ECS 165A: Introduction to Database Systems

Todd J. Green

based on material and slides by
Michael Gertz and Bertram Ludäscher

Winter 2011

1. Introduction to Relational Databases

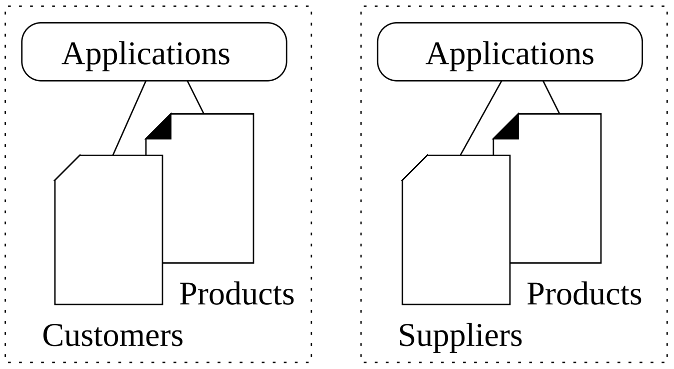
What is a Database System?

- In this class we will learn about *Databases* (DBs) and *Database Management Systems* (DBMSs)
- A Database is a (typically very large) integrated collection of interrelated data which are stored in a persistent format (files). Data describe information and activities about one or more related organizations (portion of the *real world*). For example, a university database might contain information about
  - entities (e.g., students, courses, faculties, . . .)
  - relationships among entities (e.g., 'Bill is taking ECS 165A')
- A Database Management System is a collection of software packages designed to store, access, and manage databases. It provides users and applications with an environment that is convenient, efficient, and reliable.

File System versus DBMS

File processing approach: prominent approach supported by conventional Operating Systems for data intensive application programs in the 60's and 70's (and even sometimes today).

In the file processing approach, each application uses its own files:



Drawbacks:

- Data are stored redundantly in many files using different formats ($\Longrightarrow$ inconsistencies among data, because changes are not coordinated)
- No transaction management and concurrency control to support multiple users
- Difficult to access data (no high-level *query languages*)
- No security mechanisms (OS file permissions are not enough!)
- No recovery mechanisms
- Program–data dependence ($\Longrightarrow$ high maintenance costs)
- Poor data modeling concepts
- . . .

Purpose of a Database System

- *Data Integration* All data are uniformly managed.
- *Efficient Data Access* Database languages are provided to store, access and manage data.
- *Data Dictionary* Contains all data about objects and structures of the database (*metadata*)
- *User/Application-Views* Different views for different users and applications
- *Integrity Constraints* are enforced by the DBMS.
- *Security Mechanisms* to protect data from security threats
- *Transactions* combine sets of operations on data into logical atomic units
- *Synchronization* of concurrent user transactions
- *Recovery* of data after system crash
- *ad-hoc* queries, report generation, interfaces to other database systems, interfaces for application programming, . . .

Trends in Database Systems

Some Database Application Settings

- EOS – NASA's Earth Observation System: Petabyte-sized databases, thousands of customers and applications
- E-Commerce and Financial Applications: Integration of heterogeneous information sources (e.g., catalogs)
- Health-Care Information Systems and Electronic Patient Record: Integration of heterogeneous forms of legacy data
- Digital Libraries & Digital Publishing: Management and delivery of large bodies of textual and multimedia data
- Collaborative Work and Design: Integration of heterogeneous information sources, concurrency control, workflows

Trends that Affect Database Management Systems

- Record-based data has been joined by various kinds of semistructured and multimedia data
- Information Superhighway: Web = Constantly growing collection of heterogeneous information sources
- High bandwidth, more storage capacity, high performance computing . . .

