C175 - Data Management Foundations

Data Management Foundations (Western Governors University)

C175 Data Management Foundations Notes

Lesson 1 - Introduction to Databases, Information, and Data

Why Databases?

- Databases are the best way to store and manage data as persistent and shareable in a secure way.

File Structures

- Prior to Database Management Systems, information was initially stored in file-based systems. Small amounts of information are easy to retrieve from a file-based system.
- The speed that information can be retrieved from a file depends on the structure of the file, and on how the data is organized in that structure.
- **Flat files**
    - Files having no internal hierarchy
- **Heap files**
    - Files containing an unsorted set of records that are uniquely identified by a record id, which allows them to be inserted or delete using that id.
- **Index files**
    - Files that store a list of lookup field values from a data file — along with the location (address) in the data file of the corresponding record.
    - Because the lookup field is much smaller than the entire record, the entire index will usually fit in main memory for quick look up.
    - Once the address of the record is obtained from the index, the entire record can be directly accessed from the data file instead of reading in the entire data file.
- **Hashed files**
    - Files are encrypted using hash functions that convert data consisting of various formats into numeric values. This allows for faster data lookup without the use of an index file.

Data Versus Information

- **Data**
    - Consists of raw facts.
    - The raw facts have not yet been processed to reveal their meaning.
- **Information**
    - The result of processing raw data to reveal its meaning.
    - Can be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling.
- Raw data must be properly formatted for storage, processing, and presentation.
- **Knowledge**
    - The body of information and facts about a specific subject
    - Implies familiarity, awareness, and understanding of information as it applies to an environment.
    - New knowledge can be derived from old knowledge.
- Data constitutes the building blocks of information
- Information is produced by processing data.
- Information is used to reveal the meaning of data.
- Accurate, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.
- **Data Management**
    - a discipline that focuses on the proper generation, storage, and retrieval of data.

Introducing the Database

- **Database**
    - A shared, integrated computer structure that stores a collection of the following:
        - End-user data
        - **Metadata**, or data about data.
            - Describes the data characteristics and the set of relationships that links the data found within the database.
            - Examples include information such as the name of each data element, the types of values, and whether the data element can be left empty.
- **Database management system (DMBS)**
    - A collection of programs that manages the database structure and controls access to the data stored in the database.

Role and Advantages of the DBMS

- The DBMS serves as an intermediary between user and database.
- The database structure is a collection of files, which must be accessed through DBMS.
- DBMS receives all requests and translates them into operations required to fulfill those requests.
- Hides database's internal complexity from the application programs and users.
- Enables the data *to be shared* among multiple applications or users.
- *Integrates* the many users' views of the data into a single all-encompassing data repository.
- Provides these advantages:
    - **Improved data sharing.**
        - The DBMS helps create an environment in which end users have better access to more and better-managed data.
    - **Improved data security**
        - Provides a framework for better enforcement of data privacy and security policies.
    - **Better data integration**
        - Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture.
    - **Minimized data inconsistency**
        - Data inconsistency exists when different versions of the same data appears in different places.
        - The probability of data inconsistency is greatly reduced in a properly designed database.
    - **Improved data access**

        - Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are made.
        - The quality of the information generates relies on the data quality.
        - **Data quality**
            - A comprehensive approach to promoting the accuracy, validity, and timeliness of the data.
            - A DBMS provides a framework to facilitate data quality initiatives.
    - **Increased end-user productivity**
        - Availability of data, combined with tools that transform data into information, empowers end users to make informed decisions.

Types of Databases

- **Number of users**
    - **Single-user database**
        - Supports only one user at a time
        - **Desktop database**
            - A single-user database that runs on a personal computer.
    - **Multiuser database**
        - Supports multiple users at the same time.
        - **Workgroup database**
            - A multiuser database with a relatively small number of users (usually fewer than 50), or a specific department within an organization.
        - **Enterprise database**
            - A database used by an entire organization and supports many users (more than 50) across many departments.
- **Location**
    - **Centralized database**
        - Supports data located at a single site.
    - **Distributed database**
        - Supports data distributed across several different sites.
    - Requires a well-defined infrastructure to implement and operate the database.
    - The use of cloud databases has been growing in popularity.
    - **Cloud database**
        - A database created and maintained using cloud data services, such as Microsoft Azure or Amazon AWS.
- **Type of data**
    - **General purpose database**
        - Contains a wide variety of data used in multiple disciplines
    - **Discipline-specific database**
        - Contains data focused on specific subject areas, mainly used for academic or research purposes within a small set of disciplines.

- **Usage and time sensitivity of information**
    - **Operational database / online transaction processing (OLTP) database / transactional database / production database**
        - A database that is designed primarily to support a company's day-to-day operations such as transactions.
    - **Analytical database**
        - Focuses primarily on storing historical data and business metrics used exclusively for tactical or strategic decision making.
        - Allow the end user to perform advanced analysis of business data using sophisticated tools.
        - Comprise two main components:
            - **Data warehouse**: A specialized database that stores data in a format optimized for decision support. Contains historical data obtained from the operational database as well as data from other external sources.
            - **Online analytical processing (OLAP)**: a set of tools that work together to provide an advanced data analysis environment for retrieving, processing, and modeling data from the data warehouse.
                - **Business intelligence**: Describes a comprehensive approach to capture and process business data with the purpose of generating information to support business decision making.
- **Structure**
    - **Unstructured data**
        - Exists in its original raw state. Does not lend itself to the processing that yields information.
    - **Structured data**
        - The result of formatting unstructured data to facilitate storage, use, and the generation of information.
        - Structure is applied based on the type of processing that you intend to perform on the data.
    - **Semistructured data**
        - Has already been processed to some extent
    - **XML database**
        - Supports the storage and management of semi-structured XML data.
- **NoSQL (Not only SQL)**
    - Used to describe a new generation of database management systems that is not based on the traditional database model.
    - Designed to handle unprecedented volume of data, variety of data types and structures, and velocity of data operations.

Why Database Design is Important

- Users typically lack proper data-modeling and database design skills.
- **Database design**
    - Refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data.
    - Structure must be designed carefully, even a good DBMS will perform poorly with a badly designed database.
- Designing appropriate data repositories of integrated information using the two-dimensional table structures found in most databases is a process of decomposition.
- Integrated data must be decomposed into its constituent parts, with each part stored in its own table.

Evolution of File System Data Processing

- **Data**
    - Raw facts such as a telephone number, customer name.
    - Has little meaning unless it has been organized in some logical manner.
- **Field**
    - Contains all the information within the table appropriate to a particular entity. It is a data structure for a single piece of data.
- **Record**
    - A logically connected set of one or more fields that describes a person, place, or thing.
- **File**
    - A collection of related records. Stores data, information, settings, or commands that are used in computation. It is a collection of related records.
- Early computerized file systems suffered from problems centered on having many data files that contained related and overlapping data, with no means of controlling or managing the data consistently across all of the files.

Problems with File System Data Processing

- **Lengthy development times**
    - Even simple data-retrieval tasks require extensive programming. With older file systems, programmers had to specify what must be done and how to do it.
- **Difficulty of getting quick answers**
    - The need to write programs to produce even the simplest reports makes ad hoc queries impossible.
- **Complex system administration**
    - System administration becomes more difficult as the number of files in the system expands.
- **Lack of security and limited data sharing**
    - Security and data-sharing features are difficult to program and omitted from a file system environment.
- **Extensive programming**
    - Making changes to an existing file structure can be difficult in a file system environment.

Structural and Data Dependence

- **Structural dependence**
    - Access to a file is dependent on its structure. Given changes to a file's structure, previous programs will not work. All file system programs must be modified to conform to the new file structure.
- **Structural independence**
    - Exists when you can change the file structure without affecting the application's ability to access the data.
- **Data dependence**
    - Changes in the characteristics of data, such as changing a field from integer to decimal, necessitating a change in the file system applications.
- **Data independence**
    - Exists when you can change the data storage characteristics without affecting the program's ability to access the data.
- **Logical data format**
    - How the human being views the data
- **Physical data format**
    - How the computer must work with the data.
- Data dependence makes the file system extremely cumbersome from the point of view of a programmer and database manager.

