Review of Major DB Concepts

• Data and Information
  – Data: raw facts
  – Information: processed data
• Database
• Metadata
• DBMS: Database management system
• Database system
An Example of Data

- Sales per employee for each of X-company’s two divisions
Data and Its Structure

• Data is actually stored as bits, but it is difficult to work with data at this level.
• It is convenient to view data at different levels of abstraction.
• **Schema**: Description of data at some level. Each level has its own schema.
• Three schemas: physical, conceptual, and external.
Physical Data Schema

• Describes details of how data is stored: tracks, cylinders, indices etc.

• Early applications worked at this level - explicitly dealt with details.

• **Problem:** Routines hard-coded to deal with physical representation.
  – Changes to data structure difficult to make.
  – Application code becomes complex since it must deal with details.
  – Rapid implementation of new features impossible.
Conceptual Data Level

• Hides details.
  – In the relational model, the conceptual schema presents data as a set of tables.

• DBMS maps from conceptual to physical schema automatically.

• Physical schema can be changed without changing application:
  – DBMS must change mapping from conceptual to physical.

• Referred to as *physical data independence*. 
Conceptual Data Level (con’t)

Application

Conceptual view of data

DBMS

Physical view of data
External Data Level

- In the relational model, the external schema also presents data as a set of relations.
- An external schema specifies a *view* of the data in terms of the conceptual level. It is tailored to the needs of a particular category of users.
  - Portions of stored data should not be seen by some users.
    - Students should not see faculty salaries.
    - Faculty should not see billing data.
  - Information that can be derived from stored data might be viewed as if it were stored.
    - GPA not stored, calculated when needed.
External Data Level (con’t)

- Application is written in terms of an external schema.
- A view is computed when accessed (not stored).
- Different external schemas can be provided to different categories of users.
- Translation from external to conceptual done automatically by DBMS at run time.
- Conceptual schema can be changed without changing application:
  - Mapping from external to conceptual must be changed.
- Referred to as conceptual data independence.
Levels of Abstraction

External schemas

payroll
billing
records

View 1
Conceptual schema
Physical schema
View 2
View 3

9
Data Independence

• **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.

• **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.
### Transcript

<table>
<thead>
<tr>
<th>StudentName</th>
<th>CourseNumber</th>
<th>Grade</th>
<th>Semester</th>
<th>Year</th>
<th>SectionId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>CS1310</td>
<td>C</td>
<td>Fall</td>
<td>99</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>MATH2410</td>
<td>B</td>
<td>Fall</td>
<td>99</td>
<td>112</td>
</tr>
<tr>
<td>Brown</td>
<td>MATH2410</td>
<td>A</td>
<td>Fall</td>
<td>98</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>CS1310</td>
<td>A</td>
<td>Fall</td>
<td>98</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>CS3320</td>
<td>B</td>
<td>Spring</td>
<td>99</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>CS3380</td>
<td>A</td>
<td>Fall</td>
<td>99</td>
<td>135</td>
</tr>
</tbody>
</table>

### Grade Report

<table>
<thead>
<tr>
<th>StudentNumber</th>
<th>StudentName</th>
<th>SectionIdentifier</th>
<th>CourseNumber</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Smith</td>
<td>112</td>
<td>MATH2410</td>
<td>B</td>
</tr>
<tr>
<td>17</td>
<td>Smith</td>
<td>119</td>
<td>CS1310</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>Brown</td>
<td>85</td>
<td>MATH2410</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>Brown</td>
<td>92</td>
<td>CS1310</td>
<td>A</td>
</tr>
<tr>
<td>8</td>
<td>Brown</td>
<td>102</td>
<td>CS3320</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>Brown</td>
<td>135</td>
<td>CS3380</td>
<td>A</td>
</tr>
</tbody>
</table>
Data indep is accomplished:

changing a schema at a lower level of DB,

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.
A Database System
Historical Roots

• File systems
  – Provides historical perspective
  – Avoid pitfalls of data mgmt
  – Simple characteristics
  – Knowledge of converting a file system to a DB system
Contents of the CUSTOMER File

