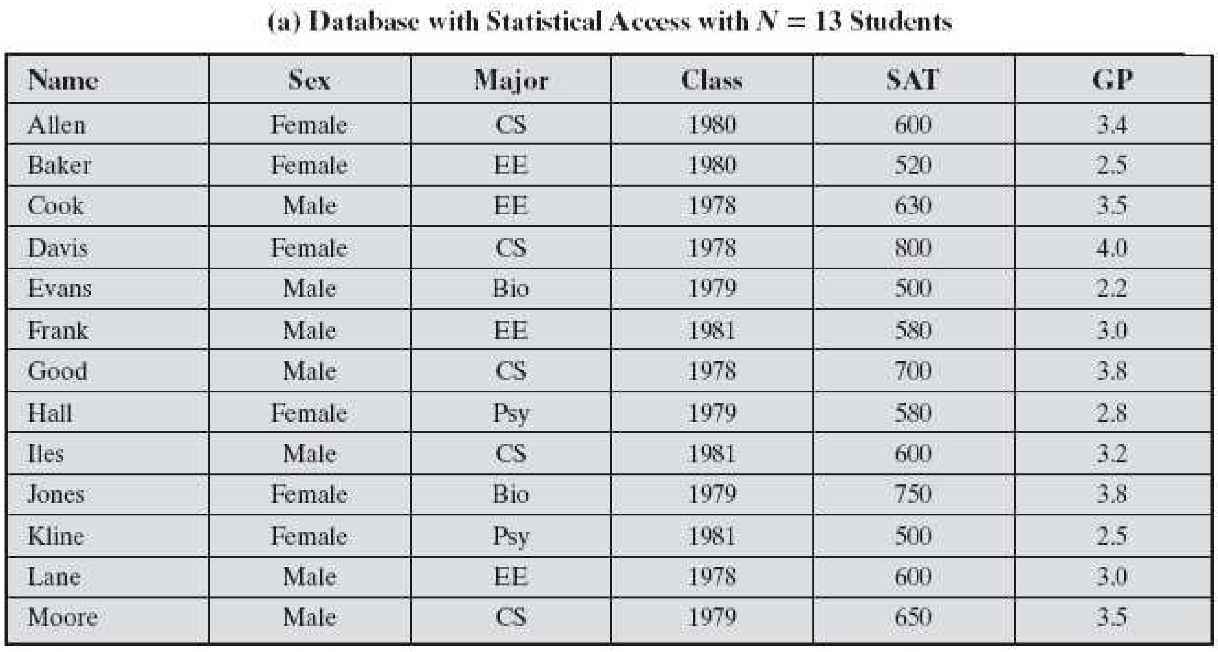
- **Pure statistical database:** This type of database only stores statistical data. An example is a census database. Typically, access control for a pure SDB is straightforward: Certain users are authorized to access the entire database.

- **Ordinary database with statistical access:** This type of database contains individual entries; this is the type of database discussed so far in this chapter. The database supports a population of non-statistical users who are allowed access to selected portions of the database using DAC, RBAC, or MAC. In addition, the database supports a set of statistical users who are only permitted statistical queries. For these latter users, aggregate statistics based on the underlying raw data are generated in response to a user query, or may be pre-calculated and stored as part of the database.

It is essentially important to know that, design of a statistical database should utilize a statistical security management facility to enforce the security constraints at the conceptual model level. Information revealed to users is well defined in the sense that it can at most be reduced to non-decomposable information involving a group of individuals. In addition, the design also takes into consideration means of storing the query information for auditing purposes, changes in the database, users' knowledge, and some security measures.

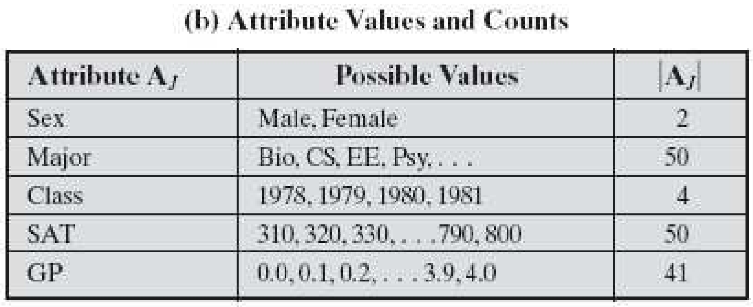
For the purposes of this section, we are concerned only with the latter type of database and, for convenience; refer to this as an SDB. The access control objective for an SDB system is to provide users with the aggregate information without compromising the confidentiality of any individual entity represented in the database. The security problem is one of inference. The database administrator must prevent, or at least detect, the statistical user who attempts to gain individual information through one or a series of statistical queries.

For this discussion, we use the abstract model of a relational database table shown as Figure 5.7. There are *N* individuals, or entities, in the table and *M* attributes. Each attribute A_j has $|A_j|$ possible values, with x_{ij} denoting the value of attribute $_j$ for entity $_i$. Table 5.3, taken from an example that we use in the next few paragraphs. The example is a database containing 13 confidential records of students in a university that has 50 departments.



