Database Fundamentals

Chapter 1

What is a Database?

- The ISO/ANSI SQL Standard does not contain a definition of the term database. In fact, the term is never mentioned in the standard.
- Most believe this is because vendors already had products on the market using their interpretation of the term.

What is a Database?

- Vendors have different interpretations:
  - DB2: a logical collection of tables
  - Oracle: a set of files that store database objects; Oracle has a "schema" that is roughly what DB2 and Sybase/MS SQL call a "database"
  - Sybase/MS SQL Server: a logical group of database objects
  - MS Access: a single file that contains a group of related database objects
  - MySQL: the terms "database" and "schema" are used interchangeably

What is a Database?

- General definition:
  - A database is a collection of interrelated data items that are managed as a single unit.
  - Databases have distinct properties that distinguish them from other data storage mechanisms.
    - In subsequent topics, we will look at properties that help define the difference between databases and other types of file storage systems, such as spreadsheets.

Other Important Terms

- ***File:*** a collection of related records that are stored as a single unit by an operating system. Databases use files for permanent storage of data.
- ***Instance:*** a copy of the database software running in memory.
- ***Database object:*** a named data structure that is stored in a database. For example, in a relational database, tables and views are types of database objects

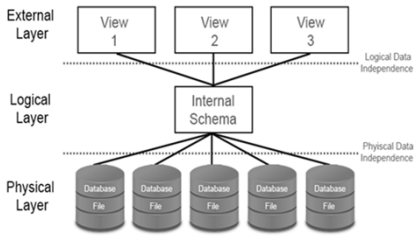
What is a Database Management System (DBMS)?

- Software provided by the database vendor
- Provides basic services:
  - Moving data to/from physical files
  - Managing concurrent access by multiple users
  - Managing transactions so that each is an "all or nothing" unit of work
  - Support for a *query language*
  - Provisions for backup and recovery
  - Security mechanisms to prevent unauthorized data access and modification

DBMS Layers of Abstraction

- The DBMS presents users with with distinct views of the data (*user views*), while storing the data only once.
- *User:* any person or application that signs onto the database
- *Application:* a set of computer programs designed to solve a particular business problem; e.g. payroll, order entry

DBMS Layers of Abstraction



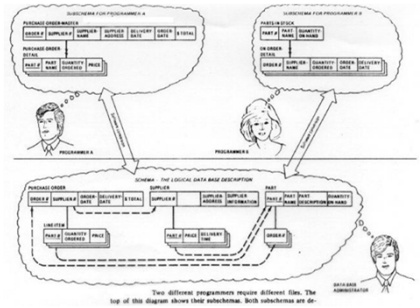
DBMS Layers of Abstraction

- *Physical Layer:* contains the data files that hold the data
- *Logical Layer:*
  - First of two layers of abstraction
  - Presents the database schema (tables in an RDBMS)
  - DBMS handles transformation from physical to logical
  - DBMS software automatically issues the appropriate operating system file commands.
  - Database user is unaware of which files are used, or even the existence of the files.

ANSI/SPARC Layers

- *External Layer:*
  - Second layer of abstraction
  - Composed of user views (predefined or dynamic)
  - DBMS transforms schema objects into views
  - Sometimes called the *sub-schema*
- These layers separate the presentation of data that application programs and business users see from the physical storage of the data.

User Views



Two different programmers require different files. The top of this diagram shows their subschemas. Both subschemas are de-

Data Independence

- ***Data independence***: the ability to make non-destructive data structure changes without disrupting existing processes and queries that use the database.
- Provided by the DBMS layers of abstraction.
- Data independence is not an absolute "have" or "have not" property, but rather a matter of degrees (having more or less data independence compared to another alternative).
- Two forms: physical and logical (next slides)

Physical Data Independence

- The ability to make changes in the physical layer without disrupting existing database users and operations.
  Examples:
  - Moving a file from one disk to another
  - Moving a database object from one file to another
- Not a "have or have not" property
- All computer systems have it to some degree
- Achieved in DBMS by the transformation from the physical layer to the logical layer