Data Redundancy

- The file system's structure makes it difficult to combine data from multiple sources, and its lack of security renders the file system vulnerable to security breaches.
- The organizational structure promotes the storage of the same basic data in different locations (a.k.a **islands of information** for such scattered data locations)
- **Data redundancy**
    - Exists when the same data is stored unnecessarily at different places.
- Uncontrolled data redundancy leads to the following:
    - **Poor data security**
        - Having multiple copies of data increases the chances for a copy of the data to be susceptible to unauthorized access.
    - **Data inconsistency**
        - Exists when different and conflicting versions of the same data appear in different places.
    - **Data entry errors**
        - Likely to occur when complex entries are made in several different files or recur frequently in one or more files.
    - **Data integrity problems**
        - It is possible to enter a nonexistent record into a file. Data entry errors can yield data integrity problems.

Data Anomalies

- **Data anomaly**
    - Develops when not all of the required changes in redundant data are made successfully.
    - **Update anomalies**
        - In a large file system, a change might occur in hundreds or thousands of records. The potential for data inconsistencies is great.
    - **Insertion anomalies**
        - If only the CUSTOMER file existed (figure 1.7) and you needed to add a new agent, you would also add a dummy customer data entry to reflect the new agent's addition.
    - **Deletion Anomalies**
        - If you delete all of an agent's customers, you will also delete an agent's data. This is not desirable.

Database Systems

- Consists of logically related data stored in a single logical data repository.
- Current generation of DBMS software stores not only the data structures, but also the relationships between those structures and the access paths to those structures.
- DBMS software also takes care of defining storing and managing all required access paths to those components.

The Database System Environment

- **Database system**
    - Refers to an organization of components that define and regulate the collection storage management and use of data within a database environment.
- Composed of five major components:
    - **Hardware**
        - All of the system's physical devices, including computers, storage devices, network devices, etc.
    - **Software**
        - The most readily identified software is the DBMS itself, there are three types of software needed to make the database system function fully:
            - **Operating system**
                - Manages all hardware components and makes it possible for all other software to run on the computers.
            - **DBMS software**
                - Manages the database within the database system
            - **Application programs and utility software**
                - used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place.
                - Most commonly used to access data within the database to generate reports, tabulations, and other information.
    - **People**
        - **System administrators**
            - Oversee the database system's general operations
        - **Database administrators**
            - Also known as DBAs, manage the DBMS and ensure the database is functioning properly.
        - **Database designers**
            - Design the database structure. They are the database architects.
        - **System analysts and programmers**
            - Design and implement the application programs

    - Design and create the data-entry screens, reports, and procedures through which end users access and manipulate the database's data.
  - **End users**
    - The people who use the application programs to run the organization's daily operations.
- **Procedures**
  - The instructions and rules that govern the design and use of the database system. Enforce the standards by which business is conducted within the organization and with customers.
- **Data**
  - The collection of facts stored in the database.

DBMS Functions

- **Data dictionary management**
  - The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary.
  - All programs that access the data in the database work through the DBMS.
  - The DBMS uses the **data dictionary** to look up the required data component structures and relationships.
  - The DBMS provides data abstraction, and removes structural and data dependence from the system.
- **Data storage management**
  - The DBMS creates and manages the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics.
  - DBMS provides storage for data and for related data-entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture formats, etc.
  - Data storage management is also important for database performance tuning.
  - **Performance tuning**
    - Relates to the activities that make the database perform more efficiently in terms of storage and access speed.
    - The DBMS stores the database in multiple physical data files.
- **Data transformation and presentation**
  - DBMS transforms entered data to conform to required data structures.
  - Relieves you of the chore of distinguishing between the logical data format and the physical data format.
- **Security Management**
  - The DBMS creates a security system that enforces user security and data privacy.

    - Users may be authenticated to the DBMS through a username and password or through biometric authentication such as fingerprint scan.
- **Multiuser access control**
    - To provide data integrity and data consistency, the DBMS uses algorithms to ensure multiple users can access the database concurrently without compromising its integrity.
- **Backup and recovery management**
    - The DBMS provides backup and data recovery to ensure data safety and integrity.
    - Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure.
- **Data integrity management**
    - The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency.
    - The data relationships stored in the data dictionary are used to enforce data integrity.
- **Database access languages and application programming interfaces**
    - The DBMS provides data access through a query language.
    - **Query language**
        - A nonprocedural language, on that lets the user specify what must be done without having to specify how.
        - **Structured Query Language (SQL)** is the de facto query language and data access standard supported by the majority of DBMS vendors.
- **Database communication interfaces**
    - Accepts end-user requests via multiple different network environments.

Lesson 2 - Data Modeling

Data Modeling and Data models

- **Data modeling**
    - The first step in designing a database, refers to the process of creating a specific data model for a determined problem domain.
- **Data model**
    - A simple representation, usually graphical, of more complex real-world data structures.
    - A data model represents data structures and their characteristics, relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.

The Importance of Data Models

- Data models can facilitate interaction among the designer, the applications programmer, and the end user. A well-developed data model can even foster improved understanding of the organization for which the database design is developed.
- Data models are communication tool.

Data Model Basic Building Blocks

- The basic building blocks of all data models are entities, attributes, relationships, and constraints.
- **Entity**
    - A person, place, thing, or event about which data will be collected and stored.
    - Represents a particular type of object in the real word, unique and distinct.
- **Attribute**
    - A characteristic of an entity.
- **Relationship**
    - Describes an association among entities.
    - Three types of relationships:
        - **One-to-many (1:M or 1..*) relationship**
            - A painter creates many different paintings, but each is painted by only one painter.
        - **Many-to-many (M:N or *..*) relationship**
            - An employee may learn many job skills and each job skill may be learned by many employees.
        - **One-to-one (1:1 or 1..1) relationship**

            - A retail company’s management structure may require each store be managed by a single employee, and each manager manages only a single store.
- **Constraint**
    - A restriction placed on the data.
    - E.g. a student’s GPA must be between 0.00 and 4.00.

Business Rules

- **Business rule**
    - A brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization.
- Business rules derived from a detailed description of an organization’s operations help to create and enforce actions within that organization’s environment.
- Properly written business rules are used to define entities, attributes, relationships, and constraints.

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.

Translating Business Rules into Data Model Components

- Business rules set the stage for the proper identification of entities, attributes, relationships, and constraints.
- In a business rule...
    - A noun will translate into an entity
    - A verb that associates the nouns will translate into a relationship among entities

Naming Conventions

- Entity names should be description of the objects in the business environment and use terminology that is familiar to the users.
- An attribute name should be descriptive of the data represented by that attribute.
- It is also a good practice to prefix the name of an attribute with the name or abbreviation of the entity in which it occurs.

The Evolution of Data Models

Hierarchical and Network Models

- **Hierarchical model**
    - Developed in the 1960s to manage large amounts of data for complex manufacturing projects.
    - Represented by an upside-down tree, containing segments.
    - **Segment**
        - 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, called the child.
- **Network model**
    - 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 network database as a collection of records in 1:M relationships.
    - Network model allows a record to have more than one parent.
        - **Schema**
            - The conceptual organization of the entire database as viewed by the database administrator
        - **Subschema**
            - Defines the portion of the database "seen" by the application programs that actually produce the desired information from the data within the database.
        - **Data manipulation language (DML)**
            - Defines the environment in which data can be managed and is used to work with the data in the database
        - **Data definition language (DDL)**
            - Enables the database administrator to define the schema components.

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 modeling and support for relational data modeling
Fourth	Mid-1980s	Object-oriented Object/relational	Versant, Objectivity/DB, DB2, UDB, Oracle 12c	Object/relational supports object data types Star scheme support for data warehousing Web databases become common
Fifth	Mid-1990s	XML Hybrid, DBMS	dbXML, Tamino, DB2 UDB, Oracle 12c, MS SQL Server	Unstructured data support, O/R model supports XML documents Hybrid DBMS adds object front end to relational databases. Supports large databases
Emerging Models: NoSQL	Early 2000s to present	Key-value store Column Store	SimpleDB, BigTable, Cassandra, MongoDB, Riak	Distributed, highly scalable, high performance, fault tolerant, very large storage, suited for sparse data, proprietary API

The Relational Model

- **Relational model**
    - Introduced in 1970, represented a major breakthrough for bother users and designers
- **Relational database management system (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.
    - Ability to hide the complexities of the relational model from the user.
- **Relational diagram**
    - A representation of the relational database's entities, the attributes within those entities, and the relationships between those entities.
- SQL-based relational database application involves three parts:
    - The end-user interface
    - A collection of tables stored in the database
    - SQL engine
- **Entity relationship (ER) Model**
    - Graphical representation of entities and their relationships in a database structure.
    - Represented in an **entity relationship diagram (ERD)** which uses graphical representations to model database components.