(a) Database with Statistical Access with N = 13 Students

Name	Sex	Major	Class	SAT	GP
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	Psy	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	Psy	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5



(b) Attribute Values and Counts

Attribute A_j	Possible Values	$\lvert A_j \rvert$
Sex	Male, Female	2
Major	Bio, CS, EE, Psy, . . .	50
Class	1978, 1979, 1980, 1981	4
SAT	310, 320, 330, . . . 790, 800	50
GP	0.0, 0.1, 0.2, . . . 3.9, 4.0	41

Table 6: Ordinary Database with Statistical Access

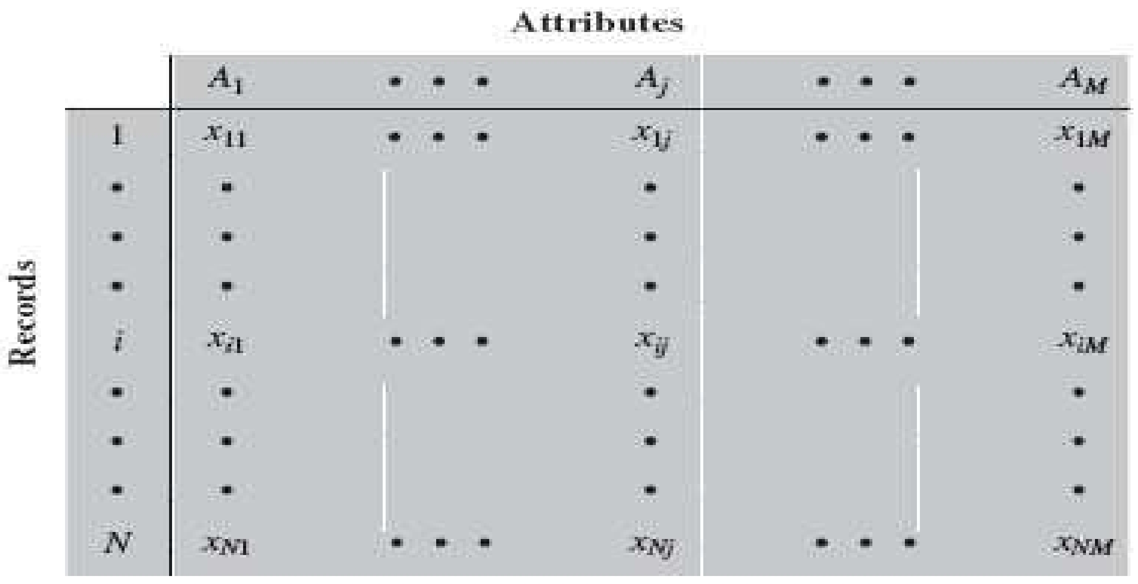


Figure 38. Abstract Model of a Relational Database

Statistics are derived from a database by means of a **characteristic formula**, *C*, which is a logical formula over the values of attributes. A characteristic formula uses the operators OR, AND, and NOT (+, ·, ~), written here in order of increasing priority. A characteristic formula specifies a subset of the records in the database. For example, the formula

$$(\text{Sex} = \text{Male}) \cdot ((\text{Major} = \text{CS}) + (\text{Major} = \text{EE}))$$

specifies all male students majoring in either CS or EE. For numerical attributes, relational operators may be used. For example, ($GP > 3.7$) specifies all students whose grade point average exceeds 3.7. For simplicity, we omit attribute names when they are clear from context. Thus, the preceding formula becomes Male · (CS + EE).

The **query set** of characteristic formula *C*, denoted as X(*C*), is the set of records matching that characteristic. For example, for *C* = Female · CS, X(*C*) consists of records 1 and 4, the records for Allen and Davis.

A statistical query is a query that produces a value calculated over a query set. Table 5.4 lists some simple statistics that can be derived from a query set. Examples: **count**(Female · CS) = 2; **sum**(Female · CS, SAT) = 1400.

