Chapter 1: Databases and Database Users

CS-6360 Database Systems

Chris Irwin Davis, Ph.D.

Email: chrisirwindavis@utdallas.edu
Phone: (972) 883-3574
Office: ECSS 4.705

- Textbook
- eLearning
  - Homework
  - Programming Assignments
  - Reading Assignments
- Syllabus

Chapter Outline

- 1.1 – Introduction
- 1.2 – An Example
- 1.3 – Characteristics of the Database Approach
- 1.4 – Actors on the Scene
- 1.5 – Workers behind the Scene
- 1.6 – Advantages of Using the DBMS Approach
- 1.7 – A Brief History of Database Applications
- 1.8 – When Not to Use a DBMS

Database Theory

- Theoretical Domains
  - Set Theory
  - Information Theory
  - Relational Model
  - Relational Algebra
  - Relational Calculus

Database Theory

- Implementation Paradigms
  - SQL
  - NoSQL
  - DOM (HTML, XHTML, XML)
    - JDOM
  - SAX (Simple API for XML)
  - Semantic Web (SPARQL, RDF/OWL)
  - OODB

1.1 Introduction

What is a Database?

- Database
  - Collection of related data
  - Known facts that can be recorded and that have implicit meaning
  - Miniworld or Universe of Discourse (UoD)
  - Represents some aspect of the real world
  - Logically coherent collection of data with inherent meaning (semantics)
  - Built for a specific purpose

Examples

- **Traditional database applications**
  - Store textual and/or numeric information, such as the Medicare database.
- **Multimedia databases**
  - Store images, audio clips, and video streams digitally. Most modern DBMSs can do this
- **Geographic information systems (GIS)**
  - Store and analyze maps, weather data, and satellite images

Examples

- **eCommerce**
  - eBay, Amazon, etc.
- **Data warehouses and online analytical processing (OLAP) systems**
  - Extract and analyze useful business information from very large databases
  - Support decision making
- **Real-time and active database technology**
  - Control industrial and manufacturing processes

Database Management System (DBMS)

- A *DBMS* is a suite of programs that allows a user to create and maintain a database.
- You can specify
  - the *types of data*,
  - *relationships* between various data elements, and
  - *constraints* on what can be stored.
- *Metadata* describes the data, and is stored in the database.
  - Also called a data dictionary.
- A *schema* is the structure, or framework, of the database

Database Management System (cont'd)

- DBMSs allow you to manipulate the database, both the schema and the actual data.
- You can query the database to retrieve information based upon criteria.
- You can update the database, which includes adding new information, changing existing information, and removing records.

Database Management System (cont'd)

- DBMSs are generally multi-user; they support requests from large numbers of users
- DBMSs generally provide security, both in terms of encrypting sensitive data and limiting access to authorized users.
- DBMSs provide a programming interface (API) to allow you to write programs to interact with the database.

Practical Stuff: Available DBs

- **MySQL** from Oracle
  - Free (upgrade path to Oracle Server)
  - You will be using for your project
  - Most lecture examples
- **Microsoft SQL Server**
  - The professional version is free to you as a UTD student, while the express edition is free to everyone.
- **PostgreSQL**
- **SQLite**

Terminology

Relational Algebra	Database
Relation	Table
Field	Column
Tuple	Row

Data Types

- Each column (field, attribute) is of a specific data type.
- Data types can include:
  - Integers
  - Fixed-length strings
  - Variable-length strings
  - Floating-point numbers
  - Decimals
  - Dates, Times
  - BLOBs (binary large objects).

1.2 An Example

An Example

- UNIVERSITY database
  - Information concerning students, courses, and grades in a university environment
- Data records (tables)
  - STUDENT
  - COURSE
  - SECTION
  - GRADE_REPORT
  - PREREQUISITE

An Example (cont'd)

- Construct UNIVERSITY database
  - Store data to represent each student, course, section, grade report, and prerequisite as a record in appropriate file
- Relationships among the records
- Manipulation involves querying and updating

An Example (cont’d)

- A good question to ask when designing a database is “What will you want to know from it?”
- For example:
  - Retrieve the transcript
  - List the names of students who took the section of the ‘Database’ course offered in fall 2013 and their grades in that section
  - List the prerequisites of the ‘Database’ course