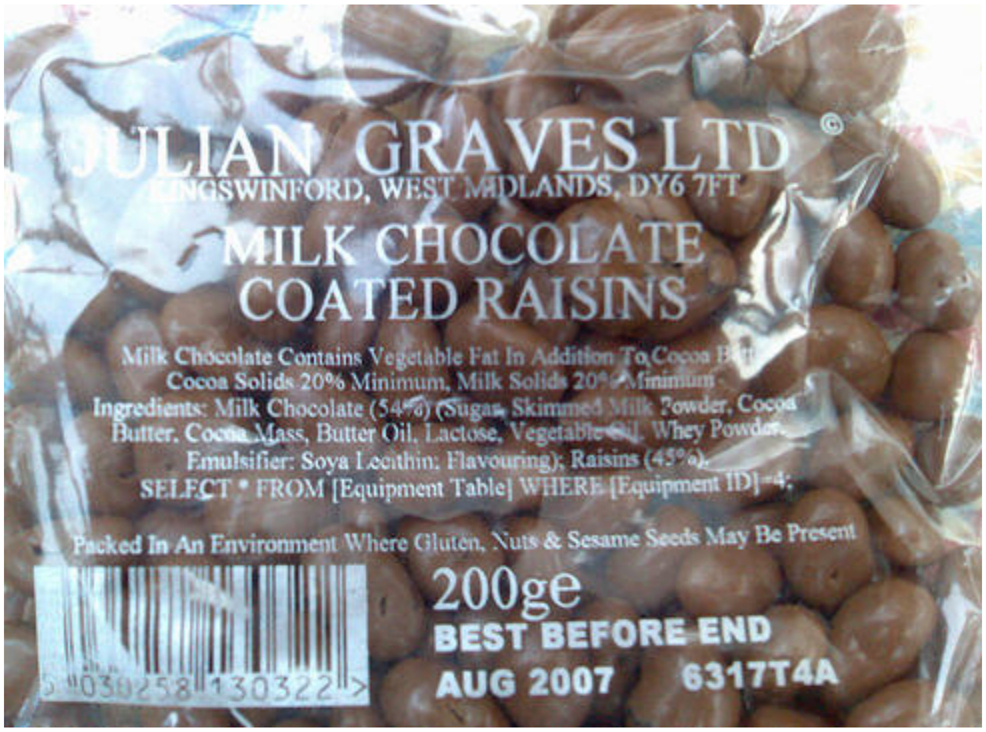
Challenges

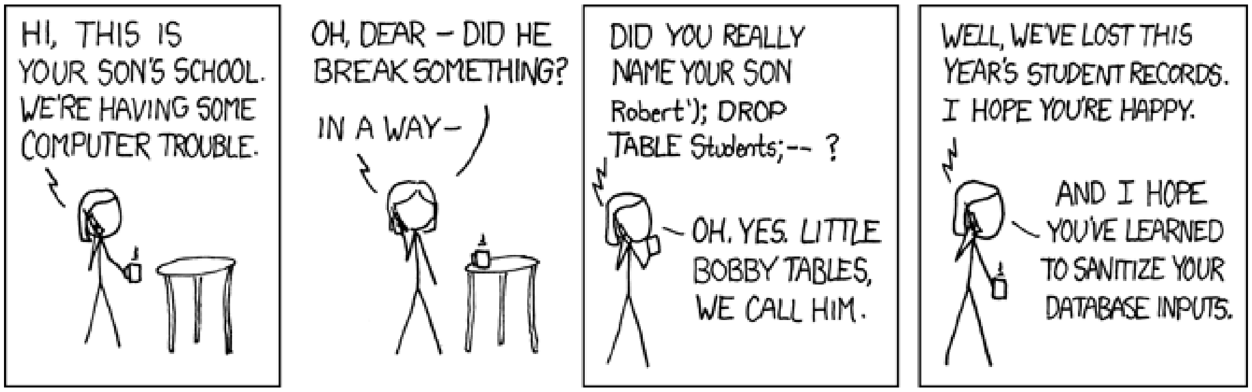
- Data management in sensor networks (e.g., RFID)
- Grid computing
- Integration of multimedia objects and "standard" data
- User interfaces development, information visualization
- Distribution, heterogeneity, complexity of data; security and privacy; data integration and conversion; data quality, . . .

Why Study Database Systems?