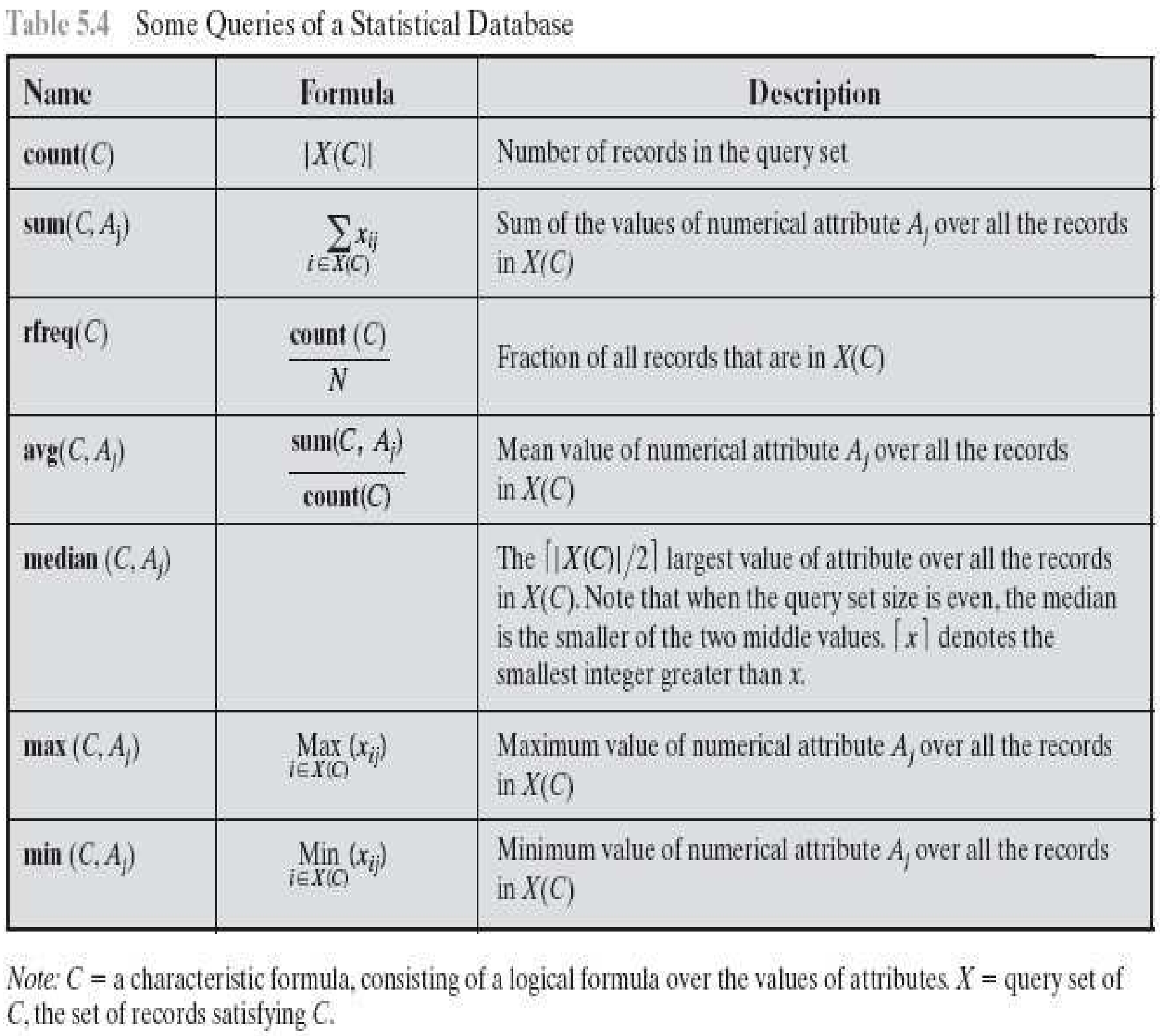


Table 5.4 Some Queries of a Statistical Database

Name	Formula	Description
count(C)	$\lvert X(C)\rvert$	Number of records in the query set
sum(C, A_j)	$\sum_{i \in X(C)} x_{ij}$	Sum of the values of numerical attribute A_j over all the records in $X(C)$
rfreq(C)	$\frac{\textbf{count}(C)}{N}$	Fraction of all records that are in $X(C)$
avg(C, A_j)	$\frac{\textbf{sum}(C, A_j)}{\textbf{count}(C)}$	Mean value of numerical attribute A_j over all the records in $X(C)$
median (C, A_j)		The $\lceil \lvert X(C)\rvert/2 \rceil$ largest value of attribute over all the records in $X(C)$. Note that when the query set size is even, the median is the smaller of the two middle values. $\lceil x \rceil$ denotes the smallest integer greater than x.
max (C, A_j)	$\underset{i \in X(C)}{\text{Max}} (x_{ij})$	Maximum value of numerical attribute A_j over all the records in $X(C)$
min (C, A_j)	$\underset{i \in X(C)}{\text{Min}} (x_{ij})$	Minimum value of numerical attribute A_j over all the records in $X(C)$

Note: C = a characteristic formula, consisting of a logical formula over the values of attributes. X = query set of C, the set of records satisfying C.

Table 7: Some Queries of a Statistical Database

3.1 Inference from a Statistical Database

A statistical user of an underlying database of individual records is restricted to obtaining only aggregate, or statistical, data from the database and is prohibited access to individual records. The inference problem in this context is that a user may infer confidential information about individual entities represented in the SDB. Such an inference is called a **compromise**. The compromise is positive if the user deduces the value of an attribute associated with an individual entity and is negative if the user deduces that a particular value of an attribute is not associated with an individual entity. For example, the statistic **sum** (EE· Female, GP) = 2.5 compromises the database if the user knows that Baker is the only female EE student.

In some cases, a sequence of queries may reveal information. For example, suppose a questioner knows that Baker is a female EE student but does not know if she is the only one. Consider the following sequence of two queries:

count (EE · Female) = 1
sum (EE · Female, GP) = 2.5

This sequence reveals the sensitive information.

The preceding example shows how some knowledge of a single individual in the database can be combined with queries to reveal protected information. For a large database, there may be few or no opportunities to single out a specific record that has a unique set of characteristics, such as being the only female student in a department. Another angle of attack is available to a user aware of an incremental change to the database. For example, consider a personnel database in which the sum of salaries of employees may be queried. Suppose a questioner knows the following information:

- Salary range for a new systems analyst with a BS degree is $[50K, 60K]
- Salary range for a new systems analyst with a MS degree is $[60K, 70K]

Suppose two new systems analysts are added to the payroll and the change in the sum of the salaries is $130K. Then the questioner knows that both new employees have an MS degree.

In general terms, the inference problem for an SDB can be stated as follows. A characteristic function *C* defines a subset of records (rows) within the database. A query using *C* provides statistics on the selected subset. If the subset is small enough, perhaps even a single record, the questioner may be able to infer characteristics of a single individual or a small group. Even for larger subsets, the nature or structure of the data may be such that unauthorized information may be released.

3.2 Query Restriction

SDB implementers have developed two distinct approaches to protection of an SDB from inference attacks (Figure 5.8):