<table>
<thead>
<tr>
<th>C_NAME</th>
<th>C_PHONE</th>
<th>C_ADDRESS</th>
<th>C_ZIP</th>
<th>A_NAME</th>
<th>A_PHONE</th>
<th>TP</th>
<th>AMT</th>
<th>REN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred A. Ramos</td>
<td>615-844-2573</td>
<td>218 Fork Rd., Babs, TN</td>
<td>36123</td>
<td>Leah F. Hahn</td>
<td>615-882-1244</td>
<td>T1</td>
<td>$100.00</td>
<td>3/12/99</td>
</tr>
<tr>
<td>Leona K. Dunne</td>
<td>713-894-1238</td>
<td>Box 12A, Fox, KY</td>
<td>25246</td>
<td>Alex B. Alby</td>
<td>713-228-1249</td>
<td>T1</td>
<td>$250.00</td>
<td>5/23/99</td>
</tr>
<tr>
<td>Kathy W. Smith</td>
<td>615-894-2285</td>
<td>125 Oak Ln, Babs, TN</td>
<td>36123</td>
<td>Leah F. Hahn</td>
<td>615-882-2144</td>
<td>S2</td>
<td>$150.00</td>
<td>1/5/98</td>
</tr>
<tr>
<td>Paul F. Olowinski</td>
<td>615-894-2180</td>
<td>217 Lee Ln., Babs, TN</td>
<td>36123</td>
<td>Leah F. Hahn</td>
<td>615-882-1244</td>
<td>S1</td>
<td>$300.00</td>
<td>8/20/99</td>
</tr>
<tr>
<td>Myron Orlando</td>
<td>615-222-1672</td>
<td>Box 111, New, TN</td>
<td>36155</td>
<td>Alex B. Alby</td>
<td>713-228-1249</td>
<td>T1</td>
<td>$100.00</td>
<td>12/4/99</td>
</tr>
<tr>
<td>Amy B. O'Brien</td>
<td>713-442-3381</td>
<td>387 Tiroll Dr., Fox, KY</td>
<td>25246</td>
<td>John T. Okon</td>
<td>615-123-5589</td>
<td>T2</td>
<td>$850.00</td>
<td>8/29/99</td>
</tr>
<tr>
<td>James G. Brown</td>
<td>615-297-1229</td>
<td>21 Tye Rd., Nash, TN</td>
<td>37118</td>
<td>Leah F. Hahn</td>
<td>615-882-1244</td>
<td>S1</td>
<td>$120.00</td>
<td>3/1/99</td>
</tr>
<tr>
<td>George Williams</td>
<td>615-290-2556</td>
<td>155 Maple, Nash, TN</td>
<td>37119</td>
<td>John T. Okon</td>
<td>615-123-5589</td>
<td>S1</td>
<td>$250.00</td>
<td>6/23/99</td>
</tr>
<tr>
<td>Anne G. Farriss</td>
<td>713-382-7185</td>
<td>2119 Elm, Crew, KY</td>
<td>25432</td>
<td>Alex B. Alby</td>
<td>713-228-1249</td>
<td>T2</td>
<td>$100.00</td>
<td>11/9/99</td>
</tr>
<tr>
<td>Olette K. Smith</td>
<td>615-297-3809</td>
<td>2782 Main, Nash, TN</td>
<td>37118</td>
<td>John T. Okon</td>
<td>615-123-5589</td>
<td>S2</td>
<td>$500.00</td>
<td>2/18/99</td>
</tr>
</tbody>
</table>

C_NAME = Customer name  
C_PHONE = Customer phone  
C_ADDRESS = Customer address  
C_ZIP = Customer ZIP code  
A_NAME = Agent name  
A_PHONE = Agent phone  
TP = Insurance type  
AMT = Insurance policy amount, in thousands of $  
REN = Insurance renewal date
# Basic File Terminology