Logical Data Independence

- The ability to make changes in the logical layer without disrupting existing database users and operations.
  Examples:
  - Adding a new database object, such as a table
  - Modifying an existing object, such as adding a new column
- Not a "have or have not" property
- Not found outside of DBMSs
- Achieved in DBMS by the transformation from the logical layer to the external layer

Historically Significant Database Models

- Flat Files
- Hierarchical Model
- Network Model
- Relational Model
- Object-Oriented Model
- Object-Relational Model

Flat File "Database" Model

- Consists of one or more readable files
  - Technically not databases
- Data can be accessed in only one form – the way it is physically stored
- Drawbacks:
  - No relationships among files
  - Difficult to ensure accuracy
  - Redundant data usually necessary
  - Physical location of data must be known
  - User/application must manage any relationships

Flat File Model (Nothwind)

Customer File

Customer ID	Company Name	Contact First Name	Contact Last Name	Job Title	City	State
6	Company F	Francisco	Pérez-Olaeta	Purchasing Manager	Milwaukee	WI
26	Company Z	Run	Liu	Accounting Assistant	Miami	FL

Employee File

Employee ID	First Name	Last Name	Title
2	Andrew	Cencini	Vice President, Sales
5	Steven	Thorpe	Sales Manager
9	Anne	Hellung-Larsen	Sales Representative

Product File

Product ID	Product Code	Product Name	Category	Quantity Per Unit	List Price
5	NWTO-5	Northwind Traders Olive Oil	Oil	36 boxes	$21.35
7	NWTDFN-7	Northwind Traders Dried Pears	Dried Fruit & Nuts	12 - 1 lb pkgs.	$30.00
40	NWTCM-40	Northwind Traders Crab Meat	Canned Meat	24 - 4 oz tins	$18.40
41	NWTSO-41	Northwind Traders Clam Chowder	Soups	12 - 12 oz cans	$9.65
48	NWTCA-48	Northwind Traders Chocolate	Candy	10 pkgs.	$12.75
51	NWTDFN-51	Northwind Traders Dried Apples	Dried Fruit & Nuts	50 - 300 g pkgs.	$53.00

Order File

Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee
51	26	9	4/5/2006	4/5/2006	$60.00
56	6	2	4/3/2006	4/3/2006	$0.00
79	6	2	6/23/2006	6/23/2006	$0.00

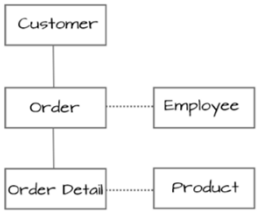
Order Detail File

Order ID	Product ID	Unit Price	Quantity
51	5	$21.35	15
51	41	$9.65	21
51	40	$18.40	2
56	48	$12.75	20
79	7	$30.00	14
79	51	$53.00	8

Hierarchical Data Model

- Record types organized into a hierarchy that appears as an inverted tree
- Parent record may have many child record types, but child may have only one parent type
- Related records connected via a physical address pointer (typically relative byte address, or RBA)
  - A *relative byte address* refers to a location within a file in terms of how many bytes it is from the beginning of a file.
  - Alternatively, address can be a block/record number.
- Child records may only be accessed via their parents

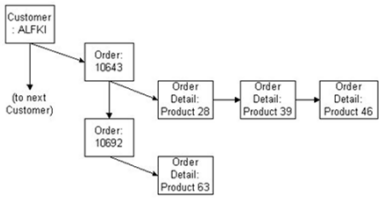
Hierarchical Data Model (Northwind)



- Red dotted lines represent relationships that cannot be directly implemented in a pure hierarchical DBMS

Hierarchical DBMS Pointer Chains

F01-04: Hierarchical Model Record Contents for Northwind



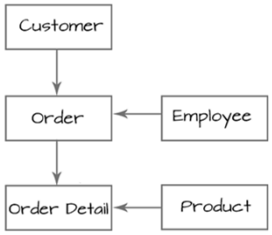
Hierarchical Data Model

- Advantages (compared to flat files):
  - Faster data retrieval
  - Data integrity easier to manage
- Drawbacks:
  - Users must know and understand data structure
  - Redundant data may be required to work around the "one parent" restriction

Network Data Model

- Similar to Hierarchical except that child ("member") record types may have multiple parent ("owner") record types
- Relationships (called "sets") are named
- Data access involves "walking the sets"

Network Data Model (Northwind)

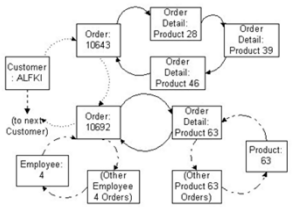


Network Data Model

- Relationships are implemented as *sets*, with each set being implemented as a chain of pointers (similar to the pointers used in hierarchical databases)
- Each set is given a unique name within the database and database users must know the names of the sets in order to navigate the data structures.
- In diagrams, sets are shown as lines that connect owner and member, with an arrowhead pointing towards the member of the set.