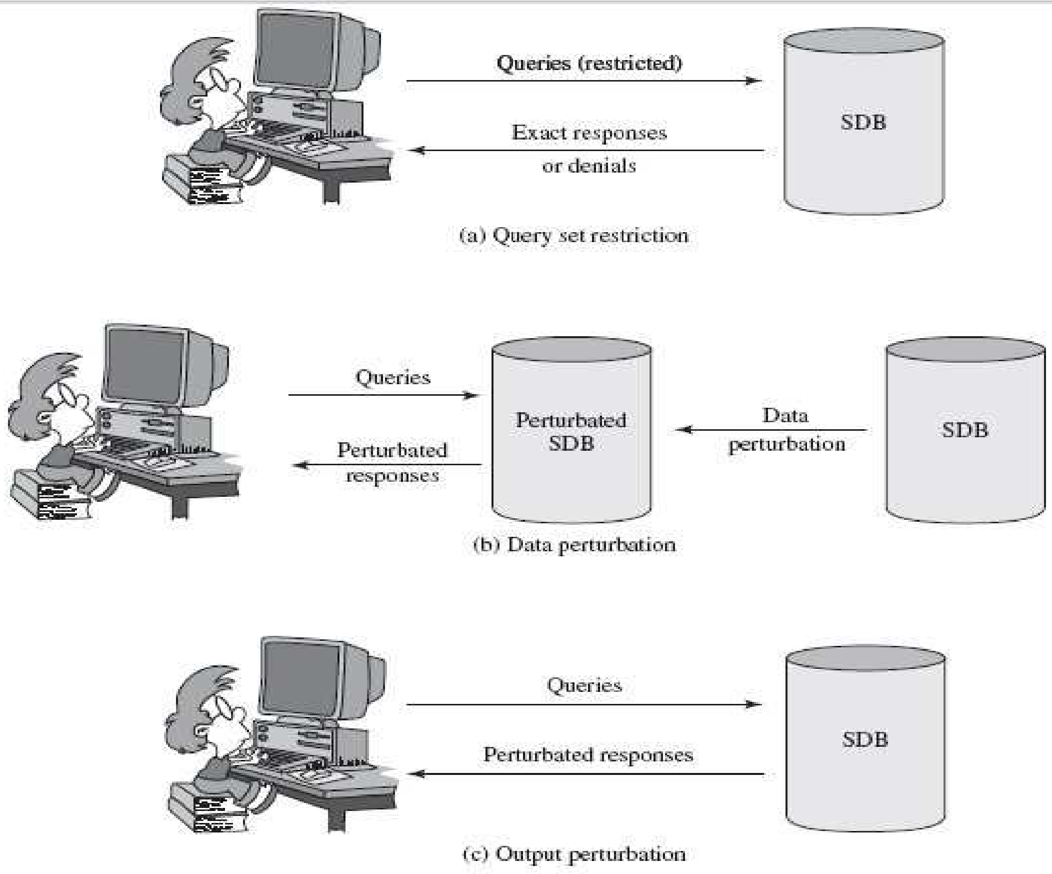


Figure 39: Approaches to Statistical Database Security

- **Query restriction:** Rejects a query that can lead to a compromise. The answers provided are accurate.
- **Perturbation:** Provides answers to all queries, but the answers are approximate.

We examine query restriction in this section and perturbation in the next. Query restriction techniques defend against inference by restricting statistical queries so that they do not reveal user confidential information. Restriction in this context simply means that some queries are denied.

Query Size Restriction

The simplest form of query restriction is query size restriction. For a database of size *N* (number of rows, or records), a query *q*(*C*) is permitted only if the number of records that match *C* satisfies

$$k \leq |X(C)| \leq N - k$$

where *k* is a fixed integer greater than 1. Thus, the user may not access any query set of less than *k* records. Note that the upper bound is also needed. Designate *All* as the set of all records in the database. If *q*(*C*) is disallowed because |X(*C*)| *k*, and there is no upper bound, then a user can compute *q*(*C*) = *q*(*All*) − *q*(˜*C*). The upper bound of *N* - *k* guarantees that the user does not have access to statistics on query sets of less than *k* records. In practice, queries of the form *q*(*All*) are allowed, enabling users to easily access statistics calculated on the entire database.

Query size restriction counters attacks based on very small query sets. For example, suppose a user knows that a certain individual *I* satisfies a given characteristic formula *C* (e.g., Allen is a female CS major). If the query **count**(*C*) returns 1, then the user has uniquely identified *I*. Then the user can test whether *I* has a particular characteristic *D* with the query **count**(*C*· *D*). Similarly, the user can learn the value of a numerical attribute *A* for *I* with the query **sum**(*C*, *A*).

Although query size restriction can prevent trivial attacks, it is vulnerable to more sophisticated attacks, such as the use of a tracker. In essence, the questioner divides his or her knowledge of an individual into parts, such that queries can be made based on the parts without violating the query size restriction. The combination of parts is called a *tracker*, because it can be used to track down characteristics of an individual. We can describe a tracker in general terms using the case from the preceding paragraph. The formula *C·D* corresponds to zero or one record, so that the query **count**(*C·&*) is not permitted. But suppose that the formula *C* can be decomposed into two parts *C* = *C1·C2*, such that the query sets for both *C1* and *T* = (*C1·˜C2*) satisfy the query size restriction. **Figure 5.9** illustrates this situation; in the figure, the size of the circle corresponds to the number of records in the query set. If it is not known if *I* is uniquely identified by C, the following formula can be used to determine if **count**(*C*) = 1:

count(C)=count(C1)−count(T)(5.2)

That is, you count the number of records in *C1* and then subtract the number of records that are in *C1* but not in *C2*. The result is the number of records that are in both *C1* and *C2*, which is equal to the number of records in *C*. By a similar reasoning, it can be shown that we can determine whether *I* has attribute D with

count(C · D) = count(T +; C1 · D)−count(T)

For example, in Table 5.3, Evans is identified by *C* = Male·Bio·1979. Let*k*= 3 in Equation 501. We can use *T* = (*C1*· ˜*C2*) = Male· ˜ (Bio· 1979). Both*C1* and *C2* satisfy the query size restriction. Using Equations (5.2) and (5.3), we determine that Evans is uniquely identified by C and whether his SAT score is at least 600:

$$\begin{aligned}
\textbf{count}(\text{Male} \cdot \text{Bio} \cdot 1979) &= \textbf{count}(\text{Male}) - \textbf{count}(\text{Male} \cdot \sim (\text{Bio} \cdot 1979)) \\
&= 7 - 6 = 1
\end{aligned}$$

$$\begin{aligned}
&\textbf{count}((\text{Male} \cdot \text{Bio} \cdot 1979) \cdot (\text{SAT} \geq 600)) = \\
&\quad \textbf{count}((\text{Male} \cdot \sim (\text{Bio} \cdot 1979) + (\text{Male} \cdot (\text{SAT} \geq 600))) \\
&\quad\quad - \textbf{count}(\text{Male} \cdot \sim (\text{Bio} \cdot 1979)) = 6 - 6 = 0
\end{aligned}$$

