COP 4710: Database Systems
Fall 2013

Chapter 2 – Introduction to Data Modeling
Part 1 – The Data Models

Instructor : Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4710/fall2013

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

Introduction to Data Modeling

- Database design focuses on how the database structure will be used to store and manage end-user data.
- Data modeling, the first step in designing a database, refers to the process of creating a specific data model for a determined problem domain.
- A problem domain is a clearly defined area within the real-world environment, with well-defined scope and boundaries, that will be systematically addressed.
- A data model is a relatively simple representation, usually, graphical in nature, of more complex real-world data structures. In general terms, a model is an abstraction of a more complex real-world object or event. The model's main function is to help you understand the complexities of the real-world environment.

Introduction to Data Modeling

- Designers, programmers, and end users see data in different ways.
- Different views of the same data lead to designs that do not reflect how an organization operates.
- Data modeling reduces the complexities of database design. It is an iterative and progressive process that begins with a simple understanding of the problem domain, and as your understanding increases, so does the level of detail in the data model.
- Various degrees of data abstraction help reconcile varying views of same data. When done properly, the final data model effectively is a "blueprint" with all the instructions to build the database that will meet all end-user requirements.

Introduction to Data Modeling

- The importance of data modeling cannot be overstated.
- Data constitute the most basic information unites employed by a system. Applications are created to manage data and to help transform data into information.
- Recall though, that data are viewed in different ways by different users. Consider how different the views of the purchasing department manager and the inventory manager might be in an organization. The inventory manager is more concerned with inventory levels, while the purchasing manager is more concerned with the cost of items and the relationships with the suppliers of those items.

Introduction to Data Modeling

- Semantic data models attempt to capture the "meaning" of a database. Practically, they provide an approach for conceptual data modeling.
- Over the years there have been several different semantic data models that have been proposed.
- By far the most common is the entity-relationship data model, most often referred to as simply the E-R data model.
- The E-R model is often used as a form of communication between database designers and the end users during the developmental stages of a database.

Introduction to Data Modeling (cont.)

- The E-R model contains an extensive set of modeling tools, some of which we will not be concerned with as our primary objective is to give you some insight into conceptual database design and not learning all of the ins and outs of the E-R model.
- Another conceptual modeling which is becoming more common is the *Object Definition Language* (ODL) which is an object-oriented approach to database design that is emerging as a standard for object-oriented database systems.

Database Design

- The database design process can be divided into six basic steps. Semantic data models are most relevant to only the first three of these steps.

1. *Requirements Analysis:* The first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements. Often this is an informal process involving discussions with user groups and studying the current environment. Examining existing applications expected to be replaced or complemented by the database system.

Database Design (cont.)

2. *Conceptual Database Design:* The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints that are known to hold on this data.

3. *Logical Database Design*: A DBMS must be selected to implement the database and to convert the conceptual database design into a database schema within the data model of the chosen DBMS.

Database Design (cont.)

4. *Schema Refinement*: In this step the schemas developed in step 3 above are analyzed for potential problems. It is in this step that the database is *normalized*. Normalization of a database is based upon some elegant and powerful mathematical theory. We will discuss normalization later in the term.

5. *Physical Database Design:* At this stage in the design of a database, potential workloads and access patterns are simulated to identify potential weaknesses in the conceptual database. This will often cause the creation of additional indices and/or clustering relations. In critical situations, the entire conceptual model will need restructuring.

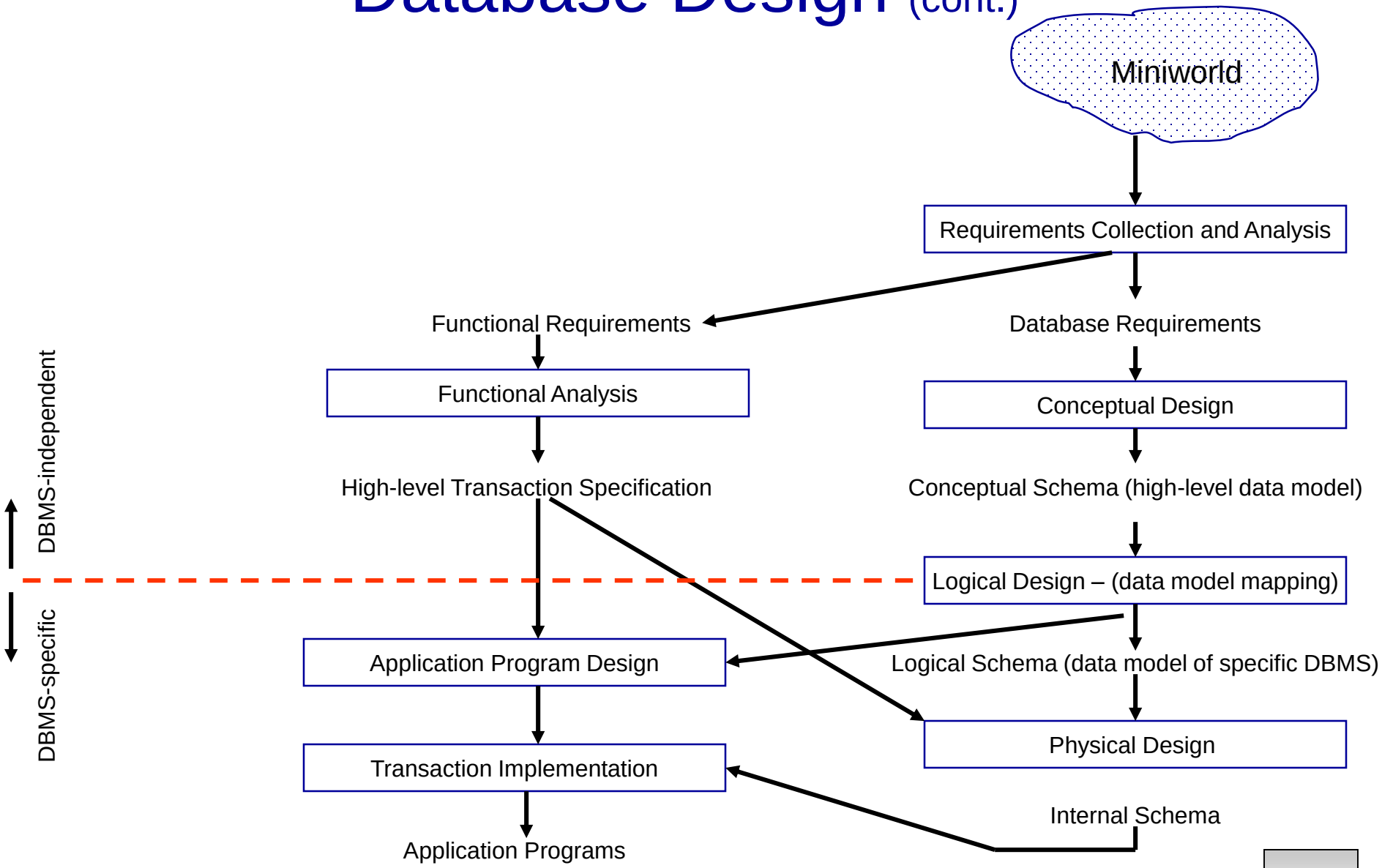
Database Design (cont.)