Object Oriented Model

- **Object-oriented data model (OODM)**
    - Both data and its relationships are contained in a single structure known as an object.
- Unlike an entity, an object includes information about relationships between the facts within the object as well as information about its relationships with other objects.
- Facts within the object are given greater meaning. The OODM is said to be a **semantic data model**.
- Subsequence OODM has allowed an object also to contain all operations that can be performed on it.

Object/Relational and XML

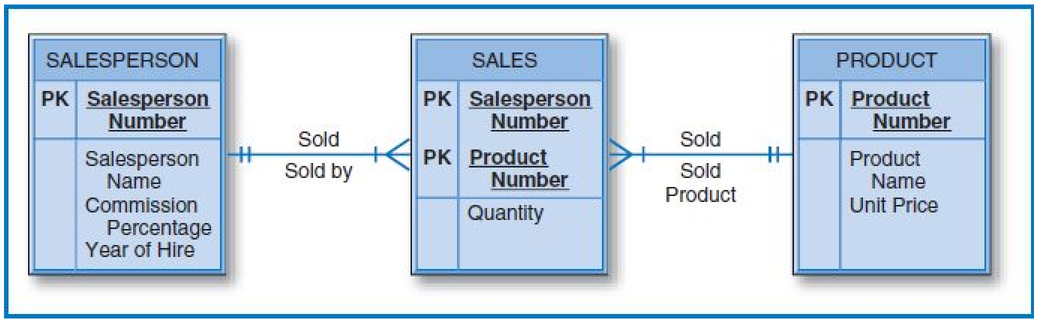
- **Extended relational data model (ERDM)**
    - Adds many of the OO model's features within the inherently simpler relational database structure. Often described as an **object/relational database management system**.
- Most relational database products can be classified as object/relational.

Emerging Data Models: Big Data and NoSQL

- **Big data**
    - Refers to a movement to find new and better ways to manage large amounts of web and sensor-generated data and derive business insights from it, while simultaneously providing high performance and scalability at a reasonable cost.
- Three basic characteristics of big data databases:
    - Volume, velocity, variety
- **Hadoop**
    - A Java-based, open source, high speed, fault-tolerant distributed storage and computational framework.
    - Uses low-cost hardware to create clusters of thousands of computer nodes to store and process data.
- **Hadoop Distributed File System (HDFS)**
    - A highly distributed, fault-tolerant file storage system designed to manage large amounts of data at high speeds.
    - HDFS uses the write-once, read many model. Once data is written, it cannot be modified.
    - Uses three types of nodes:
        - **Name node**
            - stores all the metadata about the file system
        - **Data node**
            - Stores fixed-size data blocks
        - **Client node**
            - Acts as the interface between the user application and the HDFS
- **MapReduce**
    - An open source API that provides fast data analytics services.
    - Distributes the processing of data among thousands of nodes in parallel.
    - Works with structured and non-structured data.
- **NoSQL**
    - A large scale distributed database system that stores structured and unstructured data in efficient ways.
    - They are not based on the relational model and SQL.
    - They support distributed database architectures.
    - They provide high availability, high availability, and fault tolerance.
    - Support very large amounts of sparse data.
    - Geared toward performance rather than transaction consistency.
    - **key-value data model**
        - Based on a structure composed of two data elements, a key and a value.
        - Do not store or enforce relationships among entities. The programmer is required to manage relationships in the program code.
    - **sparse data**

- for cases in which the number of attributes is very large but the number of actual data instances is low.

More About Many-to-Many Relationships

- **Intersection data**
    - Describes the combination of or the association between two entities, such as a salesperson and a particular product, where quantity sold is an attribute that falls at the intersection of both.
- **Associative entity**
    - Many-to-many relationships can be treated similarly to entities since they can both have attributes... and hence can be represented within E-R diagrams.
    - The many-to-many relationship Sells can be converted into the associative entity SALES.
    - 

- The unique identifier of the many-to-many relationship or the associative entity is the combination of the unique identifiers of the two entities in the many-to-many relationship.

Degrees of Data Abstraction

Data Model	**Data Independence**	**Structural Independence**	**Advantages**	**Disadvantages**
Hierarchical	Yes	No	Promotes data sharing. Parent/child relationship promotes conceptual simplicity. Database security is provided and enforced by DBMS. Parent/child relationship promotes data integrity. Is efficient with 1:M relationships.	Complex implementation requires knowledge of physical data storage characteristics. Navigational system yields complex application development. Changes in structure require changes in all applications. There are no multiparent or M:N relationships.

				No data definition or data manipulation language in DBMS. Lack of standards.
Network	Yes	No	Conceptual simplicity is at least equal to the hierarchical model. Handles M:N and multiparent relationships. Data access is more flexible. Data owner/member relationship promotes data integrity. Conformance to standards. Includes DDL and DML.	System complexity limits efficiency — still a navigational system. Navigational system yields complex implementation, application development, and management. Structural changes require changes in all applications.
Relational	Yes	Yes	Structural independence promoted by use of independent tables. Tabular view improves conceptual simplicity. Ad hoc query capability based on SQL. Powerful RDBMS isolates end user from physical level detail.	RDBMS requires substantial hardware and system software overhead. Conceptual simplicity gives untrained people tools to use a good system poorly. May promote islands of information problems.
Entity Relationship	Yes	Yes	Visual modeling yields exceptional conceptual simplicity. Visual representation makes an effective communication tool. Integrated with relational model.	Limited constraint representation. Limited relationship representation. No data manipulation language. Loss of information occurs when attributes are removed from entities to avoid crowded displays.
Object oriented	Yes	Yes	Semantic content is added. Visual representation includes semantic	Slow development of standards. Complex navigation system.

			content. Inheritance promotes data integrity.	Steep learning curve. High system overhead slows transactions.
NoSQL	Yes	Yes	High scalability, availability, and fault tolerance are provided. Uses low cost commodity hardware. Supports Big Data. Key-value model improves storage efficiency.	Complex programming is required. No relationship support. No transaction integrity report. Eventually consistent model.

Lesson 3 - The Relational Database Model

A Logical View of Data

Tables and Their Characteristics

- The logical view of the relational database is facilitated by the creation of data relationships based on a logical construct known as a relation.
- A table contains a group of related entity occurrences, or an entity set.
- The characteristics of a relational table are:
    - A table is perceived as a two-dimensional structure composed of rows and columns.
    - Each table row (tuple) represents a single entity occurrence within the entity set.
    - Each table column represents an attribute, and each column has a distinct name.
    - Each intersection of a row and column represents a single data value.
    - All values in a column must conform to the same data format.
    - Each column has a specific range of values known as the attribute domain.
    - The order of the rows and columns is immaterial to the DBMS.
    - Each table must have an attribute or combination of attributes that uniquely identifies each row.
- **Primary key (PK)**
    - An attribute or combination of attributes that uniquely identifies any given row.

Keys

- **key**
    - Consists of one or more attributes that determine other attributes.
    - For example, an invoice number identifies all of the invoice attributes, such as the invoice date and the customer name.

Dependencies

- **Determination**
    - The state in which knowing the value of one attribute makes it possible to determine the value of another.
- **Functional dependence**
    - The value of one or more attributes determines the value of one or more other attributes.
    - In functional dependency, the attribute whose value determines another is called the **determinant** or the **key**.

- The attribute whose value is determined by the other attribute is called the **dependent**.
- Determinants made of more than one attribute require special consideration. It is possible to have a functional dependence in which the determinant contains attributes that are not necessary for the relationship.
- **Full functional dependence** is used to refer to functional dependencies in which the entire collection of attributes in the determinant is necessary for the relationship

Types of Keys

- **Composite key**
    - A key that is composed of more than one attribute. An attribute that is part of a key is called a **key attribute**.
- **Superkey**
    - A key that can uniquely identify any row in the table.
    - **Candidate key**
        - A minimal superkey without any unnecessary attributes.
        - Based on a full functional dependency.
        - Can be chosen to be the primary key.
- **Entity integrity**
    - Each row in the table has its own unique identity.
    - To ensure entity integrity, the primary key has two requirements:
        - All of the values in the primary key must be unique
        - No key attribute in the primary key can contain a null.
- **Foreign key (FK)**
    - The primary key of one table that has been placed into another table to create a common attribute.
    - Used to ensure **referential integrity**
        - The condition in which every reference to an entity instance by another entity instance is valid.
        - Every foreign key entry must be either null or a valid value in the primary key of the related table.
- **Secondary key**
    - Used strictly for data retrieval purposes.