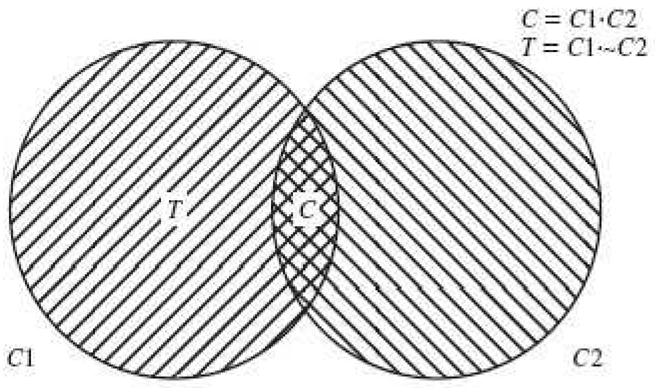


Figure 40: Example of Tracker

In a large database, the use of just a few queries will typically be inadequate to compromise the database. However, it can be shown that more sophisticated tracker attacks may succeed even against large databases in which the threshold *k* is set at a relatively high level [DENN79].

We have looked at query size restriction in some detail because it is easy to grasp both the mechanism and its vulnerabilities. A number of other query restriction approaches have been studied, all of which have their own vulnerabilities. However, several of these techniques in combination do reduce vulnerability.

Query Set Overlap Control

A query size restriction is defeated by issuing queries in which there is considerable overlap in the query sets. For example, in one of the preceding examples the query sets Male and Male· ˜ (Bio· 1979) overlap significantly, allowing an inference. To counter this, the query set overlap control provides the following limitation.

A query *q*(*C*) is permitted only if the number of records that match *C* satisfies

$$|X(C) \cap X(D)| \leq r$$

for all *q*(*D*) that have been answered for this user, and where *r* is a fixed integer greater than 0.

This technique has a number of problems, including the following [ADAM89]:

1. This control mechanism is ineffective for preventing the cooperation of several users to compromise the database.
2. Statistics for both a set and its subset (e.g., all patients and all patients undergoing a given treatment) cannot be released, thus limiting the usefulness of the database.
3. For each user, a user profile has to be kept up to date.

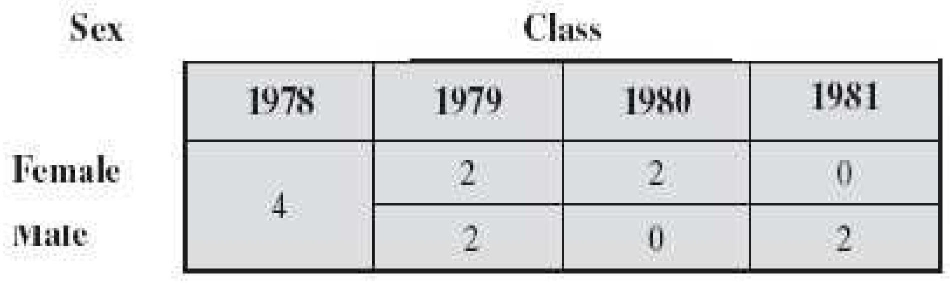
3.3 Partitioning

Partitioning can be viewed as taking query set overlap control to its logical extreme, by not allowing overlapping queries at all. With partitioning, the records in the database are clustered into a number of mutually exclusive groups. The user may only query the statistical properties of each group as a whole. That is, the user may not select a subset of a group. Thus, with multiple queries, there must either be complete overlap (two different queries of all the records in a group) or zero overlap (two queries from different groups).

The rules for partitioning the database are as follows:

1. Each group G has $g = |G|$ records, where $g = 0$ or $g \geq n$, and g even, where n is a fixed integer parameter.
2. Records are added or deleted from G in pairs.
3. Query sets must include entire groups. A query set may be a single group or multiple groups.

A group of a single record is forbidden, for obvious reasons. The insertion or deletion of a single record enables a user to gain information about that record by taking before and after statistics. As an example, the database of Table 5.3a can be partitioned as shown in Table 5.5. Because the database has an odd number of records, the record for Kline has been omitted. The database is partitioned by year and sex, except that for 1978, it is necessary to merge the Female and Male records to satisfy the design requirement.



Sex	Class			
	1978	1979	1980	1981
Female	4	2	2	0
Male		2	0	2

Table 8 Partitioned Database

Partitioning solves some security problems but has some drawbacks. The user's ability to extract useful statistics is reduced, and there is a design effort in constructing and maintaining the partitions.

3.4 Query Denial and Information Leakage

A general problem with query restriction techniques is that the denial of a query may provide sufficient clues that an attacker can deduce underlying information. This is generally described by saying that query denial can leak information.

Here is a simple example from [KENT05]. Suppose that the underlying database consists of real-valued entries and that a query is denied only if it would enable the requestor to deduce a value. Now suppose the requester poses the query **sum**(*x*1, *x*2, *x*3) and the response is 15. Then the requester queries **max**(*x*1, *x*2, *x*3) and the query is denied. What can the requester deduce from this? We know that the **max**(*x*1, *x*2, *x*3) cannot be less than 5 because then the sum would be less than 15. But if **max**(*x*1, *x*2, *x*3) > 5, the query would not be denied because the answer would not reveal a specific value. Therefore, it must be the case that **max**($x1$, $x2$, $x3$) = 5, which enables the requester to deduce that $x1 = x2 = x3 = 5$.

[KENT05] describes an approach to counter this threat, referred to as ***simulatable auditing***. The details of this approach are beyond the scope of this chapter. In essence, the system monitors all of the queries from a given source and decides on the basis of the queries so far posed whether to deny a new query. The decision is based solely on the history of queries and answers and the specific new query. In deciding whether to deny the query, the system does not consider the actual values of database elements that will contribute to generating the answer and therefore does not consider the actual value of the answer.