6. *Security Design:* Different user groups are identified and their different roles are analyzed so that access patterns to the data can be defined.

- There is often a seventh step in this process with the last step being a *tuning phase*, during which the database is made operational (although it may be through a simulation) and further refinements are made as the system is "tweaked" to provide the expected environment.

- The illustration on the following page summarizes the main phases of database design.

Database Design (cont.)



Data Model Basic Building Blocks

- The basic building blocks of all data models are entities, attributes, relationships, and constraints.
- An entity is a person, place, thing, or event about which data will be collected and stored. An entity represents a particular type of object in the real world, which means that an entity is "distinguishable" that is, each occurrence is unique and distinct.
- An entity may be a tangible object, i.e., one that you can touch such as a person or a product. An entity may also be intangible, such as a flight route, or a rock concert (an event).
- An attribute is a characteristic of an entity. For example, a customer entity would be described by attributes such as last name, first name, phone numbers, address, etc.. As we will see, it is also possible for relationships to have attributes.

Data Model Basic Building Blocks

- A relationship describes a bi-directional association among entities. For example, a relationship exists between customers and sales agents that could be described as follows: A sales agent can serve many customers, and each customer may be served by one sales agent.

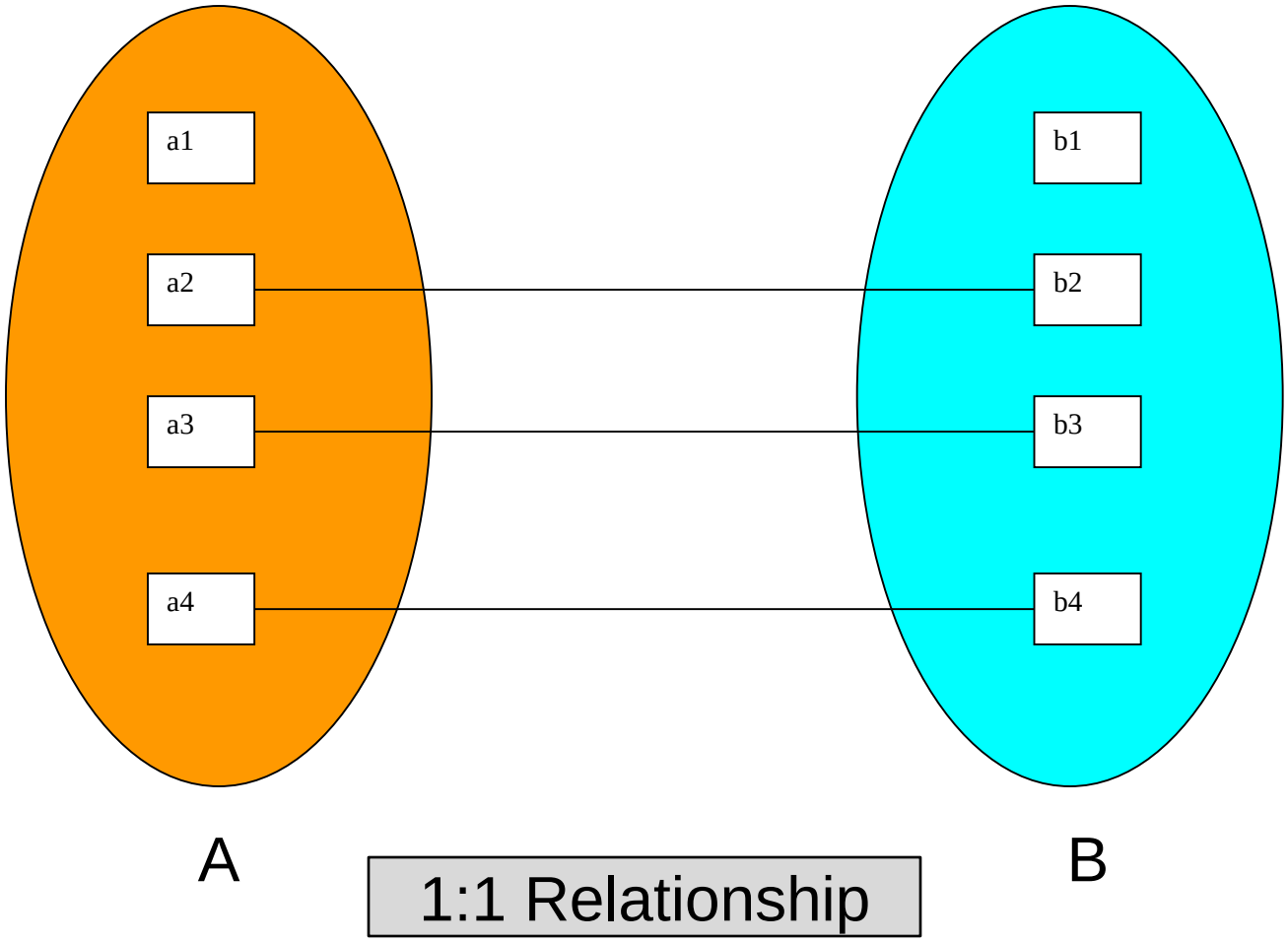
- Data models use three types of relationships: one-to-many, many-to-many, and one-to-one. Database designers and most data modeling tools use the shorthand notations 1:M or 1..*, M:N or *..*, and 1:1 or 1..1, respectively.

1:M – a painter creates many different paintings, but each is painted by only one painter.