<table>
<thead>
<tr>
<th><strong>Data</strong></th>
<th>“Raw” facts that have little meaning unless they have been organized in some logical manner. The smallest piece of data that can be “recognized” by the computer is a single character, such as the letter A, the number 5, or some symbol such as; ‘? &gt; * +. A single character requires one byte of computer storage.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Field</strong></td>
<td>A character or group of characters (alphabetic or numeric) that has a specific meaning. A field might define a telephone numbers, a birth date, a customer name, a year-to-date (YTD) sales value, and so on.</td>
</tr>
<tr>
<td><strong>Record</strong></td>
<td>A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that comprise a record for a customer named J. D. Rudd might consist of J. D. Rudd’s name, address, phone number, date of birth, credit limit, unpaid balance, and so on.</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>A collection of related records. For example, a file might contain data about Company’s vendors; or, a file might contain the records for the students currently enrolled at University.</td>
</tr>
</tbody>
</table>
Contents of the AGENT File

<table>
<thead>
<tr>
<th>A_NAME</th>
<th>A_PHONE</th>
<th>A_ADDRESS</th>
<th>ZIP</th>
<th>HIRED</th>
<th>YTD_PAY</th>
<th>YTD_FIT</th>
<th>YTD_FICA</th>
<th>YTD_SLS</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex B. Atty</td>
<td>713-228-1249</td>
<td>123 Toll, Nash, TN</td>
<td>37119</td>
<td>11/1/93</td>
<td>$20,568.24</td>
<td>$4,332.21</td>
<td>$1,534.57</td>
<td>$1,735.00</td>
<td>3</td>
</tr>
<tr>
<td>Leah F. Hahn</td>
<td>615-882-1244</td>
<td>334 Main, Fox, KY</td>
<td>25246</td>
<td>5/23/94</td>
<td>$25,213.76</td>
<td>$5,934.75</td>
<td>$1,788.36</td>
<td>$4,967.00</td>
<td>0</td>
</tr>
<tr>
<td>John T. Okon</td>
<td>615-123-5589</td>
<td>452 Elm, New, TN</td>
<td>36155</td>
<td>6/15/95</td>
<td>$23,198.29</td>
<td>$4,332.24</td>
<td>$1,689.44</td>
<td>$3,093.00</td>
<td>2</td>
</tr>
</tbody>
</table>

A_NAME = Agent name
A_PHONE = Agent phone
A_ADDRESS = Agent address
ZIP = Agent ZIP code
HIRED = Agent date of hire
YTD_PAY = Year-to-date pay
YTD_FIT = Year-to-date federal income tax paid
YTD_FICA = Year-to-date Social Security paid
YTD_SLS = Year-to-date sales, in thousands of $
DEP = Number of dependents
A Simple File System
File System Critique

• File System Data Management
  – Require extensive programming in a 3GL
  – System admini. becomes difficult, as the # of file expands
  – Difficult and important to make changes in existing file structure
  – Omit security feature to safeguard data
  – Island of data information
File System Critique

• Structural and Data Dependence
  – Structural Dependence
  – Data Dependence

• Data dependence makes file system cumbersome
File System Critique

• Field Definitions and Name Conventions
  – A flexible (good) record definition anticipates reporting requirements by breaking up fields into their components.
  – Example:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUS_LNAME</td>
<td>Customer last name</td>
</tr>
<tr>
<td>CUS_FNAME</td>
<td>Customer first name</td>
</tr>
<tr>
<td>CUS_INITIAL</td>
<td>Customer initial</td>
</tr>
<tr>
<td>CUS_AREAACODE</td>
<td>Customer area code</td>
</tr>
<tr>
<td>CUS_PHONE</td>
<td>Customer phone</td>
</tr>
<tr>
<td>CUS_ADDRESS</td>
<td>Customer street address or box number</td>
</tr>
<tr>
<td>CUS_CITY</td>
<td>Customer city</td>
</tr>
<tr>
<td>CUS_STATE</td>
<td>Customer state</td>
</tr>
</tbody>
</table>
File System Critique

– Selecting proper file names is very important
  • descriptive within restrictions
  • Reflect documentation

• Data Redundancy
  – Data inconsistency (lack of data integrity)
  – Data anomalies
    • Modification anomalies
    • Insertion anomalies
    • Deletion anomalies
DB System Vs File System

[Diagram showing a database system with departments and a database, and a file system with separate files for employees, customers, sales, inventory, and accounts.]
DBMS Languages

- **Data Definition Language (DDL):**
  - Used by the DBA 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.
  - Result of compiling DDL is *catalog* (a set of tables stored in a file).