Thus, the system makes the denial decision solely on the basis of information that is already available to the requester (the history of prior requests). Hence the decision to deny a query cannot leak any information. For this approach, the system determines whether any collection of database values might lead to information leakage and denies the query if leakage is possible. In practice, a number of queries will be denied even if leakage is not possible. In the example of the preceding paragraph, this strategy would deny the **max** query whether or not the three underlying values were equal. Thus, this approach is more conservative in that it issues more denials than an approach that considers the actual values in the database.

3.5 Perturbation

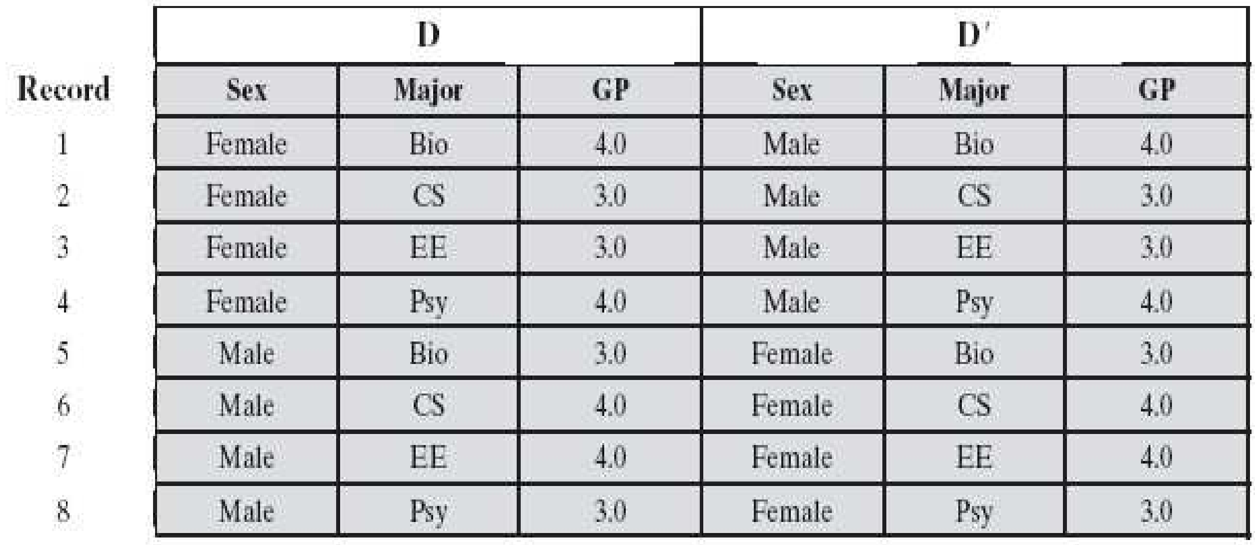
Query restriction techniques can be costly and are difficult to implement in such a way as to completely thwart inference attacks, especially if a user has supplementary knowledge. For larger databases, a simpler and more effective technique is to, in effect, add noise to the statistics generated from the original data. This can be done in one of two ways (Figure 5.8): The data in the SDB can be modified (perturbed) so as to produce statistics that cannot be used to infer values for individual records; we refer to this as **data perturbation**. Alternatively, when a statistical query is made, the system can generate statistics that are modified from those that the original database would provide, again thwarting attempts to gain knowledge of individual records; this is referred to as **output perturbation**.

Regardless of the specific perturbation technique, the designer must attempt to produce statistics that accurately reflect the underlying database. Because of the perturbation, there will be differences between perturbed results and ordinary results from the database. However, the goal is to minimize the differences and to provide users with consistent results. As with query restriction, there are a number of perturbation techniques. In this section, we highlight a few of these.

Data Perturbation Techniques

We look at two techniques that consider the SDB to be a sample from a given population that has a given population distribution. Two methods fit into this category. The first transforms the database by substituting values that conform to the same assumed underlying probability distribution. The second method is, in effect, to generate statistics from the assumed underlying probability distribution.

The first method is referred to as **data swapping**. In this method, attribute values are exchanged (swapped) between records in sufficient quantity so that nothing can be deduced from the disclosure of individual records. The swapping is done in such a way that the accuracy of at least low-order statistics is preserved. Table 5.6, from [DENN82], shows a simple example, transforming the database D into the database D. The transformed database D has the same statistics as D for statistics derived from one or two attributes. However, three-attribute statistics are not preserved. For example, **count**(Female· CS· 3.0) has the value 1 in D but the value 0 in D.



	D			D′		
Record	Sex	Major	GP	Sex	Major	GP
1	Female	Bio	4.0	Male	Bio	4.0
2	Female	CS	3.0	Male	CS	3.0
3	Female	EE	3.0	Male	EE	3.0
4	Female	Psy	4.0	Male	Psy	4.0
5	Male	Bio	3.0	Female	Bio	3.0
6	Male	CS	4.0	Female	CS	4.0
7	Male	EE	4.0	Female	EE	4.0
8	Male	Psy	3.0	Female	Psy	3.0

Table 9 Example of Data Swapping

Another method is to generate a modified database using the estimated underlying probability distribution of attribute values. The following steps are used:

1. For each confidential or sensitive attribute, determine the probability distribution function that best matches the data and estimate the parameters of the distribution function.
2. Generate a sample series of data from the estimated density function for each sensitive attribute.
3. Substitute the generated data of the confidential attribute for the original data in the same rank order. That is, the smallest value of the new sample should replace the smallest value in the original data, and so on.