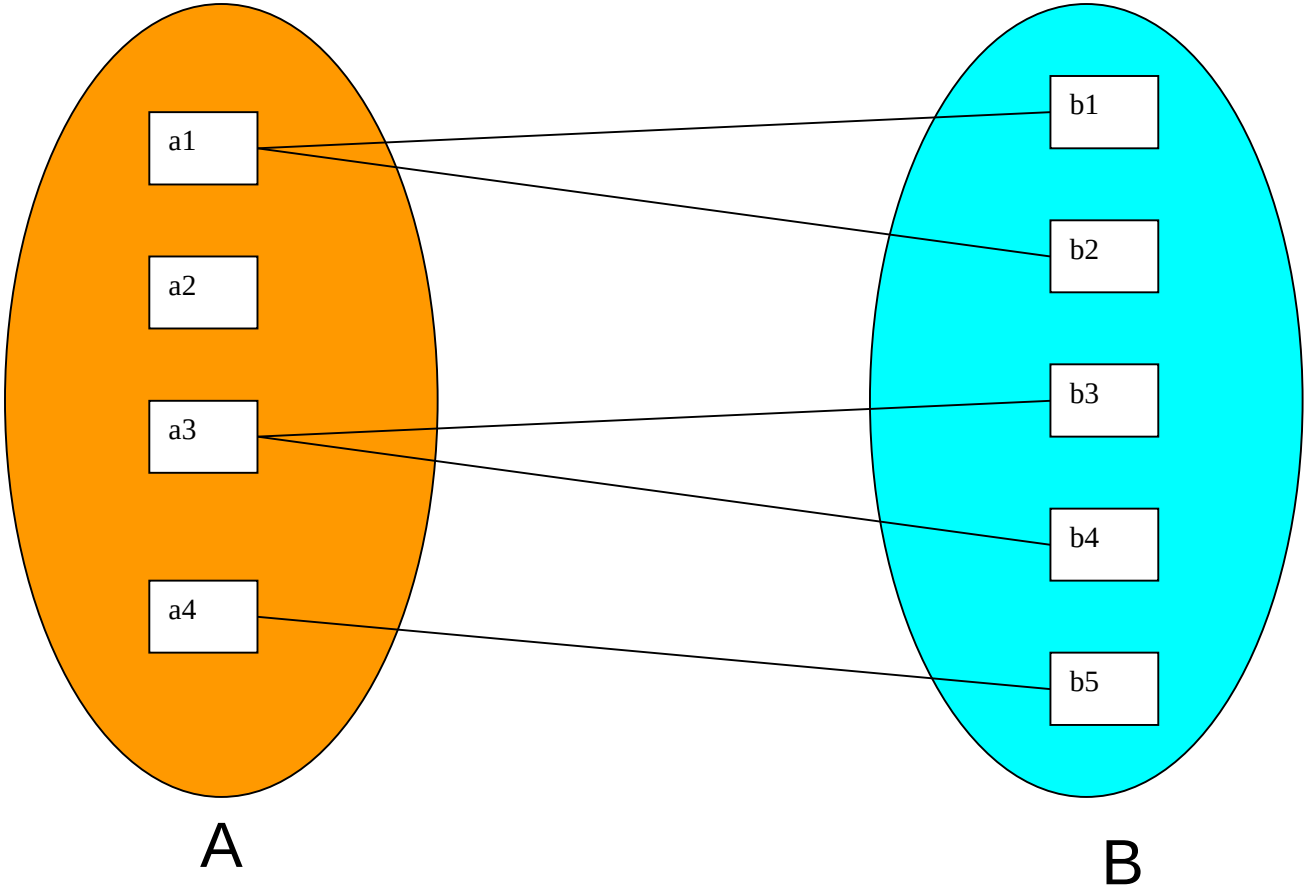
M:N – an employee may learn many job skills, and each job skill may be learned by many employees.

1:1 – Each department must have only one employee who is a manager and each employee who is a manager can manage only one department.

Data Model Basic Building Blocks

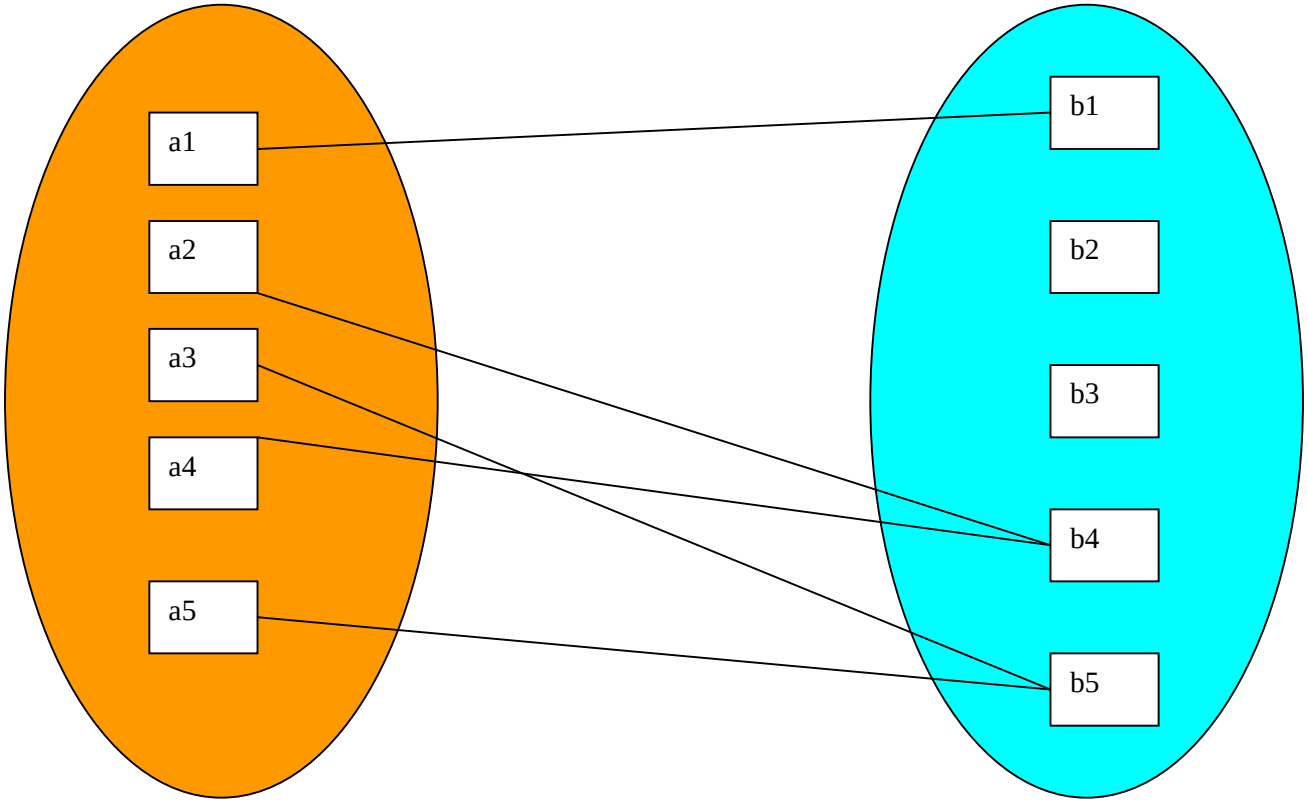


Data Model Basic Building Blocks



1:M Relationship
(from A to B)