Referential Integrity

- **Cascade delete rule**
    - This rule states that if an attempt is made to delete a record in one table where one or more records with matching foreign key values exist in another table, all associated records will be deleted.
- **Restrict delete rule**
    - This rule states that if an attempt is made to delete a record in one table where one or more records with matching foreign key values exist in another table, the delete operation will not be allowed.
- **Set-to-Null delete rule**
    - If an attempt is made to delete a record in one table where one or more records with matching foreign key values exist in another table, then the foreign key values are set to NULL so we know that the record they used to point to has been deleted.

Integrity Rules

Entity Integrity

- **Requirement**
    - All primary key entries are unique, and no part of a primary key may be null.
- **Purpose**
    - Each row will have a unique identity, and foreign key values can properly reference primary key values.
- **Example**
    - No invoice can have a duplicate number, nro can it be null; in short, all invoices are uniquely identified by their invoice number.

Referential Integrity

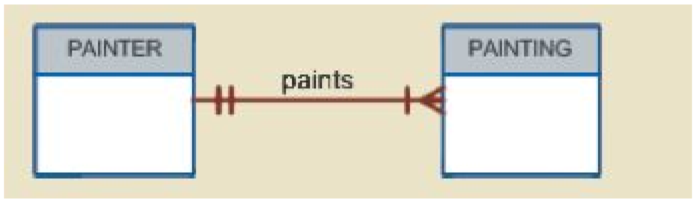
- **Requirement**
    - A foreign key may have either a null entry, as long as it's not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related.
- **Purpose**
    - It is possible for an attribute not to have a corresponding value, but it will be impossible to have an invalid entry; the enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
- **Example**

    - A customer might not yet have an assigned sales representative number, but it will be impossible to have an invalid sales representative number.

- To avoid nulls, some designers use special codes, known as **flags** to indicate the absence of some value. This could be a primary key associated with a dummy row.

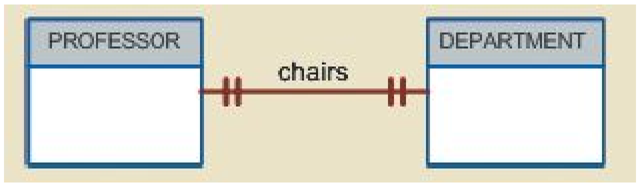
Relationships within the Relational Database

The 1:M Relationship

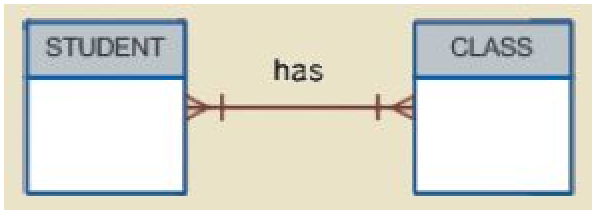


- The 1:M relationship is easily implemented by putting the primary key of the “1” side in the table of the “many” side as a foreign key.
- Any candidate key must have the not-null and unique constraints enforced.

The 1:1 Relationship



The M:N Relationship



- Not supported directly in the relational environment.
- M:N relationships can be implemented by creating a new entity in 1:M relationships with the original entities.
- The problems inherent in the many-to-many relationship can easily be avoided by creating a **composite entity** (also referred to as a **bridge entity** or an **associative entity**).
- The composite entity structure includes — as foreign keys — at least the primary keys of the tables that are to be linked.

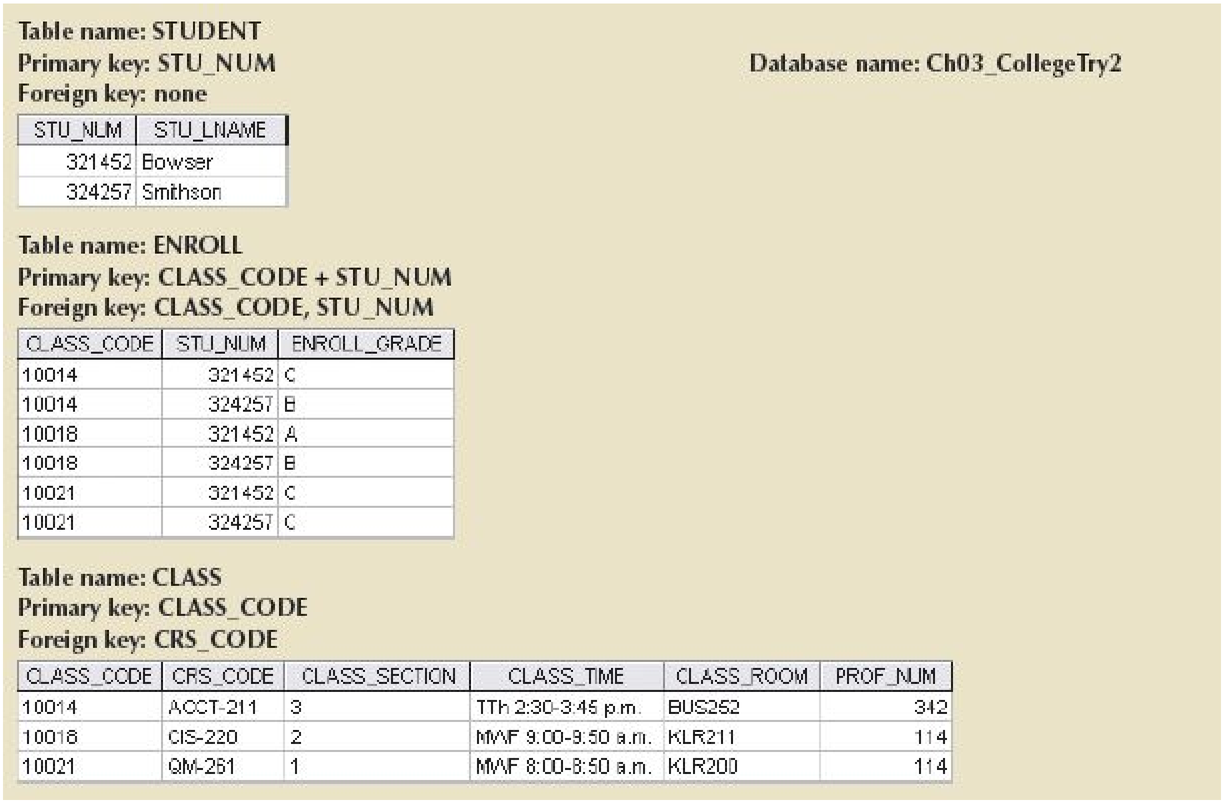


Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch03_CollegeTry2

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL
Primary key: CLASS_CODE + STU_NUM
Foreign key: CLASS_CODE, STU_NUM

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

- Because the ENROLL table links two tables, STUDENT and CLASS, it is also called a **linking table** — the implementation of a composite entity.

Data Redundancy Revisited

- The proper use of foreign keys is crucial to controlling data redundancy, although they do not totally eliminate the problem because the foreign key values can be repeated many times.
- The proper use of foreign keys minimizes data redundancies and the chances that destructive data anomalies will develop.

Indexes

- **Index**
  - An orderly arrangement used to logically access rows in a table
- **Index key**
  - The index's reference point.
  - Points to the location of the data identified by the jey.
- **Unique index**
  - The index key can only have one pointer value associated with it.