Output Perturbation Techniques

A simple output perturbation technique is known as **random-sample query**. This technique is suitable for large databases and is similar to a technique employed by the U.S. Census Bureau. The technique works as follows:

1. A user issues a query $q(C)$ that is to return a statistical value. The query set so defined is $X(C)$.
2. The system replaces $X(C)$ with a sampled query set, which is a properly selected subset of $X(C)$.

3. The system calculates the requested statistic on the sampled query set and returns the value.

Other approaches to output perturbation involve calculating the statistic on the requested query set and then adjusting the answer up or down by a given amount in some systematic or randomized fashion. All of these techniques are designed to thwart tracker attacks and other attacks that can be made against query restriction techniques.

With all of the perturbation techniques, there is a potential loss of accuracy as well as the potential for a systematic bias in the results.

Limitations of Perturbation Techniques

The main challenge in the use of perturbation techniques is to determine the average size of the error to be used. If there is too little error, a user can infer close approximations to protected values. If the error is, on average, too great, the resulting statistics may be unusable. For a small database, it is difficult to add sufficient perturbation to hide data without badly distorting the results. Fortunately, as the size of the database grows, the effectiveness of perturbation techniques increases.

The last-mentioned reference reported the following result. Assume the size of the database, in terms of the number of data items or records, is *n*. If the number of queries from a given source is linear to the size of the database (i.e., on the order of *n*), then a substantial amount of noise must be added to the system in terms of perturbation, to preserve confidentiality. Specifically, suppose the perturbation is imposed on the system by adding a random amount of perturbation ≤x . Then, if the query magnitude is linear, the perturbation must be at least of order √n . This amount of noise may be sufficient to make the database effectively unusable. However, if the number of queries is sublinear (e.g., of order √n), then much less noise must be added to the system to maintain privacy. For a large database, limiting queries to a sublinear number may be reasonable.

4.0 Conclusion

As we all know that Computer security and Statistical database management system is the life wire of any existing organization. With the introduction to basic concepts Statistics database system and Computer security, students must have acquired the basic sense of the central importance of statistical databases in today's information systems environment. And with this, they should be able to design and model various Statistical database system to safeguard crackers in real life situations.

5.0 Summary

In this unit we have learnt that:

- A general problem with query restriction techniques is that the denial of a query may provide sufficient clues that an attacker can deduce underlying information.
- The main challenge in the use of perturbation techniques is to determine the average size of the error to be used. If there is too little

error, a user can infer close approximations to protected values. Fortunately, as the size of the database grows the effectiveness of perturbation techniques increases.

- Partitioning can be viewed as taking query set overlap control to its logical extreme, by not allowing overlapping queries at all.
- A statistical user of an underlying database of individual records is restricted to obtaining only aggregate, or statistical, data from the database and is prohibited access to individual records

6.0 Tutor Marked Assignment

1. (a) Differentiate between the following:

   (i) Query Denial and Information leakage (ii) Statistical Database and Query restriction management.

   (b) What did you understand by the term 'Partitioning'?

2. (a) What are the characteristics of Perturbation approach?

   (b) Mention the advantages of using Perturbation technique over other methods.

7.0 Further Reading and Other Resources

Dorothy E. Denning, Secure statistical databases with random sample queries, ACM Transactions on Database Systems (TODS), Volume 5, Issue 3 (September 1980), Pages: 291 – 315.

Wiebren de Jonge, Compromising statistical databases responding to queries about means, ACM Transactions on Database Systems, Volume 8, Issue 1 (March 1983), Pages: 60 – 80.

Dorothy E. Denning, Jan Schlörer, A fast procedure for finding a tracker in a statistical database, ACM Transactions on Database Systems, Volume 5, Issue 1 (March 1980) . Pages: 88 – 102.

CCSP Self-Study: Advanced AAA Security for Cisco Router Networks By John Roland

Unwitting Collaborators: Series Introduction By Frank Fiore, Jean Francois

Intrusion Detection Systems by Earl Carter

Module 3 Application of Statistical Database System

Unit 1 SPEA SMART Airport Statistical Data Management System (SMART STAT)

1.0 Introduction

2.0 Objective

3.0 Uses of the Airport Statistical Data Management System

3.1 Benefits of the Airport Statistical Data Management System

3.2 Outline of the Airport Statistical Data Management System

3.3 Features and Functionalities of the STAT

4.0 Conclusion
5.0 Summary
6.0 Tutor Marked Assignment
7.0 Further Reading and Other Resources

1.0 Introduction

The **SPEA Smart Airport Statistical Data Management System (SMART STAT)** is an intelligent system that transforms raw flight and Airport operational data into management information. STAT performs statistical analysis on flight and Airport operational data and makes data available to other SMART systems.