Data Model Basic Building Blocks

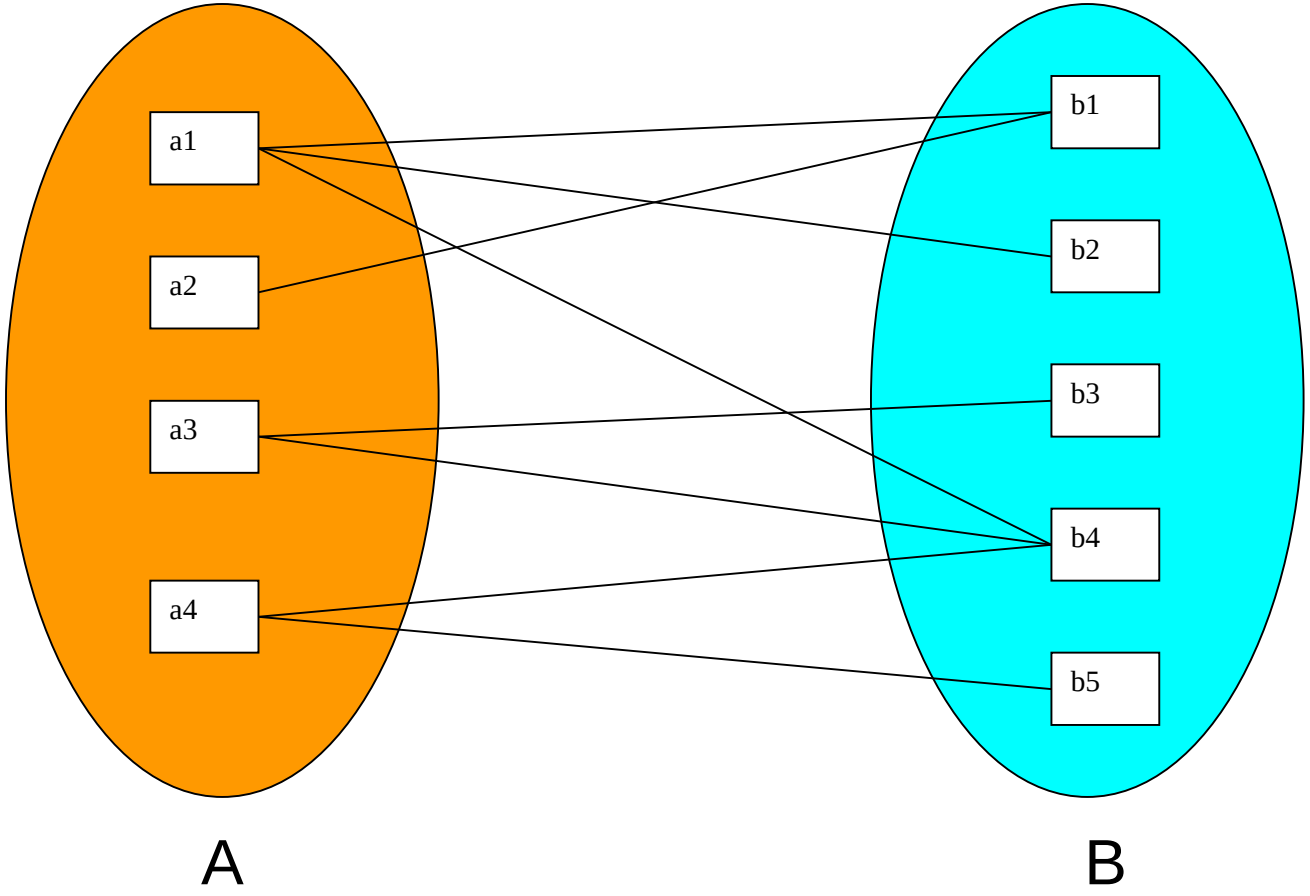


A

M:1 Relationship
(from B to A)

B

Data Model Basic Building Blocks



M:N Relationship

Data Model Basic Building Blocks

- The fourth and final basic building block is a constraint, which is a restriction placed on the data.
- Constraints are important because they help to ensure data integrity.
- Constraints are normally expressed in the form of rules. For example:
  - An employee's salary must have values between 45,000 and 200,000.
  - A student's GPA must be between 0.00 and 4.00.
  - Each class must have one and only one teacher.
- Constraints arise from business rules which are derived from a detailed description of how an organization operates.

Business Rules

- When database designers go about selecting or determining the entities, attributes, and relationships that will be used to build a data model, they start by gaining a thorough understanding of what types of data exit in an organization, how the data is used, and in what time frames they are used.
- Such data by itself does not yield the required understanding of the total business. From a database point of view, the collection of data becomes meaningful only when it reflects properly defined business rules.
- A business rule is a brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization.

Business Rules

- Properly written business rules are used to define entities, attributes, relationship, and constraints.
- Any time you see relationship statements such as "an agent can server many customers, and each customer can be served by only one agent," business rules are at work.
- To be effective, business rules must be easy to understand and widely disseminated to ensure that every person in the organization shares a common interpretation of the rules. Business rules describe, in simple language, the main and distinguishing characteristics of the data as *viewed by the company*.

Business Rules

- Examples of business rules are as follows:
  - A customer may generate many invoices.
  - An invoice is generated by only one customer.
  - A training session cannot be scheduled for fewer than 10 employees or for more than 30 employees.
- Note that the business rules establish entities, relationships, and constraints. For example, the first two business rules above establish two entities (Customer and Invoice) and a 1:M relationship between them. The third business rule above establishes a constraint (no fewer than 10 people and no more than 30 people, two entities (Employee and Training), and a relationship between Employee and Training.

Discovering Business Rules

- The main source of business rules are company managers, policy makers, department managers, and written documentation such as a company's procedures, standards, and operations manuals.
- A faster and more direct source of business rules is direct interviews with end users. Unfortunately, because perceptions differ, end users are sometimes a less reliable source when it comes to specifying business rules.
  - For example, a maintenance department mechanic might believe that any mechanic can initiate a maintenance procedure, when actually only mechanics with inspection authorization can perform such a task. Such a distinction might seem trivial, but it can have major legal ramifications.
  - Too often interview with several people who perform the same job may yield very different perceptions of what the job components are. While such a discovery might point to "management problems", this is of little help to the database designer.

Discovering Business Rules

- The process of identifying and documenting business rules is essential to database design for several reasons:
  - They help to standardize the organization's view of data.
  - They can be a communication tool between users and designers.
  - They allow the designer to understand the nature, role, and scope of the data.
  - They allow the designer to understand business processes.
  - They allow the designer to develop appropriate relationship participation rules and constraints and to create an accurate data model.
- Not all business rules can be modeled. For example, a business rule that specifies "no pilot can fly more than 10 hours within any 24 hour period" cannot be modeled. However, such a business rule can be enforced by application software.

Translating Business Rules Into Data Model Components

- Business rules set the stage for the proper identification of entities, attributes, relationships, and constraints. In the real world, names are used to identify object. If the business environment wants to keep track of the objects, there will be specific business rules for the objects.
- As a general rule, a noun in a business rule will translate to an entity in the model, and a verb (active or passive) that associates the nouns will translate into a relationship among the entities.
  - For example, the business rule "a customer may generate many invoices" contains two nouns (customer and invoices) and a verb (generate) that associates the nouns. From this business rule, you could deduce that:
    - Customer and invoice are objects of interest for the environment and should be represented by their respective entities.
    - There is a "generate" relationship between customer and invoice.

Translating Business Rules Into Data Model Components

- To properly identify the type of relationship, you should consider that relationships are bidirectional, that is, they go both ways.
  - For example, the business rule "a customer may generate many invoices" is complemented by the business rule "an invoice is generated by only one customer." In that case, the relationship is 1:M. Customers is the "1" side, and invoice is the "M" side.
- As a general rule, to properly identify the relationship type, you should ask two questions"
  - How many instances of B are related to one instance of A?
  - How many instances of A are related to one instance of B?
- You can assess the relationship between student and class by asking two questions: (1) in how many classes can one student enroll? (answer: many), and (2) how many students can enroll in one class? (answer: many). Thus the relationship is N:M.

Business Rules: Naming Conventions

- During the translation of business rules to data model components, you identify the entities, attributes, relationships, and constraints. This identification process includes naming the object in a way that make it unique and distinguishable from other objects in the problem domain. It is therefore important to pay special attention to how you name the objects you are discovering.