Lesson 4 - The Relational Model of Data

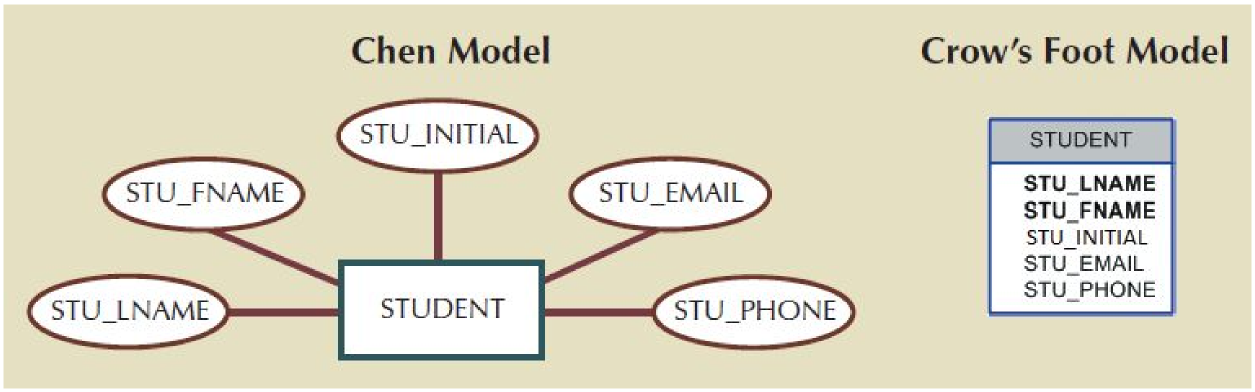
The Entity Relationship Model

- The ERM forms the bases of an ERD.
- The ERD represents the conceptual database as viewed by the end user.
- The Chen notation favors conceptual model.
- The Crow's Foot notation favors a more implementation-oriented approach
- The UML notation can be used for both conceptual and implementation modeling.

Entities

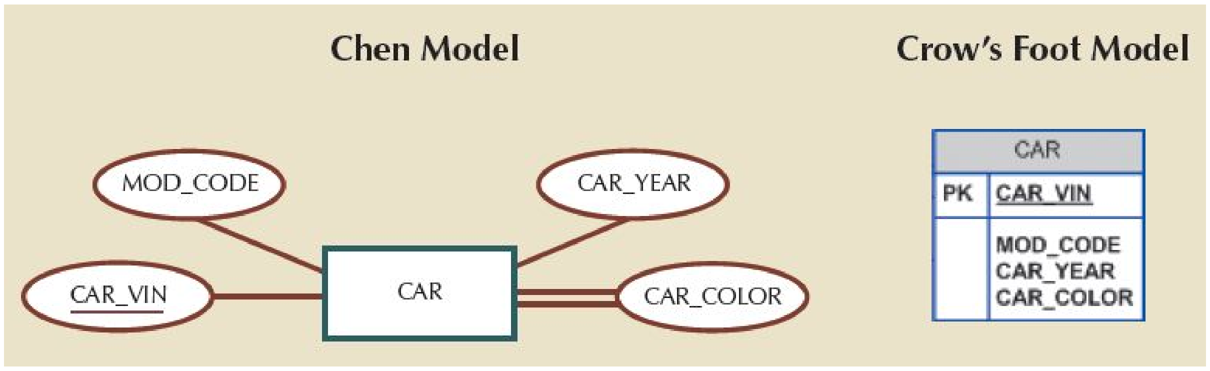
- At the ER modeling level, an entity refers to the *entity set* and not a single entity occurrence.
- In other words, an *entity* in the ERM corresponds to a table, not a row, in the relational environment.
- The ERM refers to a table row as an *entity instance* or *entity occurrence*.

Attributes

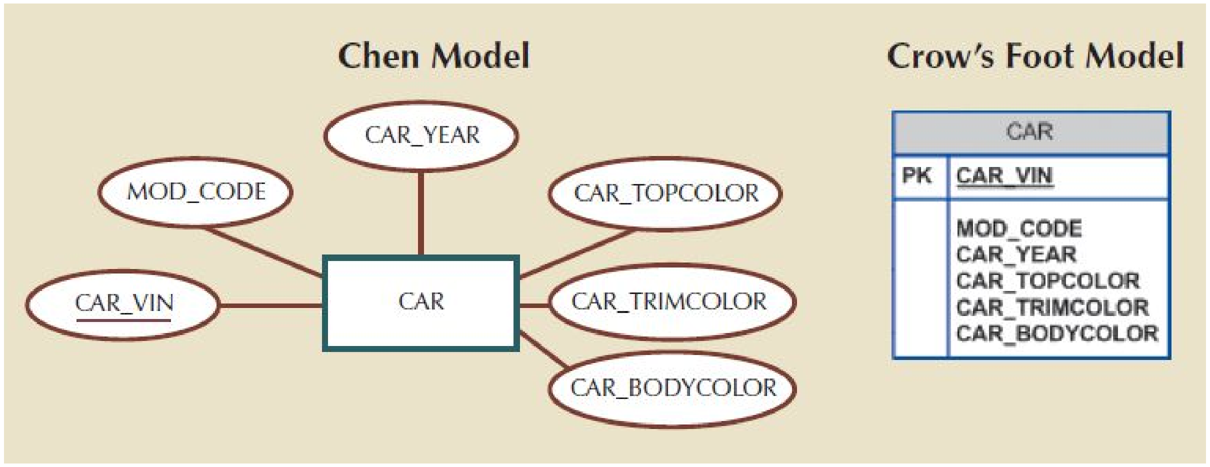
- In the Chen notation, attributes are represented by ovals and are connected to the entity rectangle with a line.
- In the Crow's foot notation, the attributes are written in the attribute box below the entity rectangle.
- 

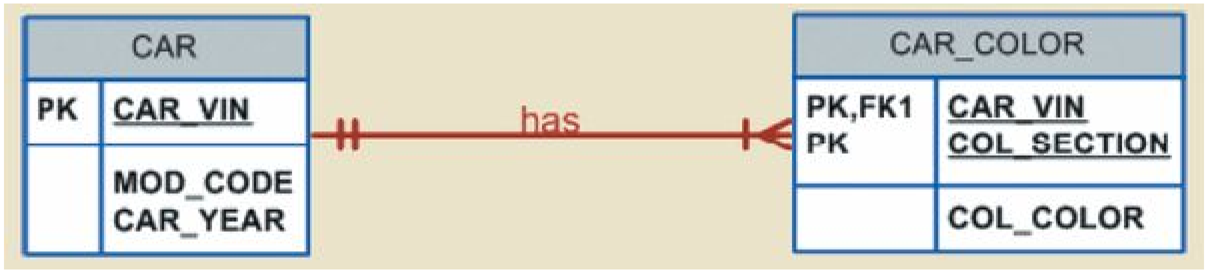
- **Required attribute**
    - Must have a value.
- **Optional attribute**
    - Can be left empty
- **Domains**
    - Attributes have a domain, a set of possible values for a given attribute.
- **Identifiers (primary keys)**

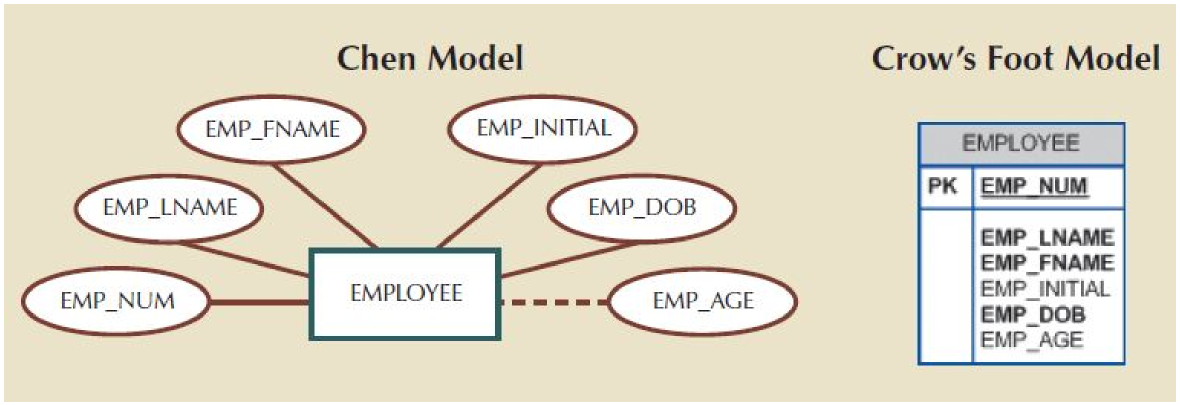
- The ERM uses **identifiers**. In the relational model, entities are mapped to tables, and the entity identifier is mapped as the table's primary key.
- Identifiers are underlined in the ERD.
- Key attributes are also underlined in a frequently used shorthand notation for the table structure, called a **relational schema**, that uses the following format:
    - TABLE NAME (**<u>KEY ATTRIBUTE 1,</u>** ATTRIBUTE 2, ...ATTRIBUTE K)
    - CAR (**<u>CAR VIN</u>**, MOD_CODE, CAR_YEAR, CAR_COLOR)
- **Composite identifiers**
    - Ideally, an entity identifier is composed of only a single attribute. However it is possible to use a **composite identifier**.