An Example (cont’d)

- Examples of updates:
  - Change the class of ‘Smith’ to sophomore
  - Create a new section for the ‘Database’ course for this semester
  - Enter a grade of ‘A’ for ‘Smith’ in the ‘Database’ section of last semester

University Database

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

Phases for Designing a Database

- Requirements specification and analysis
  - Does this sound like software engineering?
- Conceptual design
  - ER Model (Chapter 3)
  - EER Model (Chapter 4)
- Logical design
  - Relational Model (Chapter 5)
- Physical design

1.3 Characteristics of the Database Approach

The Traditional File Approach

- In traditional file processing…
  - Each user defines and implements the files needed for a specific software application
    - This is not strictly true; usually we designed suites of applications that used common files.
    - However, the layout of the files was defined in the code, not in the file. (e.g. preferences files, configuration files)

The Database Approach

- Single repository maintains data that is defined once and then accessed by various users, and the database contains the layout.
- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

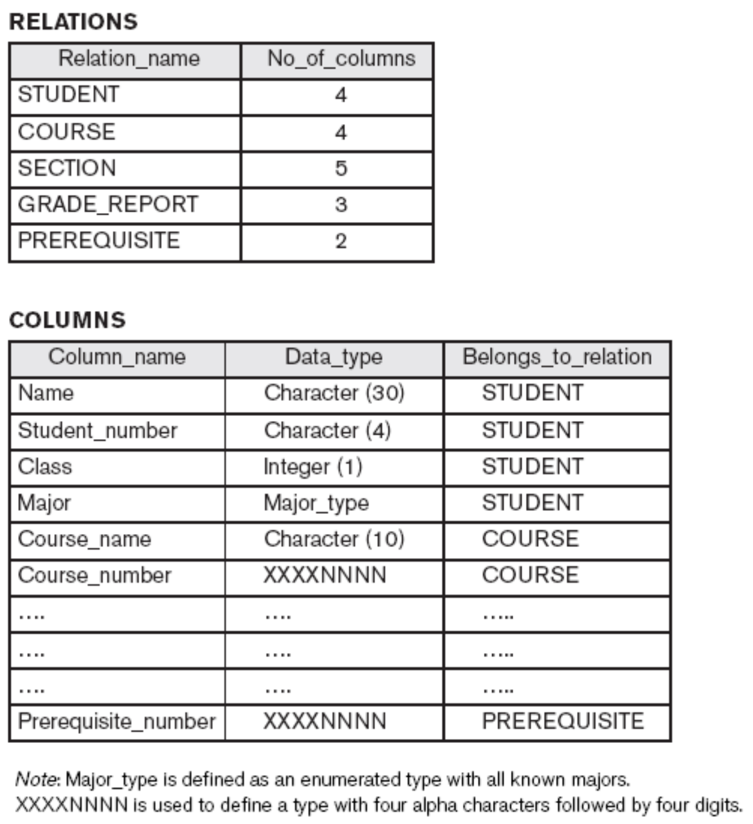
Self-Describing Database

- Database system contains complete definition of structure and constraints

- Meta-data
    - "Database Catalog" – Describes structure of the database

- Database catalog used by:
    - DBMS software
    - Database users who need information about database structure

Database Catalog (i.e. Meta-data) Example



RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerated type with all known majors.
XXXXNNNN is used to define a type with four alpha characters followed by four digits.

Figure 1.3
An example of a database catalog for the database in Figure 1.2.

Insulation Between Programs and Data

- Program-data independence
  - Structure of data files is stored in DBMS catalog separately from access programs

- Program-operation independence
  - Operations specified in two parts:
    - Interface includes operation name and data types of its arguments
    - Implementation can be changed without affecting the interface

Support of Multiple Views of the Data

- View
  - Subset of the database
  - Contains virtual data derived from the database files but is not explicitly stored
- Multiuser DBMS
  - Users have a variety of distinct applications
  - Must provide facilities for defining multiple views

Sharing of Data and Multiuser Transaction

- Allow multiple users to access the database at the same time
- Concurrency control software
  - Ensure that several users trying to update the same data do so in a controlled manner
  - Result of the updates is correct
- **Online transaction processing (OLTP) application (e.g. filling in forms)**