- Entity names should be descriptive of the objects in the business environment and use terminology that is familiar to the users. An attribute name should also be descriptive of the data represented by that attribute.

  – For example, if you are modeling a student entity and have their name as an attribute it should be something such as: STUDENT(name) and not OBJECT1(item1).

Business Rules: Naming Conventions

- It is also good practice to prefix the name of an attribute with the name (or an abbreviation) of the entity in which it occurs.
  - For example, in the CUSTOMER entity, the customer's credit limit might be called CUSTOMER_CREDIT_LIMIT, or perhaps CUS_CREDIT_LIMIT.
- The reason for this will become more apparent later on when you learn about the need to use common attributes to specify relationships between entities.
- The use of a proper naming convention will improve the data model's ability to facilitate communication among the designer, application programmer, and the end users.
- A proper naming convention can go a long way toward making your model self-documenting.

Evolution Of Data Models

- The quest for better data management has led to several different models that attempt to resolve the previous model's critical shortcomings and to provide solutions to ever-evolving data management needs.
- The various data models that have been developed, represent schools of thought as to what a database is, what it should do, the types of structures it should employ, and the technology that would be used to implement these structures.
- The table on the next page illustrates an overview of the major data models in roughly chronological order.

Evolution Of Data Models

GENERATION	TIME	DATA MODEL	EXAMPLES	COMMENTS
First	1960s–1970s	File system	VMS/VSAM	Used mainly on IBM mainframe systems Managed records, not relationships
Second	1970s	Hierarchical and network	IMS, ADABAS, IDS-II	Early database systems Navigational access
Third	Mid-1970s	Relational	DB2 Oracle MS SQL Server MySQL	Conceptual simplicity Entity relationship (ER) modeling and support for relational data modeling
Fourth	Mid-1980s	Object-oriented Object/ relational (O/R)	Versant Objectivity/DB DB2 UDB Oracle 11g	Object/relational supports object data types Star Schema support for data warehousing Web databases become common
Fifth	Mid-1990s	XML Hybrid DBMS	dbXML Tamino DB2 UDB Oracle 11g MS SQL Server	Unstructured data support O/R model supports XML documents Hybrid DBMS adds object front end to relational databases Support large databases (terabyte size)
Emerging Models: NoSQL	Late 2000s to present	Key-value store Column store	SimpleDB (Amazon) BigTable (Google) Cassandra (Apache)	Distributed, highly scalable High performance, fault tolerant Very large storage (petabytes) Suited for sparse data Proprietary API

The Hierarchical Model

- The hierarchical model was developed in the 1960s to manage large amounts of data from complex manufacturing projects, such as the Apollo rocket program (moon landing in 1969).
- The model's basic logical structure is represented as a tree. The tree contains levels, or segments. A segment is the equivalent of a file system's record type. Within the hierarchy, a higher layer is perceived as the parent of the segment directly beneath it, which is called the child.
- The hierarchical model depicts a set of 1:M relationships between a parent and its children segments. Each parent can have many children, but each child has only one parent.

The Network Model

- The network model was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard.
- In the network model, the user perceives the database as a collection of records in 1:M relationships. Unlike, the hierarchical model, the network model allows a record to have more than one parent.
- Although it is generally not used today, the network model defined many standards and concepts that are still in use today, such as the terms schema and sub-schema and definitions for a data manipulation language (DML) and data definition language (DDL).

The Relational Model

- The relational model was introduced in 1970 by E.F. Codd (working for IBM at the time), in his landmark paper "A Relational Model of Data for Large Shared Databanks" (*Communications of the ACM*, June 1970, pp.377-387).
- The relational model's foundation is a mathematical concept known as a relation. To avoid the complexity of abstract mathematical theory, think of a relation as a matrix composed of intersecting rows and columns. Each row is called a tuple. Each column represents an attribute.
- The relational model also describes a precise set of data manipulation constructs based on advanced mathematical concepts.

The Relational Model

- The relational model is implemented through a very sophisticated relational database management system (RDBMS). The RDBMS performs the same basic functions provided by the hierarchical and network DBMS systems, in addition to a host of other functions that make the relational data model easier to understand and implement.
- Arguably, the most important advantage of the RDBMS is its ability to hide the complexities of the relational model from the user.
- The RDBMS manages all of the physical details, while the user sees the relational database as a collection of tables in which the data are stored. The user can manipulate and query the data in a way that seems intuitive and logical.

The Relational Model

- Tables are related to each other through the sharing of a common attribute (a value in a column).
- For example, the next page illustrates how the CUSTOMER table might contain a sales agent's number that is also contained in the AGENT table.
- The common link between the CUSTOMER and AGENT tables enables you to match the customer to their sales agent, even though the customer data are stored in one table and the sales agent data are stored in another table.
  - For example, you can easily determine that customer Dunne's agent is Alex Alby because for customer Dunne, the CUSTOMER table's AGENT_CODE is 051, which matches the AGENT table's AGENT_CODE for Alex Alby.

Table name: AGENT (first six attributes)

Database name: Ch02_InsureCo

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

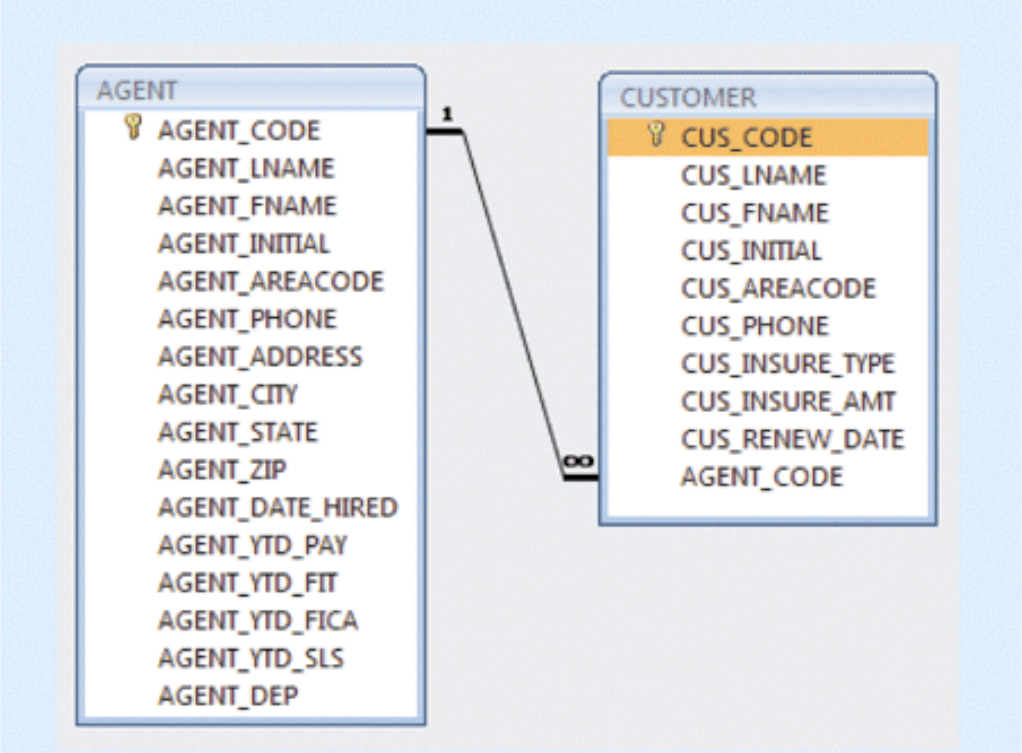
Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_INSURE_TYPE	CUS_INSURE_AMT	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	T1	100.00	05-Apr-2012	502
10011	Dunne	Leona	K	713	894-1238	T1	250.00	16-Jun-2012	501
10012	Smith	Kathy	W	615	894-2285	S2	150.00	29-Jan-2013	502
10013	Olowski	Paul	F	615	894-2180	S1	300.00	14-Oct-2012	502
10014	Orlando	Myron		615	222-1672	T1	100.00	28-Dec-2013	501
10015	O'Brian	Amy	B	713	442-3381	T2	850.00	22-Sep-2012	503
10016	Brown	James	G	615	297-1228	S1	120.00	25-Mar-2013	502
10017	Williams	George		615	290-2556	S1	250.00	17-Jul-2012	503
10018	Farriss	Anne	G	713	382-7185	T2	100.00	03-Dec-2012	501
10019	Smith	Olette	K	615	297-3809	S2	500.00	14-Mar-2013	503