- **Data Manipulation Language (DML):** Used to specify database retrievals and updates.
  - High-level (nonprocedural, declarative) DML:
    - Describe what data is needed w/o specifying how to get it
    - DML commands (*data sublanguage*) can be *embedded* in a general-purpose programming language (*host language*), such as COBOL, PL/1 or PASCAL.
  - Low-level (procedural) DML:
    - Describe what & how
    - *stand-alone* DML commands can be applied directly (*query language*).
Data Model

• **Schema**: description of data at some level (e.g., tables, attributes, constraints, domains)

• **Model**: tools and language for describing:
  – Conceptual and external schema
    • Data definition language (DDL)
  – Integrity constraints, domains (DDL)
  – Operations on data
    • Data manipulation language (DML)
  – Directives that influence the physical schema (affects performance, not semantics)
    • Storage definition language (SDL)
DBMS Interfaces

• Stand-alone query language interfaces
• Programmer interfaces for embedding DML in programming languages:
  – Pre-compiler Approach
  – Procedure (Subroutine) Call Approach
• User-friendly interfaces
  – Menu-based
  – Graphics-based (Point and Click, Drag and Drop etc.)
  – Forms-based
  – Natural language
  – Combinations of the above
  – Web Browser as an interface
DBMS Interfaces (Cont.)

- Parametric interfaces using function keys
- Report generation languages
- Interfaces for the DBA
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access path
Database System Environment
Database System Components

• Hardware
  – Computer
  – Peripherals

• Software
  – Operating system software
  – DBMS software
  – Applications programs and utilities software
Database System Components

• People
  – Systems administrators
  – Database administrators (DBAs)
  – Database designers
  – Systems analysts and programmers
  – End users

• Procedure
  – Instructions and rules

• Data
  – Collection of raw facts
Database System Components

• Complexity of DB system depends on
  – Organization’s size
  – Function
  – Corporate culture
  – Activities and environment

• DB solutions must be cost effective & strategically effective
DBMS Component Modules

- Stored data manager
- DDL compiler
- Run-time database processor
- Query compiler
- Pre-compiler
Figure 2.3  Typical component modules of a DBMS. Dotted lines show accesses that are under the control of the stored data manager.
Database System Utilities

• *Loading* data stored in files into a database.
• *Backing up* the database periodically on tape.
• *Reorganizing* database file structures.
• *Report generation* utilities.
• *Performance monitoring* utilities.
• Other functions,
  – such as *sorting*, *user monitoring*, *security mgmt*, *data integrity*, *data compression*, etc.
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.
Types of Database Systems

• **Data model used:**
  – Traditional: Relational, Network, Hierarchical.
  – Emerging: Object-oriented, Object-relational, Temporal, Spatial

• **Other classifications:**
  – Single-user (typically used with micro-computers) vs. multi-user (most DBMSs).
  – Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

  – Use
    • Transactional (Production)
    • Decision support
    • Data warehouse
Database Models

• A set of logical constructs
  – Represent data structure & data relationships

• Two types of database models
  – Conceptual model
    • Logical nature of data representation
    • What is represented
  – Implementation model
    • How the data are represented
    • How the data structure are implemented
Database Models

• Three types of relationships
  – One-to-many (1:M)
    • PAINTER(1) *paints* PAINTING (M)
  – Many-to-many (N:M)
    • STUDENT(N) *takes* COURSE (M)
  – One-to-one (1:1)
    • EMPLOYER (1) *manages* STORE (1)
Implementation Database Models

- Hierarchical database model
- Network database model
- Relational database model
A Hierarchical Structure
Hierarchical Data Model

• Pros
  – Conceptual simplicity
  – Database security
  – Data independence
  – Database integrity
  – Efficiency dealing with a large DB

• Cons
  – Complex implementation
  – Difficult to manage
  – Lacks structural independence
  – Applications programming and use complexity
  – Implementation limitation
  – Lack of standards
• Child with multiple parents
A Network Database Model
Network Database Model

• Pros
  – Conceptual simplicity
  – Handles more relationship types
  – Data access flexibility
  – Data independence
  – Conformance to standards