- Shift from DBS as a specialized application to central component of modern computing environments ("Data and Information Science")
- Shift from computation to information
  - at the low end, we have a very messy Web space containing hundreds of thousands of information sources
  - at the high end, we have various needs for scientific applications
- Datasets are increasing in diversity, complexity, and volume
  - Digital Libraries, E-, Human Genome Project, . . .
  - The need for (advanced) DBMS is exploding . . . !!!
- Database Systems encompass most of CS topics
  - Operating Systems (file management, process management, . . .)
  - Theory (languages, algorithms, computation, complexity, . . .)
  - Artificial Intelligence (knowledge based systems, intelligent search, . .
  - Software Engineering (application development, GUIs, . . .)
  - Multimedia (video on demand, audio, movies, . . .)
  - Logic (query languages, query optimization, . . .)

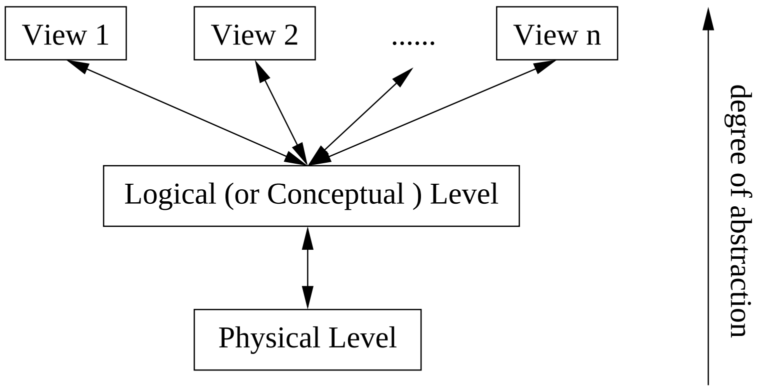
Databases are Ubiquitous


JULIAN GRAVES LTD ©
KINGSWINFORD, WEST MIDLANDS, DY6 7FT
MILK CHOCOLATE COATED RAISINS
Milk Chocolate Contains Vegetable Fat In Addition To Cocoa Butter
Cocoa Solids 20% Minimum, Milk Solids 20% Minimum
Ingredients: Milk Chocolate (54%) (Sugar, Skimmed Milk Powder, Cocoa Butter, Cocoa Mass, Butter Oil, Lactose, Vegetable Oil, Whey Powder, Emulsifier: Soya Lecithin; Flavouring); Raisins (45%),
SELECT * FROM [Equipment Table] WHERE [Equipment ID]=4;
Packed In An Environment Where Gluten, Nuts & Sesame Seeds May Be Present
200ge
BEST BEFORE END
AUG 2007 6317T4A
5 030258 130322


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Different Views of Data

A major purpose of a DBMS is to provide users with an abstract view of data, i.e. it hides details of how data are stored and maintained on a computer.



Levels of Abstraction in a DBMS

- **Physical Level** describes *how* data records are actually stored and how files and indexes are organized and used
- **Logical Level** (sometimes also called Conceptual Level) describes *what* data are stored in the DBS in terms of entities and relationships; emphasis on logical structure of the database
- **View Level** describes how users and applications see the data

$\rightsquigarrow$ Abstraction is achieved by describing each level in terms of a database schema, which, in turn, is based on a data model

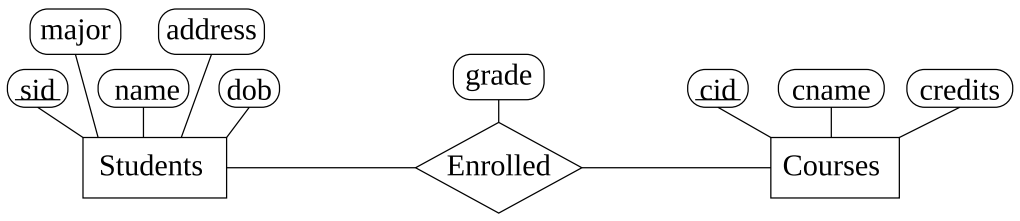
Data Models, Schemas, and Instances

- A *Data Model* is a collection of concepts for describing
  - data and relationships among data
  - data semantics and data constraints
- Object-Based logical Models
  - Entity-Relationship (ER) Model
  - Object-Oriented (OO) Model
- Record-Based logical Models
  - Relational Model
  - Network Model
  - Hierarchical Model
- A *database schema* is a description of a particular collection of data, using a given data model.