The Relational Model

- The relational model provides a minimum level of controlled data redundancy to eliminate most of the redundancies commonly found in file systems.
- The relationship type (1:1, 1:M, M:N) is often shown in a relational schema. An example is shown on the next slide.
- A relational diagram is a representation of the relational database's entities, the attributes within those entities, and the relationships between those entities.
- The example on the next page illustrates a Microsoft Access representation of a 1:M relationship. The infinity symbol is used by Access to indicate the many side of a relationship.

The Relational Model



A Microsoft Access relational diagram illustrating a 1:M relationship from Agent to Customer. (One agent can have many customers, but a customer can have only one agent.)

The Relational Model

- A relational table stores a collection of related entities. In this respect, the relational database table resembles a file, but there is a critical difference between a table and a file. A table yields complete data and structural independence because it is purely a logical structure. How the data are physically stored in the database is of no concern to the user or the designer; the perception is what counts.
- It was this factor that led to the relational data model revolutionizing the database world.
- Another reason for the relational data model's rise to dominance is its powerful and flexible query language. Most relational database software uses SQL, which allows the user to specify what must be done without having to specify how it is done.

The Relational Model

- From an end-user perspective, and SQL-based relational database application involves three parts: a user interface, a set of tables stored in the database, and the SQL "engine".
  - The end-user interface. Basically, the interface allows the end user to interact with the data (by automatically generating SQL code). Each interface is a product of the software vendor's ides of meaningful interaction with the data.
  - A collection of tables stored in the database. In a relational database, all data are perceived to be stored in tables. The tables simply present the data to the end user in a way that is easy to understand. Each table is independent. Rows in different tables are related by common values in common attributes.
  - SQL engine. Largely hidden from the end user, the SQL engine executes all queries or data requests. The SQL engine is part of the DBMS software.

The Entity-Relationship Model

- The conceptual simplicity of the relational database technology triggered the demand for RDBMSs. In turn, the rapidly increasing requirements for transactions and information created the need for more complex database implementation structures, thus creating the need for more effective database design tools.
  - For example, building a skyscraper requires more detailed design activities than building a doghouse.
- Complex design activities require conceptual simplicity to yield successful results. Although the relational model was a huge improvement over the hierarchical and network models, it still lacked the features that would make it an effective database *design* tool.

The Entity-Relationship Model

- Because it is easier to examine structures graphically than to describe them in text, database designers prefer to use a graphical tool in which entities and their relationships are pictured.
- As a result, the entity-relationship (ER) model, or ERM, has become the most widely accepted standard for database modeling.
- Peter Chen first introduced the ER data model in 1976 and it quickly became popular because it complemented the relational data model concepts.
- The relational data model (a logical model) and the ER model (a conceptual model) are combined to provide the foundation for tightly structured database design.

The Entity-Relationship Model

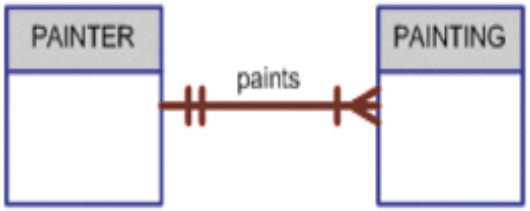
- The ER model is based on the following components:
  - Entity. An entity is represented in the ERD by a rectangle, also known as an entity box. The name of the entity, a noun, is written in the center of the rectangle. The entity name is generally written in all capital letters and singular in form: PAINTER rather than PAINTERS and EMPLOYEE rather than EMPLOYEES. When applying the ERD to the relational model, an entity is mapped to a relational table. Each row in the relational table is known as an entity instance or entity occurrence in the ER model.

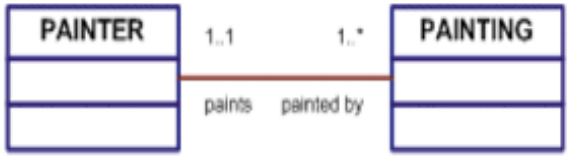
    Each entity consists of a set of attributes that describes particular characteristics of the entity. We'll see how to include these shortly.

  - Relationships. Relationships describe associations among data. Most relationships describe associations between two entities. The ER model uses the term connectivity or cardinality to label the relationship types.
- The next slide illustrates the different types of relationships using three different ER notations.

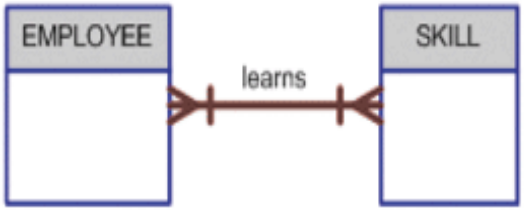
Chen Notation *Crow's Foot Notation* *UML Notation*

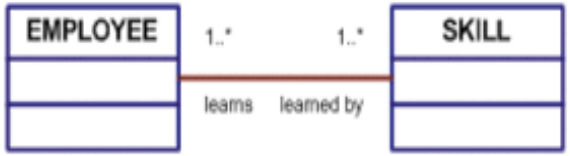
A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.



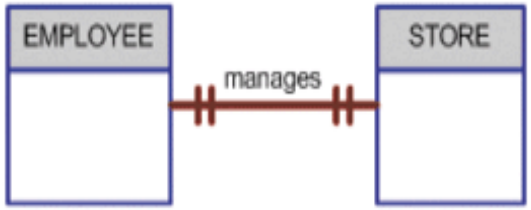


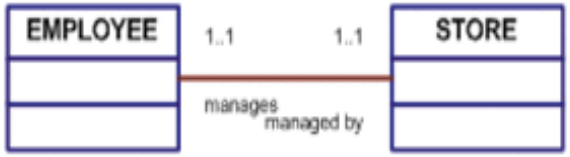
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.





A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.