Network Model Data Access

F01-06: Network Model Record Contents for NorthWind



- Data access requires "walking the sets" by issuing commands (e.g. "get next member") against sets

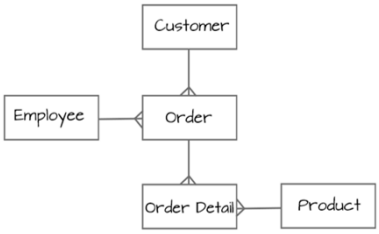
Network Data Model

- Benefits:
  - Fast data access
  - Can start access with any record type
  - Models more complex data structures
  - Easier to develop complex queries
- Drawbacks:
  - Data structure difficult to modify
  - Changes to data structure often affect the applications
  - User must understand complex data structure

Relational Data Model

- Developed in response to inflexible data structures
- Primary unit of storage is the table
- Each table may be used independently or joins may be used to combine tables
- Relationships defined using primary and foreign keys instead of pointers
- Easily defined integrity constraints

Relational Data Model (Northwind)



Relationship Definition

Customer Table

Customer ID	Company Name	Contact First Name	Contact Last Name	Job Title	City	State
6	Company F	Francisco	Perez-Olaeta	Purchasing Manager	Milwaukee	WI
26	Company Z	Run	Lou	Accounting Assistant	Miami	FL

Order Table

Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee
51	26	9	4/5/2006	4/5/2006	$60.00
58	6	2	4/3/2006	4/3/2006	$0.00
79	6	2	6/22/2006	6/23/2006	$0.00

Employee Table

Employee ID	First Name	Last Name	Title
2	Andrew	Cencini	Vice President, Sales
5	Steven	Thorpe	Sales Manager
9	Anne	Hellung-Larsen	Sales Representative

- Relationships can be formally defined in the DBMS
- DBMS ensures that foreign key values always have a matching primary key in the parent table

Relational Database Model

- Benefits:
  - Very fast retrieval in most cases
  - Data structure easily changed
  - Users need no awareness of physical storage
  - Complex queries easily developed
  - Data usually more accurate
  - Easier application programming
  - Standard query language (SQL)

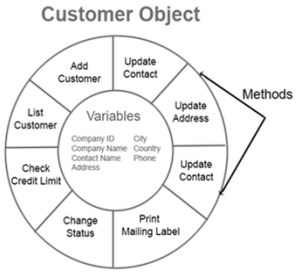
Relational Database Model

- Drawbacks:
  - Tables must be joined to retrieve related data (joins can be hidden in views, however)
  - Users must understand relationships between tables
  - Users must learn SQL or a front-end query tool

Object-Oriented (OO) Data Model

- Primary tenet: data (variables) should not be freely accessed because doing so compromises the the independence of data from the processes.
  - Program modules called *methods* perform all data access
- Object-oriented language (e.g. C++, Java) allows programmers to access abstract objects containing both processing methods and data
- Pure OO database stores objects in a specialized database for persistence
- Earliest OO system was Smalltalk, developed in the 1970s

The Anatomy of an Object



OO Data Model

- Benefits
  - Programmers need to understand only OO concepts
  - Objects can inherit from a class structure
  - Potentially highly automated application development
  - Easier to manage (at least in theory)
  - More compatible with OO languages and tools than Object-Relational
- Disadvantages
  - Database user must understand OO concepts
  - Niche market products up to now
  - General lack of industry standards