- **Composite Attribute**
    - An attribute that can be further subdivided to yield additional attributes.
    - For example, ADDRESS can be subdivided into street, city, state, and zip code.
- **Single-Valued Attribute**
    - An attribute that can have only a single value.
    - For example, a person can have only one Social Security number.
- **Multivalued Attribute**
    - Attributes that can have many values.
    - For example, a person may have several college degrees.
    - Identified in Chen notation with a double line connecting the attribute to the entity.
    - Crow's foot notation does not identify multivalued attributes.
    - 

    - Although the conceptual model can handle M:N relationships and multivalued attributes, you should not implement them in the RDBMS.
    - If multivalued attributes exist, the designer must decide on one of two possible courses of action:
    - 1. Within the original entity, create several new attributes, one for each component of the original multivalued attribute.

- 

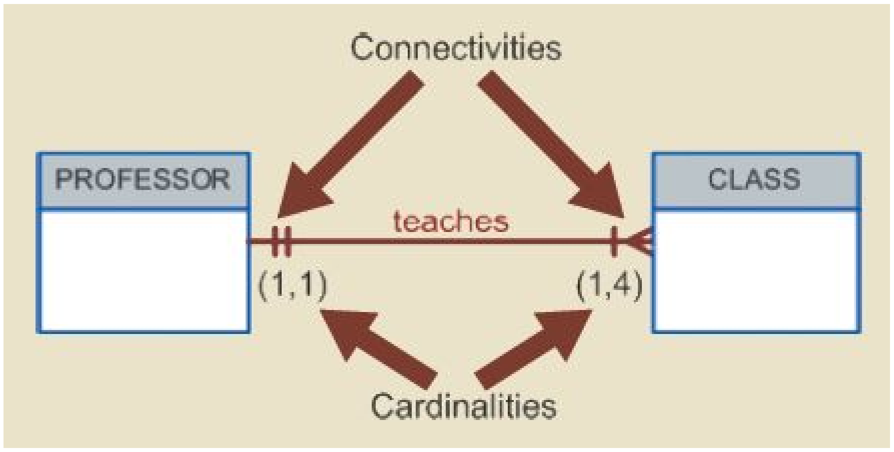
- 2. Create a new entity composed of the original multivalued attributes' components.
- 

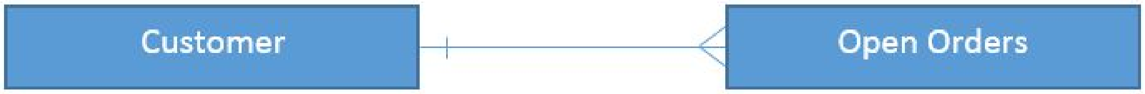
- **Derived attribute**
    - An attribute whose value is calculated from other attributes.
    - Need not be physically stored within the database, instead it can be derived using an algorithm.
    - For example, EMP_AGE may be found using EMP_DOB.
        - If using Microsoft SQL Server, you would use SELECT DATEDIFF("YEAR", EMP_DOB, GETDATE()).
    - A derived attribute is indicated in Chen notation by a dashed line connecting the attribute and the entity.
    - The Crow's Foot notation does not have a method for distinguishing derived attributes.
    - 

Relationships

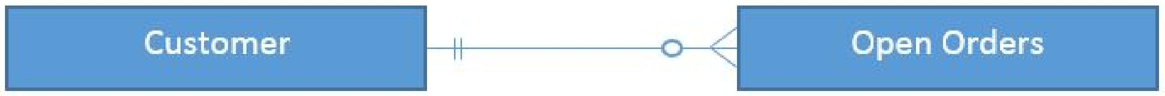
- A relationship is an association between entities.
- Entities that participate in a relationship are also known as **participants**.
- Each relationship is identified by a name that describes the relationship.

Connectivity and Cardinality

- **Connectivity** is used to describe the relationship classification, such as one-to-one, one-to-many, or many-to-many.
- **Cardinality** expresses the minimum and maximum number of entity occurrences associated with one occurence of the related entity.
- In the ERD, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x,y).
- The first value represents the minimum number of associated entities, while the second value represents the maximum number.
- 

- The DBMS cannot handle the implementation of the cardinalities at the table leve — that capability is provided by the application software or by triggers.
- Cardinalities represent the number of occurrences in the *related* entity.
- If a cardinality is written as (1, N) there would be no upper limit to the number of classes a professor might teach.
- Connectivities and cardinalities are established by concise statements known as business rules.
- **Cardinality** denotes how many instances of one object are related to instances of another object. One order can be assigned to only one customer, but a customer can have multiple orders.
  - 

- **Modality** denotes an instance of a specific entity is optional or mandatory in a relationship

    - For instance, an order must have a customer associated with it, but every customer does not need to have an open order.
    - 

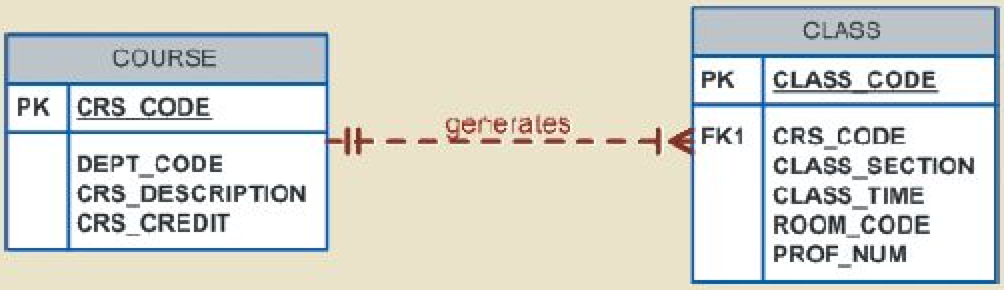
    - Modality is expressed with a straight line for modality 1, or a circle for modality 0. These are drawn in the relationship inside of the cardinality symbols.
    - In other words, modality refers to the **minimum** number of times an instance of an entity can be associated with an instance in a related entity while cardinality refers to the **maximum** number of times this relationship may occur.

Existence Dependence

- An entity is said to be **existence-dependent** if it can exist in the database only when it is associated with another related entity occurrence. In implementation terms, an entity is existence-dependent if it has a mandatory foreign key that cannot be null.
- If an entity can exist apart from all of its related entities, then it is **existence-independent**, and is referred to as a **strong entity** or **regular entity**.

Relationship Strength

- Relationship strength is based on how the primary key of a related entity is defined.
- To implement a relationship, the primary key of one entity appears as a foreign key in the related entity (the child entity).
- Sometimes, the foreign key also is a primary key component in the related entity.
- **Weak Relationship**
    - Also known as a **non-identifying relationship**, exists if the primary key of the related entity does not contain a primary key component of the parent entity.
    - By default, relationships are established by having the primary key of the parent entity appear as a foreign key on the related/child entity.
    - COURSE (**CRS_CODE**, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)
    - CLASS (**CLASS_CODE**, CRS_CODE, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)
    - In this case, a weak relationship exists between COURSE and CLASS because CRS_CODE (the primary key of the parent entity) is only a foreign key in the CLASS entity.
    - Crow’s Foot notation depicts a weak relationship by placing a dashed relationship line between the entities.