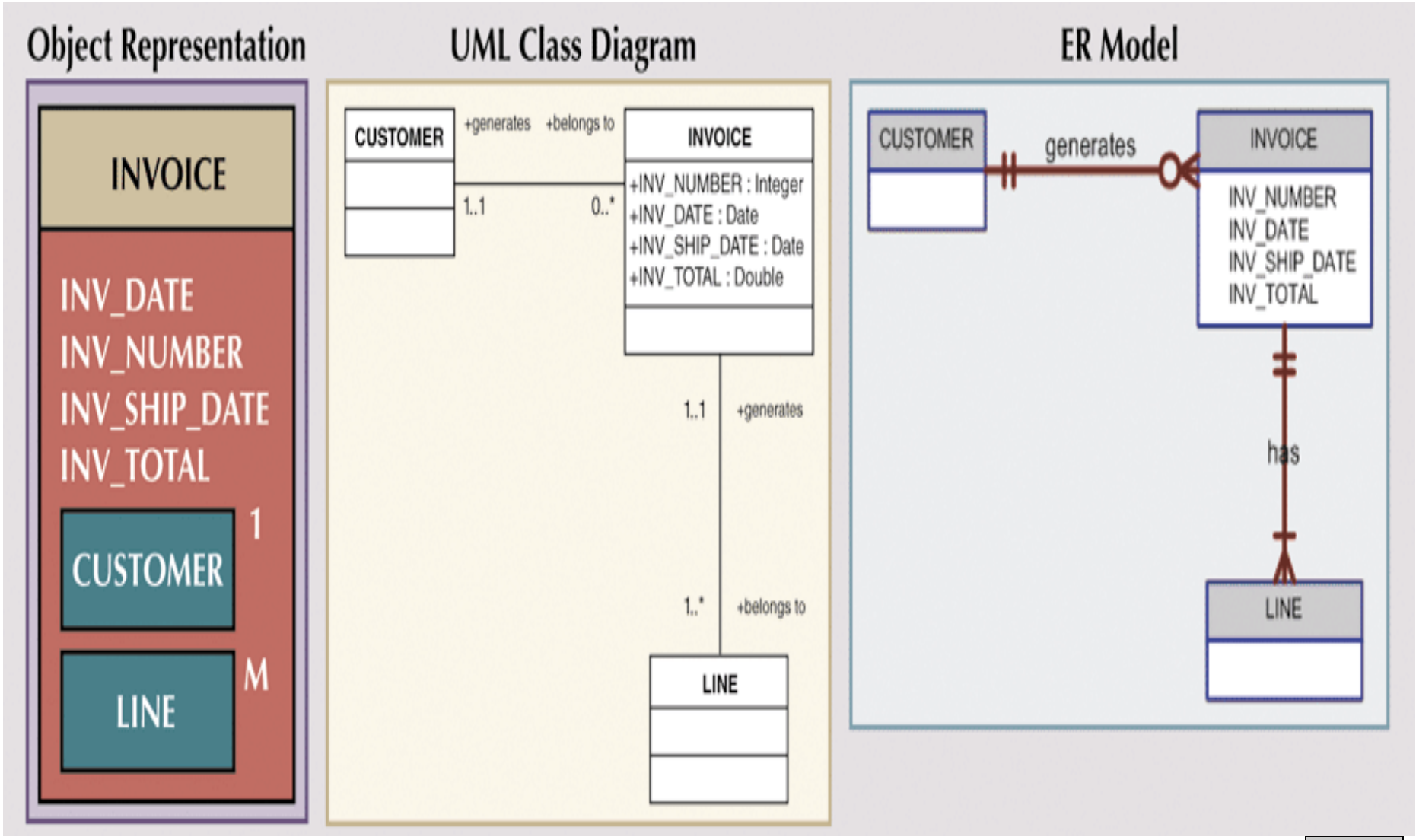




The Object-Oriented (OO) Model

- Increasingly complex real-world problems demonstrated a need for a data model that more closely represented the real world.
- In the object-oriented data model (OODM), both data and their relationships are contained in a single structure known as an object. In turn, the OODM is the basis for the object-oriented database management system (OODBMS).
- An OODM reflects a very different way to define and use entities. Like the relational model's entity, an object is described by its factual content. But, quite unlike an entity, an object includes information about relationships between the facts within the object, as well as information about its relationship with other objects. Therefore, facts within the object are given greater meaning. The OODM is a semantic data model.
- An OODM example is shown on the next slide.

Evolution Of Data Models



Object/Relational And XML Models

- Facing the demand to support more complex data representations, the relational model's main vendors evolved the model further and created the extended relational data model (ERDM).
- The ERDM adds many of the OO model's features within the inherently simpler relational database structure. The ERDM gave rise to a new generation of relational databases that support OO features such as objects, extensible data types based on classes and inheritance.
- DBMS based on the ERDM are often described as an object/relational database management system (O/R DBMS).
- Today, most relational database products can be classified as object/relational, and they represent the dominate market share of OLTP and OLAP database applications.

Object/Relational And XML Models

- The success of the O/R DBMS can be attributed to the model's conceptual simplicity, data integrity, easy-to-use query language, high transaction performance, high availability, security, scalability, and expandability.
- In contrast, the OO DBMS is popular in niche markets such as CAD/CAM systems, geographic information systems (GIS), telecommunications, and multimedia, which require support for more complex objects.
- From the start, the OO and relational models were developed in response to different problems. The OO model was created to address very specific engineering needs, not the wide-ranging needs of general data management tasks.

Object/Relational And XML Models

- Given its focus on a smaller set of problem areas, it is not surprising that the OO market has not grown as rapidly as the relational data model market.
- Although relational and object/relational databases address most current data processing needs, a new generation of databases has emerged to address some very specific challenges found in some Internet-era organizations.

Emerging Data Models: Big Data And NoSQL

- Deriving usable business information from the mountains of Web data that organizations have accumulated over the years has become an imperative need.
- Web data in the form of browsing patterns, purchasing histories, customer preferences, behavior patterns, and social media data from sources such as Facebook, Twitter, and LinkedIn have inundated organizations with combinations of structured and unstructured data.
- According to many studies, the rapid pace of data growth is the top challenge for many organizations, with system performance and scalability as the next biggest challenges.
- Today's IT managers are constantly balancing the need to manage this rapidly growing data with shrinking budgets.

Emerging Data Models: Big Data And NoSQL

- The need to manage and leverage all of these converging trends (rapid data growth, performance, scalability, and lower costs) has triggered a phenomenon called "Big Data."
- Big Data refers to a movement to find new and better ways to manage large amounts of Web-generated data and derive business insight from it, while simultaneously providing high performance and scalability at a reasonable cost.

Unstructured data might be word-processing documents, emails, web pages, and diagrams. XML is the de facto standard for the efficient and effective exchange of structured, semi-structured, and unstructured data. XML databases were created to handle the unstructured data within the XML format. Both the relational data model and the O/R data model are easily extended to support XML.