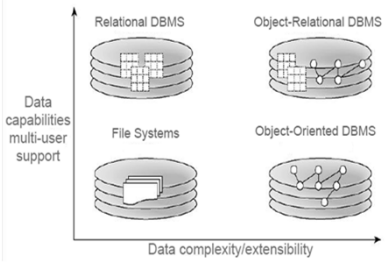
Object-Relational (OR) Database Model

- Combines OO and Relational DBMS concepts, supposedly taking the best of both
- SQL3 (SQL99) incorporated object operations

OR Data Model

- Benefits
  - Better OO language support (less need to translate tables to objects and back)
  - Adds User Defined Types to relational
- Drawbacks:
  - Database user must understand relational and OO concepts
  - Currently no support for object inheritance
  - Object purists don't accept it

Database Model Comparison



Why Use Relational?

- Most stable
- Well established ANSI and ISO standards
- Many vendors from which to choose
- Relatively easy to convert between vendor implementations
- Easy to define, maintain and manipulate data with SQL
- Simple ad hoc queries
- Data is well protected using defined integrity constraints

History of Database Management Systems

- GUAM (Generalized Update Access Method from North American Aviation) provided the first ***hierarchical structure***
- In the mid-60's, IBM joined NAA to develop GUAM into IMS (Information Management System), the first commercial hierarchical DBMS

History of DBMSs

- IDS (Integrated Data Store) from General Electric as the first ***network*** DBMS
- IBM extended IMS to include some network capabilities, overcoming the “single parent” restriction
- CODASYL (Conference on Data Systems Languages) and the DBTG (Database Task Group) published standards in 1969-1971

History of DBMSs

- Dr. E.F. (Ted) Codd delivers pioneering white paper "A Relational Model of Data for Large Shared Data Banks" in June, 1970.
- CODASYL published specifications for a standard Network DBMS in 1971.
- This began 5 years of heated controversy between a group of Network DBMS advocates (including Charles Bachman) and a group of Relational advocates.

CODASYL “Camp" Position

- Relational Model too mathematical; programmers would not be able to understand the new manipulation languages.
- An efficient implementation of Relational could not be built.
- Online transaction processing applications want to do record at a time processing.

Relational "Camp" Position

- Nothing as complicated as the DBTG proposal could possibly be the right way to do data management.
- Set-oriented queries are too difficult to program using the DBTG language.
- CODASYL has no formal underpinnings with which to define the semantics of the complex operations of the model.

CODASYL vs. Relational

- The debate came to head at the 1975 ACM-SIGMOD conference in a debate where Codd and 2 others squared off against Bachman and 2 others. The audience was more confused at the end of the two talks and ensuing discussion than at the outset.
- Several years of debate followed.

CODASYL vs. Relational

- By the late 1970's, interest in CODASYL began to decline. Michael Stonebraker believes this was because of 2 factors:
  - Easier to use relational languages such as QUEL and SQL were developed.
  - Prototype Relational RDBMS's were developed that proved that implementations could be done with reasonable efficiency:

Prototype Relational DBMSs

- System R, developed by 15 IBM researchers in San Jose under the direction of Frank King from 1974 to 1978.
- INGRES developed by a pickup team of UC Berkeley students under the direction of Michael Stonebraker and Eugene Wong from 1973 to 1977.

Early Commercial Relational DBMSs

- System R was built commercially and became the basis for HP ALLBASE and IDMS/SQL. A bit later, Larry Ellison started Oracle and independently implemented the external specifications for System R. IBM, with some rewriting, developed System R into SQL/DS and DB2.

Early Commercial Relational RDBMS's

- INGRES also went commercial and was very successful. Since Bob Epstein moved from INGRES to Britton-Lee and then to startup SYBASE, there are some INGRES roots in these products.

Source: "Readings in Database Systems" (second edition), edited by Michael Stonebraker,
Morgan Kaufmann Publishers, 1994.

History of DBMSs

- Object Oriented systems emerged in the 1970's, but were not commercially successful until the 1980's.
- Object Relational systems emerged in the the 1990's and many RDBMS products added object-like features
- XML databases showed promise for a time and subsequently faded.
- NoSQL databases (mostly column and document oriented) are emerging quickly.

History of DBMSs

- In 1976, Peter Chen presented the Entity-Relationship model, bolstering modeling weaknesses in the relational model. Many other modeling techniques followed.