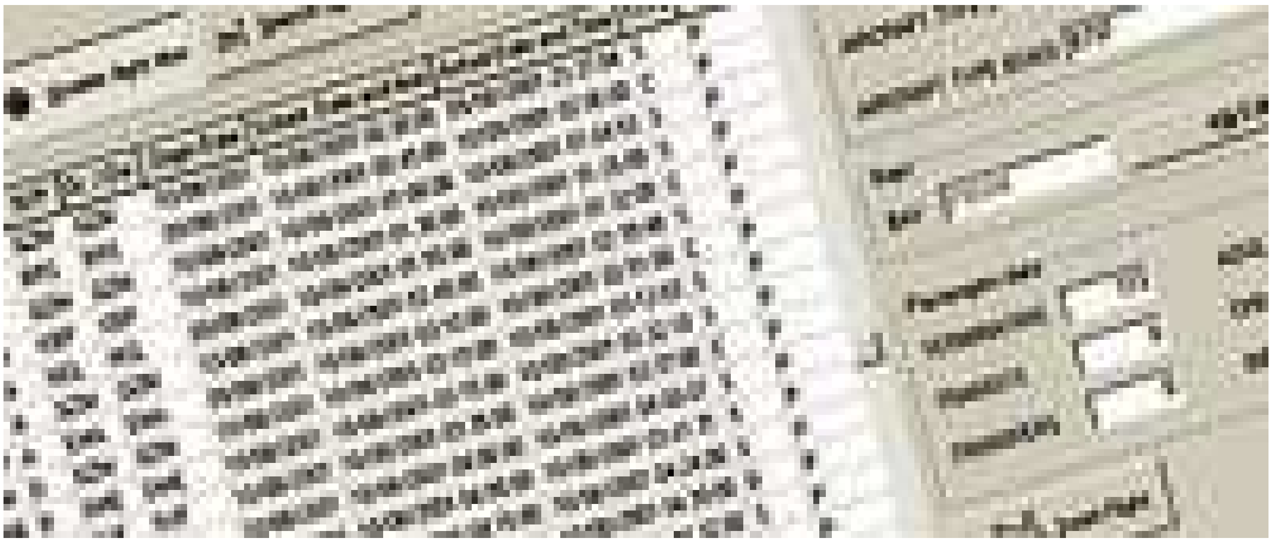


Figure 41: Diagram of SPEA Smart Airport Statistical Data Management System

2.0 Objective

At the end of this unit, you should be able to:

- ❖ Appreciate the uses of the Airport Statistical Data Management System.
- ❖ State the major concepts in Airport Statistical Database System.
- ❖ Draw the major components of Airport Statistical Management Diagrams, Naming Conventions and Design Issues in data management system
- ❖ Explain what Airport Statistical database model is all about
- ❖ Mention benefits of Airport Statistical Management System.

3.0 Uses of the Airport Statistical Data Management System

- Evaluate the current status of ground handling operations.
- Optimize the deployment of existing Airport resources.
- Determine future need for additional Airport resources.
- Consolidate data for billing.

3.1 Benefits of the Airport Statistical Data Management System

- Improved tactical and strategic planning.
- Improved management decision making.

- Reduced operating cost and capital required for Airport expansion.
- Availability of concise structured information from customized reports.
- Flexibility in accessing data using standard query languages.
- Accurate and precise information for billing.
- Authorized users can access information via workstations or the Internet.

3.2 Outline of the Airport Statistical Data Management System

The SPEA Smart Airport Statistical Data Management System (SMART STAT) is an intelligent system that transforms raw flight and Airport operational data into management information. The ability to manipulate and perform statistical analysis on historical data from any source is a powerful tool for Airport Management. It allows them to evaluate the current status of ground handling operations and to make informed decisions on current and future operations and resource requirements.

Simple selections allow the user to run queries and also create useful customized reports on data within the SMART system and other Airport databases. SMART STAT allows users to access the database via Standard Query Language (SQL). Microsoft .Net, Open Database Connectivity (ODBC), Java Database Connectivity (JDBC) and other.

The following flight data is available for statistical analysis:

- Flight information - flight number, flight date and airline.
- Complete flight routing - scheduled time of departure and arrival for all Airports, including multiple city/leg flights.
- Flight qualification - flight type, traffic type, loading type and service type.
  •Aircraft information - aircraft type, aircraft registration number and aircraft configuration.
- Passenger information - terminating/transfer/transit passengers, passengers per weight category, passengers per class and special requirements.
- Baggage information – number and type bags, mishandling and irregularities.
  •Load information - mail, cargo, crew and passengers.
- Times - scheduled times, estimated times, touch-down/take-off times, block on/off times, red times, zone-in times and boarding/last call/flight closed times.
- Resources allocated to the flight - check-in counters, boarding gate, baggage reclaim belt and parking stand.
- Punctuality and regularity - flight delays and delay codes.

SMART STAT integrates fully with the Airport Operational DataBase System (SMART AODB) and other SMART Systems and interfaces with other Airport databases. Data is immediately available for analysis as it is recorded in any database. The Airport Billing System (SMART BILL) extracts data from SMART STAT for accurate billing. Statistical data can also be exported to any other billing software operated at the Airpor

3.3 Features and Functionalities of the STAT

- High performing and secure Oracle 10g database.
- Access the database via SQL, Microsoft . Net, ODBC, JDBC and other.
- User-friendly and highly configurable Windows browser based Graphical User Interface (GUI).
- Comply with International Air Transport Association (IATA) standards.
- Full seamless integration with all SMART Systems.
- Robust, reliable and scalable system.
- Supports concurrent users and is accessible from any authorized workstation.
- Data can be automatically backed-up on different media types.

❖ 4.0 Conclusion

With the overview of the Airport database management system, individuals and organizations can uncover hidden processes, methodologies, as well as the benefits of managing their data, which they can use to predict the behaviour of customers, products and processes.

5.0 Summary

In this unit we have learnt that:

- ❖ How to use Statistical database system in an Airport.
- ❖ How to design a model for an Airport Statistical Database Management system, which is a collection of programs that manage the database structure and that control shared access to the data in the database.
- ❖ The various benefits that can be derived in using this system, like controlling redundancy, restricting unauthorized access, saving time etc, with flexibility, economies of scale, and potential for enforcing standards, are some of the implications of database approach.

6.0 Tutor Marked Assignment

1. (a) What are the benefits of Airport database management system?

2. (a) What are the characteristics of Airport database approach?

   (b) Mention the advantages of using Airport database management approach over other methods

7.0 Further Reading and Other Resources

SPEA SMART Airport Passenger and Baggage Management Systems

SPEA SMART Airport Resource Utilization Systems

SPEA SMART Airport Technology Support Systems

SPEA SMART Airport Data Management Systems

Statistical Data Management System

Semantic Message Processor System

Airport Operational DataBase System

Airport Departure Control System

Airport Operational DataBase System

Airport Resource Management System

Airport Stand Allocation System

Airport Baggage Reconciliation System