Emerging Data Models: Big Data And NoSQL

- The big problem is that the relational data model does not always match the needs of organizations with Big Data challenges:
  - It is not always possible to fit unstructured, social media data into the conventional relational structure of rows and columns.
  - Adding millions of rows of multi-format (structured and unstructured) data on a daily basis will inevitably lead to the need for more storage, processing power, and sophisticated data analysis tools that may not be available in the relational environment.
  - Generally speaking, the type of high-volume implementation required in the RDBMS environment for the Big Data problem comes with a hefty price tag for expanding hardware, storage, and software licenses.
  - Data analysis based on OLAP tools has proven to be very successful in relational environments with highly structured data. However, mining for usable data in the vast amounts of unstructured data collected from Web sources requires a different approach.

Emerging Data Models: Big Data And NoSQL

- There is no "one-size-fits-all" cure to data management.
- For some organizations, creating a highly scalable, fault-tolerant infrastructure for Big Data analysis could prove to be a matter of business survival. The business world has many examples of companies that leverage technology to gain a competitive advantage, and others that miss it.
- Consider the current business landscape if:
  - MySpace had responded to Facebook's challenge in time.
  - Blockbuster had reacted to the Netflix business model sooner.
  - Barnes & Noble had developed a viable Internet strategy before Amazon.
- Therefore, it is not surprising that some organizations are turning to NoSQL databases to mine the wealth of information in Web data.

NoSQL Databases

- Every time you search for a product on Amazon, send a message to friends in Facebook, watch a video on YouTube, or search for directions in Google Maps, you are using a NoSQL database.
- As with any new technology, the term NoSQL can be loosely applied to many different types of technologies.
- However, in general, NoSQL tends to refer to a new generation of databases that address the specific challenges of the Big Data era and have the following general characteristics:
  - Not based on the relational data model.
  - Support distributed database architectures.
  - Provides high scalability, high availability, and fault tolerance.
  - Supports very large amounts of sparse data.
  - Geared toward performance rather than transaction consistency.

NoSQL Databases

NOTE:

Does all of this mean that relational databases don't have a place in organizations with Big Data challenges?

No, relational databases remain the preferred and dominant databases to support most day-to-day transactions and structured data analytics needs. Each DBMS technology has its areas of applications, and the best approach is to use the best tool for the job. In perspective, object/relational databases server 98% of market needs. For the remaining 2%, NoSQL databases are an option.

NoSQL Databases

- While NoSQL databases are not based on the relational data model, there is no standard NoSQL model.
- Many different data models are grouped under the NoSQL umbrella, from document databases to graph stores, column stores, and key-value stores.
- It is still too early to know, which, if any of these data models will survive and grow to become the dominant force in the database arena.
- The early success of products such as Amazon's SimpleDB, Google's BigTable, and Apache's Cassandra point to the key-value stores and column stores as the early leaders.
- The word stores indicates that these models permanently store data in secondary storage, just like any other database.

NoSQL Databases

- The key-value data model is based on a structure composed of two data elements: a key and a value, in which every key has a corresponding value or set of values.
- The key value model is also referred to as the attribute-value or associative data model.
- Consider the example shown on the next slide which illustrates a small truck-driving company called Trucks-R-Us. Each of its three drivers has one or more certifications and other general information.
- Based on this example we can define the following important points:

NoSQL Databases

Data stored using traditional relational model

DID	CERT1	CERT2	CERT3	DOB	LICTYPE
2732	80		90	1/24/1962	P
2946		92	90	4/11/1970	
3650	86			6/27/1968	C

Data stored using key-value model

DID	KEY	VALUE
2732	CERT1	80
2732	CERT3	90
2732	DOB	1/24/1962
2732	LICTYPE	P
2946	CERT2	92
2946	CERT3	90
2946	DOB	4/11/1970
3650	CERT1	86
3650	DOB	6/27/1968
36500	LICTYPE	C

- In the relational model:
  - Each row represents one entity instance
  - Each column represents one attribute of the entity
  - The values in a column are of the same data type
- In the key-value model:
  - Each row represents one attribute/value of one entity instance
  - The “key” column could represent any entity’s attribute
  - The values in the “value” column could be of any data type and therefore it is generally assigned a long string data type

NoSQL Databases

- In the relational model, every row represents a single entity occurrence and every column represents an attribute of the entity occurrence Each column has a defined data type.
- In the key-value model, each row represents one attribute of one entity instance. The "key" column points to an attribute and the "value" column contains the actual value for the attribute.
- The data type of the "value" column is generally a long string to accommodate the variety of actual data types of the values placed in the column.
- To add a new entity attribute in the relational model, you need to modify the table definition (schema modification). To add a new attribute in the key-value model, you add a row to the key-value store, which is why it is said to be "schema-less".
- NoSQL databases do not store or enforce relationships among entities. The programmer is required to manage the relationships in the code. Also, all data and integrity validations must be done in the code. (Some implementations have been extended to include metadata support.)

NoSQL Databases

- No SQL databases use their own native application programming interface (API) with simple data access commands, such as put, read, and delete. Because there is no declarative SQL-like syntax to retrieve data, the program code must take care of retrieving related data in the correct way.
- Indexing and searches can be difficult. Because the "value" column in the key-value model could contain many different data types, it is often difficult to create indices on the data. At the same time, searches can become very complex.

- You could use the key-value modeling technique for any situation in which the attributes are numerous, but the actual data values are scarce. The key-value model is not exclusive of NoSQL databases; actually, key-value structures could exist inside a relational database. However, because of the problems with maintaining relationships and integrity within the data, and the increased complexity of even simple queries, key-value structures would be a poor design for most structured business data.

NoSQL Databases

- Several NoSQL database implementations, such as Google's BigTable and Apache's Cassandra, have extended the key-value model to group multiple key-value sets into column families or column stores.

- In addition, such implementations support features such as versioning using a date/time stamp. For example, BigTable stores data in the syntax of [row, column, time, value], where row, column and value are string data types and time is a date/time data type. The key used to access the data is composed of (row, column, time), where time can be left blank to indicate the most recent stored value.

NoSQL Databases

- One of the big advantages of NoSQL databases is that they generally use a distributed architecture.
- NoSQL databases can handle very large volumes of data. In particular, they are suited for sparse data.
- Sparse data occurs when the number of attributes is very large, but the number of actual data instances is low. Using the truck driving company as an example, drivers can take any certification exam, but they are not required to take them all. In this case there were three drivers and three possible certificates for each driver, so there will be nine possible data points (we illustrated only 4). Extrapolate this to a hospital with 50,000 patients and more than 2500 possible medical tests that could be performed. We would not expect to see 50000 x 2500 data points, in reality we would see far less.

NoSQL Databases

- Most NoSQL databases are geared toward performance rather than transaction consistency.
- Enforcing data consistency on a distributed database is a very difficult problem. Distributed databases make copies of data elements at multiple nodes to ensure high availability and fault tolerance. Updating values and requiring consistency amongst all copies is a very tough problem to solve. NoSQL database sacrifice this consistency to attain high levels of performance.
- Some NoSQL databases provide a feature called eventual consistency, which means that updates to data will propagate through the system and eventually all copies will be consistent. With eventual consistency, data are not guaranteed to be consistent across all copies immediately after an update.

Evolution Of Data Models