  An *instance* of a database schema is the actual content of the database at a particular point in time.
- Schemas exist at different levels of abstraction
  - Conceptual (or Logical) Schema: typically builds the basis for designing a database (main focus in this course)
  - View (or External) Schemas: typically determined during requirements analysis (often require integration into one conceptual schema)
  - Physical Schema: storage structures associated with relations

Example: University Database

- Conceptual Schema: can be based on Entity-Relationship, Object-Oriented, or Relational Model.
  Example of an Entity-Relationship schema:



- Relational schema:
  Students(sid : integer, name : string, major : string,
  address : string, dob : date)
  Courses(cid : integer, cname : string, credits : integer)
  Enrolled(sid : integer, cid : integer, grade : string)

- Physical Schema:
  - Relations are stored as unordered files
  - Index on first column of Students

- External Schemas (Views):
  - Course_Info(cid : integer, enrollment : integer)

Data Independence

- Ability to modify definition of schema at one level without affecting a schema definition at a higher level
- Achieved through the use of three levels of data abstraction (also called *three level schema architecture*)
  - *Logical Data Independence*: Ability to modify logical schema without causing application programs to be rewritten
  - *Physical Data Independence*: Ability to modify physical schema without causing logical schema or applications to be rewritten (occasionally necessary to improve performance)

Database Languages

A Database Management System offers two different types of languages

- Data Definition Language (DDL)
  - Specification language (notation) for defining a database schema; includes syntax and semantics
  - DDL compiler generates set of tables stored in the DBMS's *data dictionary* (contains *metadata*, i.e. data about data)
  - Data storage and definition language – special type of DDL in which storage structures and access methods used by the DBS are specified
  - Data Definition in SQL (Structured Query Language):

    ```
    CREATE TABLE STUDENTS(
      SID       NUMBER,
      NAME      CHAR(70),
      MAJOR     CHAR(50),
      DOB       DATE,
      ADDRESS   CHAR(100));
    ```

- Data Manipulation Language (DML)
  - Language for accessing and manipulating the data that is organized according to underlying data model
  - Two classes of languages

    Procedural – user specifies how required data is retrieved

    Declarative – user specifies what data is required without specifying how to get those data (declarative $\hat{=}$ non-procedural)

Transaction Management

- A *transaction* is an atomic sequence of database operations that perform a logical function in a database application.
- Transaction-management component of DBMS ensures that the DB remains in a consistent (correct) state despite system failures (e.g., system crash, power failure) and transaction failures.
- Concurrency control manager controls interaction among concurrent user transactions to ensure the (read) consistency of the database.

Storage Management

- The storage manager is a DBMS software component that implements interface between low-level data stored in a database (files) and the application programs and queries submitted to the system.
- The storage manager is responsible for the following tasks:
  – interaction with the operating system's file manager
  – efficient storage, retrieval, and update of the data

Database Administrator

- Coordinates all the activities with respect to the design and implementation of the database on a DBMS; has a good understanding of the enterprise's information resources and needs.
- Database administrator's (sometimes designer's) duties include:
  - Schema definition (conceptual schema, external schema)
  - Specifying integrity constraints
  - Specifying storage structures and access method (physical schema)
  - Modifying the schema and physical organization
  - Granting user privileges and managing user roles (security)
  - Monitoring performance and responding to changes in requirements

Database Users

- Users are differentiated by the way they interact with the system
- Application programmers: interact with system through DML and DDL calls
- Sophisticated users: formulate requests in a database query language
- Specialized users: write specialized database applications that do not fit into the traditional data processing framework
- Naive users: invoke one of the permanent application programs that have been written previously