Sharing of Data and Multiuser Transaction Processing

- Transaction
  - Central to many database applications
  - Is an executing program or process that includes one or more database accesses (read, update, etc.)
  - Two properties…
- Isolation property
  - Ensures that each transaction appears to execute in isolation from other transactions
- Atomicity property
  - Ensures that either all the database operations in a transaction are executed or none are

1.4 Actors on the Scene

Actors on the Scene

- Database administrators (DBA) are responsible for:
  - Authorizing access to the database
  - Coordinating and monitoring its use
  - Acquiring software and hardware resources

- Database designers are responsible for:
  - Identifying the data to be stored
  - Choosing appropriate structures to represent and store this data

Actors on the Scene (cont'd.)

- End users
  - People whose jobs require access to the database
  - Types
    - Casual end users
    - Naive or parametric end users
    - Sophisticated end users
    - Standalone users

Actors on the Scene (cont'd.)

- System analysts
  - Determine requirements of end users, especially naive and parametric end users, and develop specifications for standard canned transactions that meet these requirements.
- Application programmers
  - Implement these specifications as programs, then they test, debug, document, and maintain these canned transactions.

1.5 Workers behind the Scenes

Workers behind the Scenes

- DBMS system designers and implementers
  - Design and implement the DBMS modules and interfaces as a software package
  - The DBMS must interface with other system software such as the operating system and compilers for various programming languages.
- Tool developers
  - Design and implement tools
- Operators and maintenance personnel
  - Responsible for running and maintenance of hardware and software environment for database system

1.6 Advantages of Using the DBMS Approach

Advantages of Using the DBMS Approach

- Controlling redundancy
  - Data normalization
  - **Denormalization**
    - Sometimes necessary to use controlled redundancy to improve the performance of queries
- Restricting unauthorized access
  - Security and authorization subsystem
  - Privileged software

Advantages of Using the DBMS Approach

- Providing persistent storage for program objects
  - Complex object in C++ can be stored permanently in an object-oriented DBMS
  - **Impedance mismatch problem**
    - Object-oriented database systems typically offer data structure compatibility

Advantages of Using the DBMS Approach

- Providing backup and recovery
  - Backup and recovery subsystem of the DBMS is responsible for recovery
- Providing multiple user interfaces
  - **Graphical user interfaces (GUIs)**
- Representing complex relationships among data
  - May include numerous varieties of data that are interrelated in many ways

Advantages of Using the DBMS Approach

- Enforcing integrity constraints
  - **Referential integrity constraint**
    - Every section record must be related to a course record
  - **Key or uniqueness constraint**
    - Every course record must have a unique value for Course_number
  - **Business rules**
  - Inherent rules of the data model

Advantages of Using the DBMS Approach

- Permitting inferencing and actions using rules
  - **Deductive database systems**
    - Provide capabilities for defining deduction rules
    - Inferring new information from the stored database facts
  - **Triggers**
    - Rule activated by updates to the table
  - **Stored procedures**
    - More involved procedures to enforce rules

Advantages of Using the DBMS Approach

- Additional implications of using the database approach
  - Reduced application development time
  - Flexibility
  - Availability of up-to-date information
  - Economies of scale

1.7 A Brief History of Database Applications

History of Database Applications

- Hierarchical and Network Systems
  - Intermixing of conceptual relationships with the physical storage and placement of records on disk
- Relational Databases
  - Data Abstraction and Application Flexibility
  - Separate physical storage from conceptualization
- Object-Oriented Applications
- Interchanging Data on the Web for E-Commerce Using XML

History of Database Applications

- Extending DB Capabilities for New Applications
  - Scientific applications
  - Storage and retrieval of images
  - Videos
  - Data mining
  - Spatial applications
  - Time series applications
- Databases versus Information Retrieval

1.8 When Not to Use a DBMS

When Not to Use a DBMS

- Sometimes more desirable to use regular files for:
  - Simple, well-defined database applications not expected to change at all
  - Stringent, real-time requirements that may not be met because of DBMS overhead
  - Embedded systems with limited storage capacity
  - No multiple-user access to data