• Cons
  – System complexity
  – Lack of structural independence
Relational Database Model

<table>
<thead>
<tr>
<th>CUS_CODE</th>
<th>CUS_LNAME</th>
<th>CUS_FNAME</th>
<th>CUS_INITIAL</th>
<th>CUS_AREA_CODE</th>
<th>CUS_PHONE</th>
<th>CUS_RENEW_DATE</th>
<th>AGENT_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010</td>
<td>Ramnas</td>
<td>Alfred</td>
<td>A</td>
<td>615</td>
<td>844-2573</td>
<td>3/12/99</td>
<td>502</td>
</tr>
<tr>
<td>10011</td>
<td>Dunne</td>
<td>Leon</td>
<td>K</td>
<td>713</td>
<td>594-1238</td>
<td>5/23/99</td>
<td>501</td>
</tr>
<tr>
<td>10012</td>
<td>Smith</td>
<td>Kathy</td>
<td>W</td>
<td>615</td>
<td>894-2285</td>
<td>1/5/99</td>
<td>502</td>
</tr>
<tr>
<td>10013</td>
<td>Olversti</td>
<td>Paul</td>
<td>F</td>
<td>615</td>
<td>894-2188</td>
<td>8/20/99</td>
<td>502</td>
</tr>
<tr>
<td>10014</td>
<td>Oriaide</td>
<td>Myron</td>
<td>D</td>
<td>615</td>
<td>222-1872</td>
<td>12/4/99</td>
<td>501</td>
</tr>
<tr>
<td>10015</td>
<td>O'Brian</td>
<td>Amy</td>
<td>D</td>
<td>713</td>
<td>442-3391</td>
<td>8/26/99</td>
<td>503</td>
</tr>
<tr>
<td>10016</td>
<td>Brown</td>
<td>James</td>
<td>G</td>
<td>615</td>
<td>297-1228</td>
<td>3/19/99</td>
<td>502</td>
</tr>
<tr>
<td>10017</td>
<td>Williams</td>
<td>George</td>
<td>G</td>
<td>015</td>
<td>290-2556</td>
<td>6/23/99</td>
<td>503</td>
</tr>
<tr>
<td>10018</td>
<td>Fairies</td>
<td>Anne</td>
<td>G</td>
<td>713</td>
<td>362-7185</td>
<td>11/6/99</td>
<td>501</td>
</tr>
<tr>
<td>10019</td>
<td>Smith</td>
<td>Gladis</td>
<td>K</td>
<td>015</td>
<td>287-3009</td>
<td>2/16/99</td>
<td>503</td>
</tr>
</tbody>
</table>

Table name: CUSTOMER

Table name: AGENT

Link through AGENT_CODE
Relational Database Model

• Pros
  – Structural independence
  – Improved conceptual simplicity
  – Easier database design, implementation, mgmt, and use
  – Ad hoc query capability (SQL)
  – Powerful DBMS

• Cons
  – Hardware and software overhead
  – Possibility of poor design and implementation
  – Potential “islands of information” problems
Object-Oriented Database Model

• Basic Structure
  – Objects
  – Attribute
  – Class

• Comparison of OO data model and ER data model
OO Database Model

• **Pros**
  – Add semantic content
  – Visual presentation includes semantic content
  – Database integrity
  – Both structural and data **independence**

• **Cons**
  – Lack of OODM standards
  – Complex navigational data access
  – High system overhead slows transaction
The Development of Data Model
60's  
Hierarchical  
Network  

70's  
Relational  
Choice for most new applications  

80's  
Object Bases  
Knowledge Bases  

90's  
now  

Wrap-up: The Evolution of Data Models

• Common characteristics required for data models
  – Some degree of conceptual simplicity
  – Represent the real word as close as possible
  – Representation of mini-word behavior must be in compliance with the consistency and integrity of any data model
• Database models and the internet
  – Flexible, efficient, and secure internet access
  – Support for complex data types and relationships
  – Seamless interfacing with multiple data sources and structures
  – Simplicity of conceptual database model
  – An abundance of available database tools
  – A powerful DBMS makes DBA’s job easier