Anatomy of a DBMS

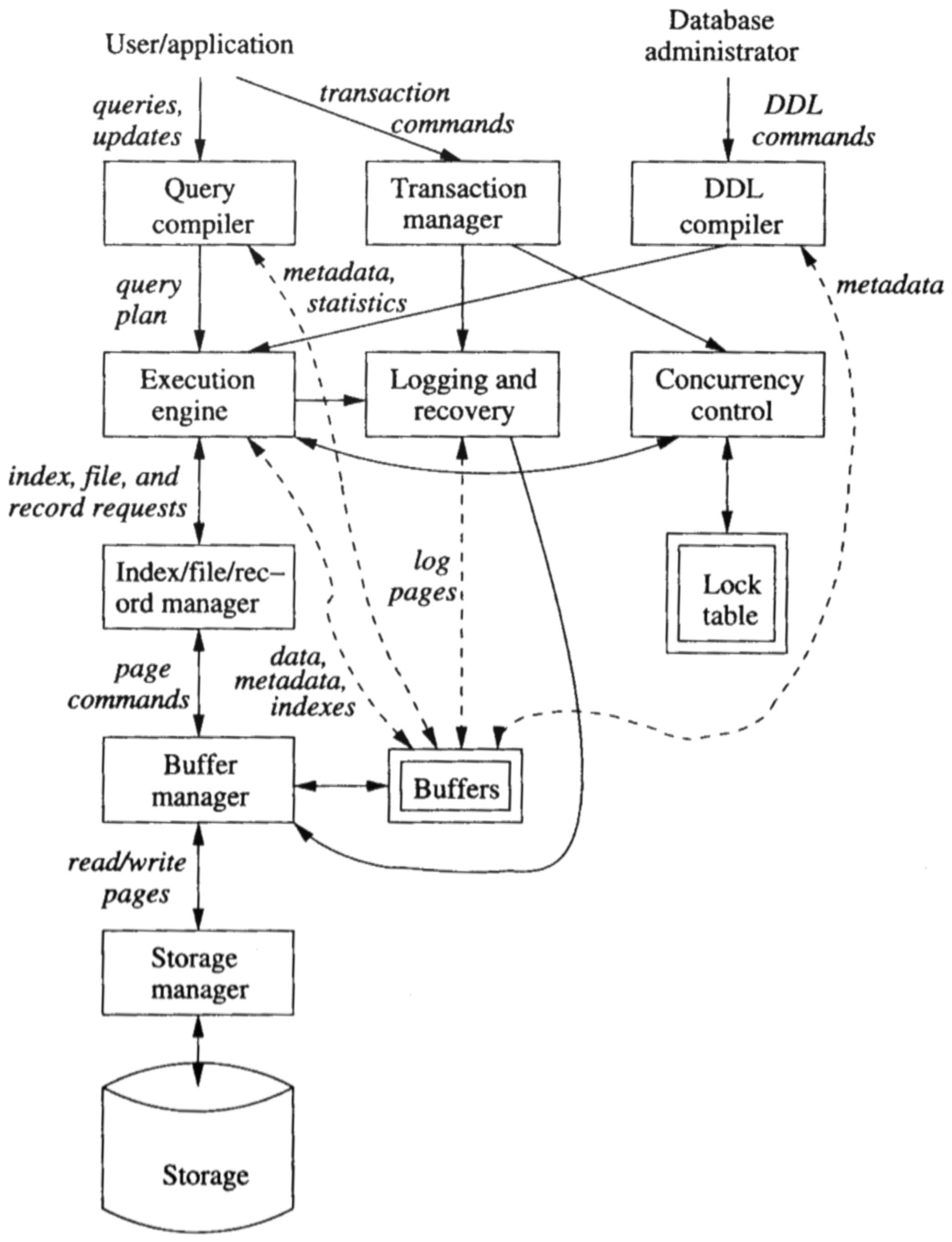


Figure 1.1: Database management system components