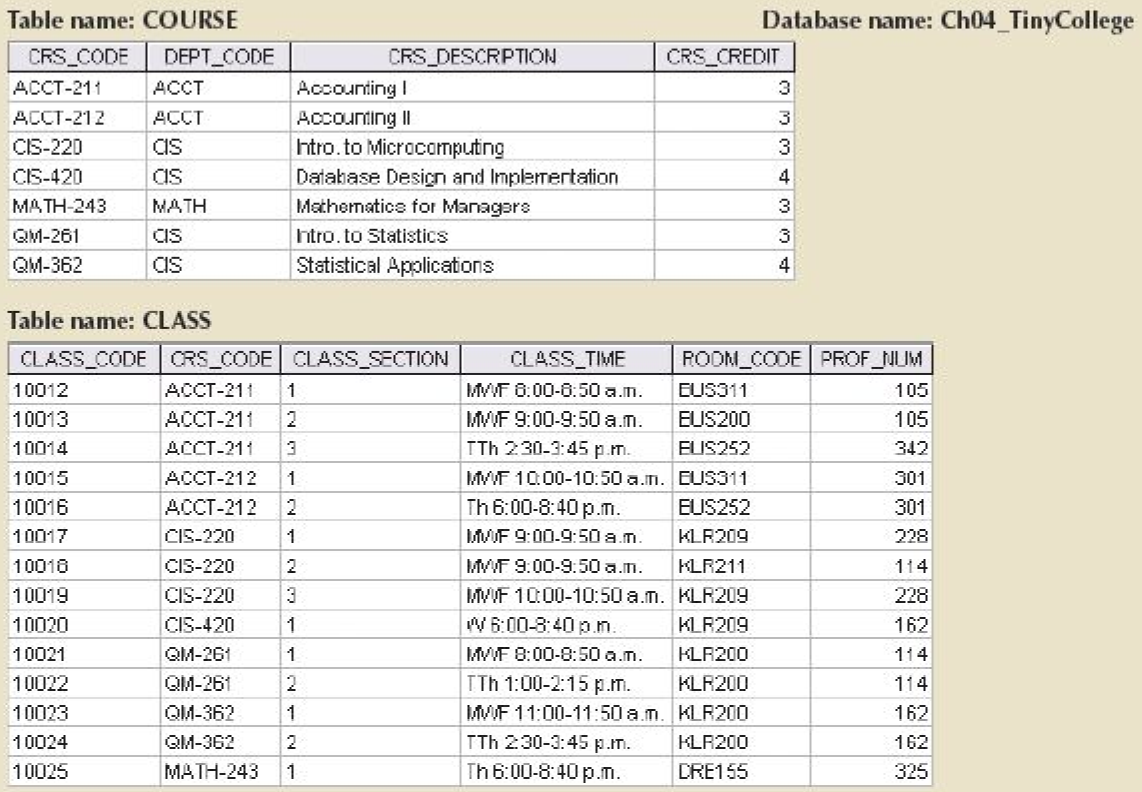


Table name: COURSE **Database name: Ch04_TinyCollege**

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
MATH-243	MATH	Mathematics for Managers	3
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162
10025	MATH-243	1	Th 6:00-8:40 p.m.	DRE155	325

-

- **Strong (identifying) relationship**
    - Exists when the primary key of the related entity contains a primary key component of the parent entity.
    - COURSE (**CRS_CODE**, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)
    - CLASS (**CRS_CODE, CLASS_SECTION**, CLASS_TIME, ROOM_CODE, PROF_NUM)
    - In this case, the CLASS entity primary key is composed of CRS_CODE and CLASS_SECTION. Therefore, a strong relationship exists between COURSE and CLASS.
    - The Crow's Foot notation depicts the strong relationship with a solid line.

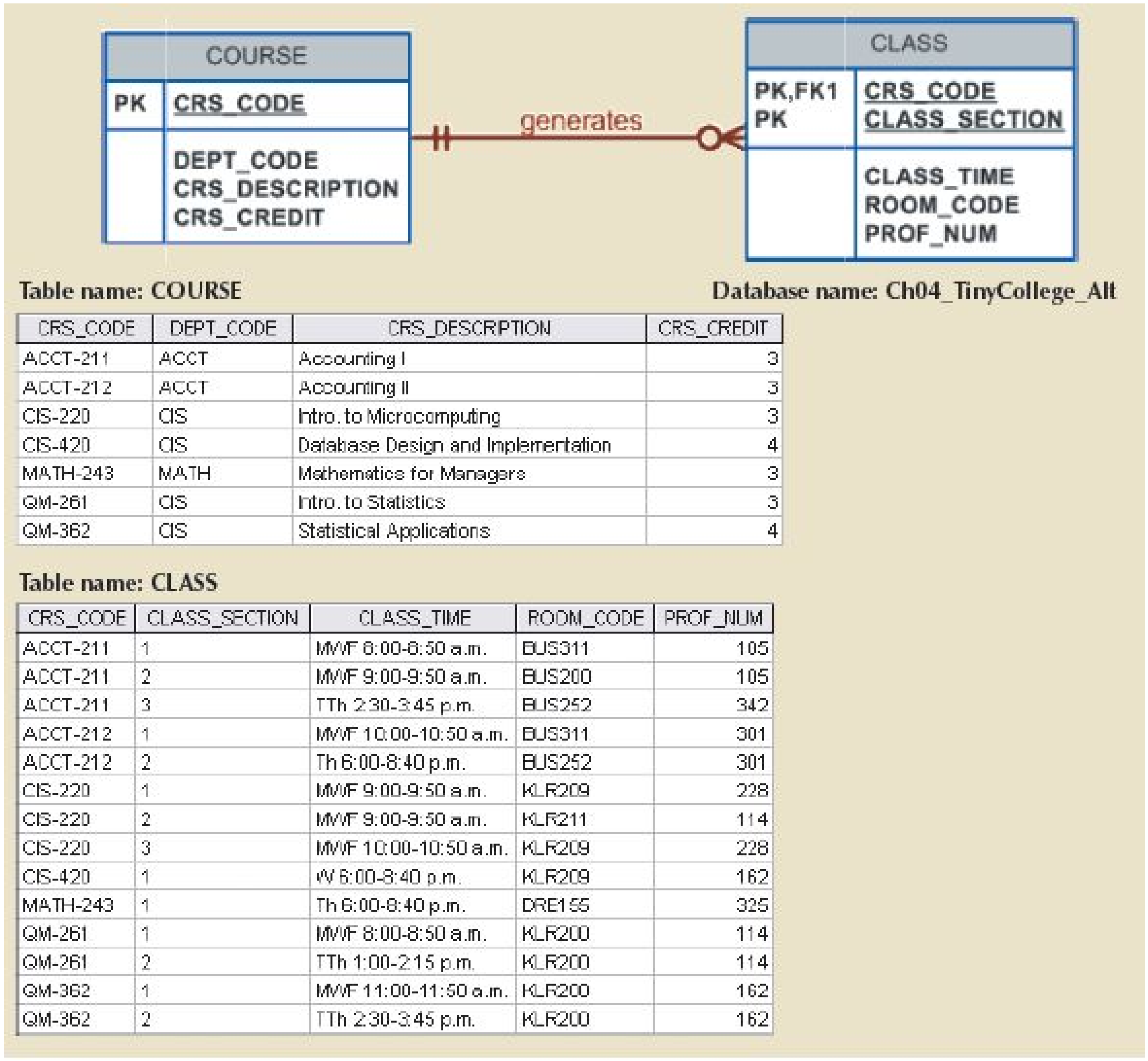
- 

Table name: COURSE **Database name: Ch04_TinyCollege_Alt**

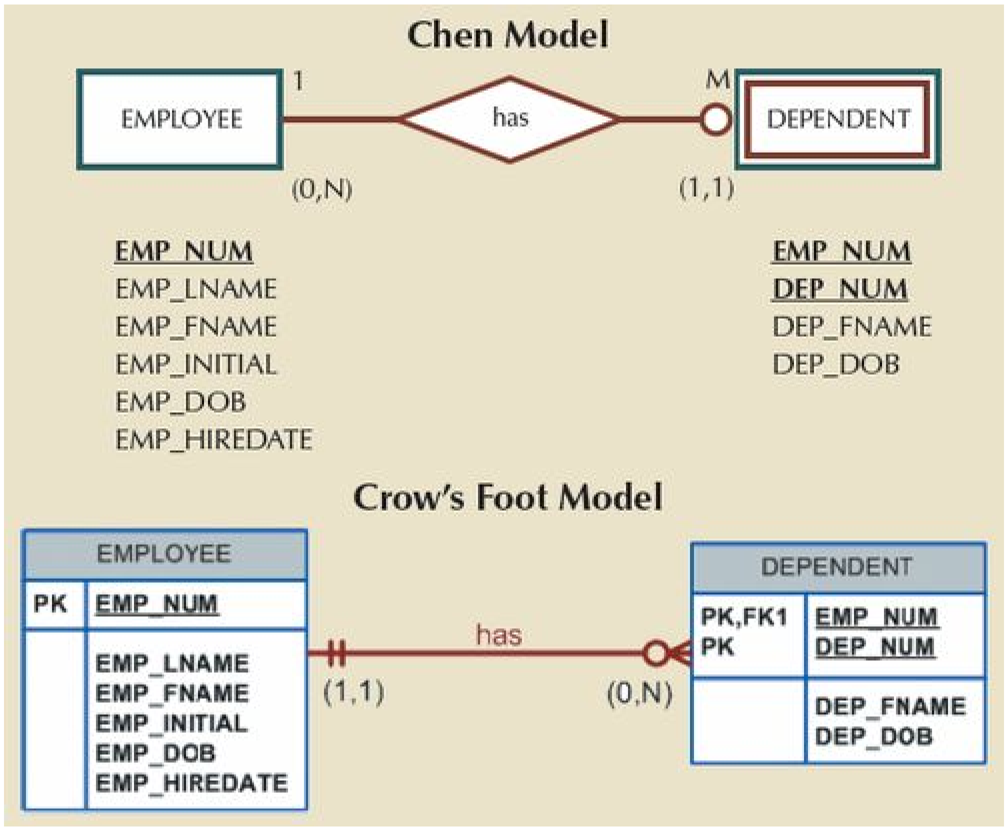
CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
MATH-243	MATH	Mathematics for Managers	3
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS

CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
MATH-243	1	Th 6:00-8:40 p.m.	DRE155	325
QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

Weak Entities

- **Weak Entity**
  - The entity is existence-dependent; it cannot exist without the entity with which it has a relationship.
  - The entity has a primary key that is partially or totally derived from the parent entity in the relationship.

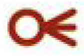
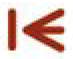
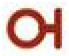
- 

- Note that Chen notation identifies the weak entity using a double-walled entity rectangle.
- Crow's Foot notation uses the relationship line and the PF,FK designation to indicate whether the related entity is weak.

Relationship Participation

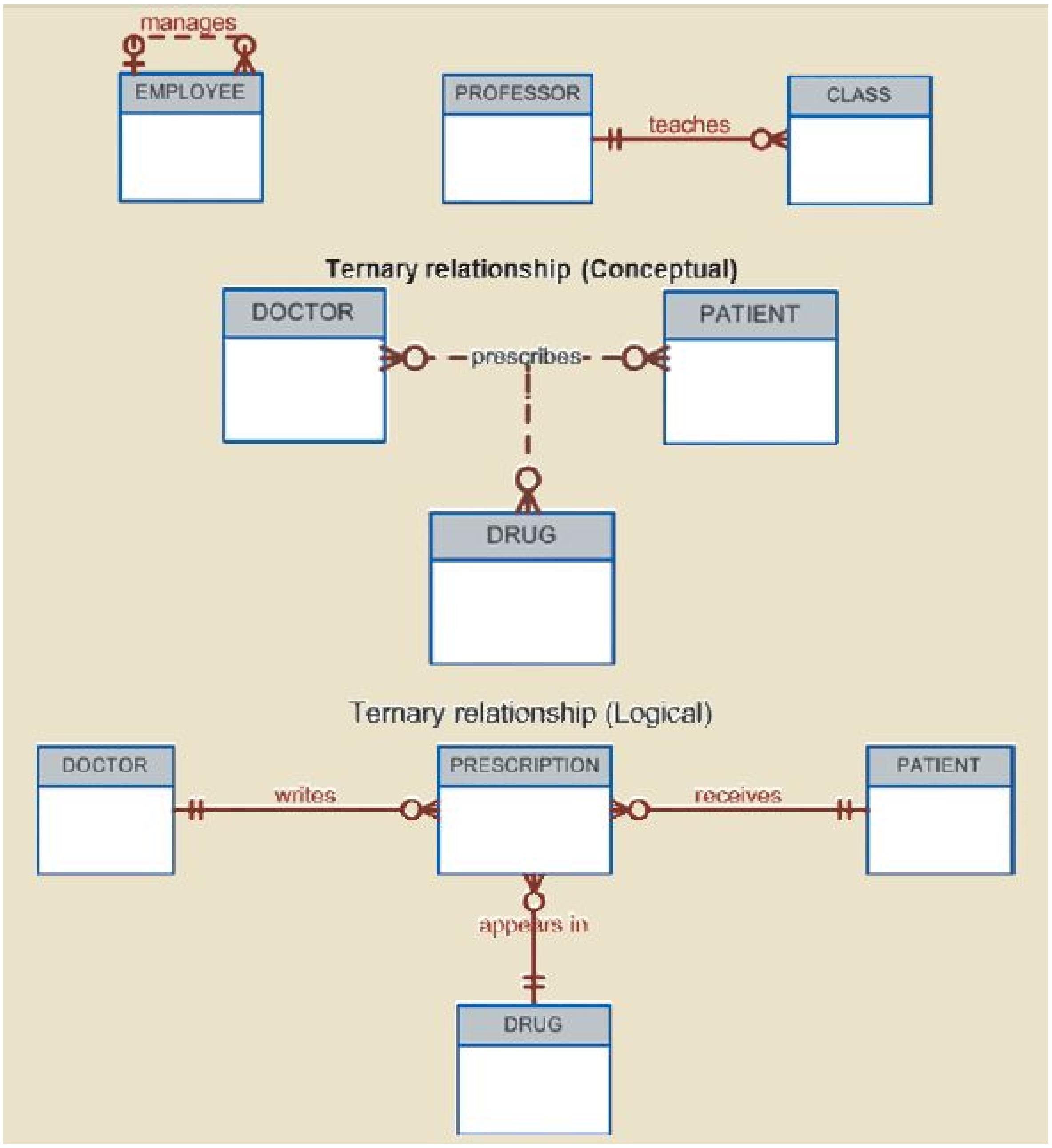
- Participation in an entity relationship is either optional or mandatory.
- **Optional participation**
    - Means that one entity occurrence does not require a corresponding entity occurrence in a particular relationship.
    - For example, in the "COURSE generates CLASS" relationship, you noted that at least some courses do not generate a class.
    - In other words, an entity occurrence (row) in the COURSE table does not necessarily require the existence of a corresponding entity occurrence in the CLASS table.
    - In Crow's Foot notation, an optional relationship is shown by drawing a small circle (O) on the side of the optional entity. The existence of an **optional entity** indicates that its minimum cardinality is 0.
- **Mandatory participation**
    - Means that one entity occurrence requires a corresponding entity occurrence in a particular relationship.

- If no optionality symbol is depicted with the entity, the entity is assumed to exist in a mandatory relationship with the related entity.
- If the mandatory participation is depicted graphically, it is shown as a small hash mark across the relationship line, similar to the Crow's Foot depiction of a connectivity of 1
- The existence of a mandatory relationship indicates the minimum cardinality is at least 1 for the mandatory entity.

Symbol	**Cardinality**	**Comment**
	(0, N)	Zero or many; the "many" side is optional.
	(1, N)	One or many; the "many" side is mandatory.
	(1, 1)	One and only one; the "1" side is mandatory
	(0, 1)	Zero or one; the "1" side is optional

Relationship Degree

- **Relationship degree** indicates the number of entities or participants associated with a relationship.
- **Unary relationship**
    - Exists when an association is maintained within a single entity.
- **Binary relationship**
    - Exists when two entities are associated
- **Ternary relationship**
    - Exists when three entities are associated.
-



- **Unary Relationships**
    - In the case of the unary relationship above, an employee is the manager for one or more employees within that entity. That is, EMPLOYEE has a relationship with itself. Such a relationship is known as a **recursive relationship**.
- **Binary Relationships**
    - The most common type of relationship. Most higher-order relationships are decomposed into appropriate equivalent binary relationships.
- **Ternary and Higher-Order Relationships**
    - A DOCTOR writes one or more PRESCRIPTIONs.
    - A PATIENT may receive one or more PRESCRIPTIONs.
    - A DRUG may appear in one or more PRESCRIPTIONs.

Associative (Composite) Entities

- M:N relationships are valid construct at the conceptual level, and therefore are found frequently during the ER modeling process.
- THE ER model uses the associative entity to represent an M:N relationship between two or more entities.
- The associative entity, also called a *composite* or *bridge entity* is in a 1:M relationship with the parent entities and is composed of the primary key attributes of each parent entity.
- When using the Crow's Foot notation, the associative entity is identified as a strong (identifying) relationship, as indicated by the solid relationship lines between the parents and the associative entity.

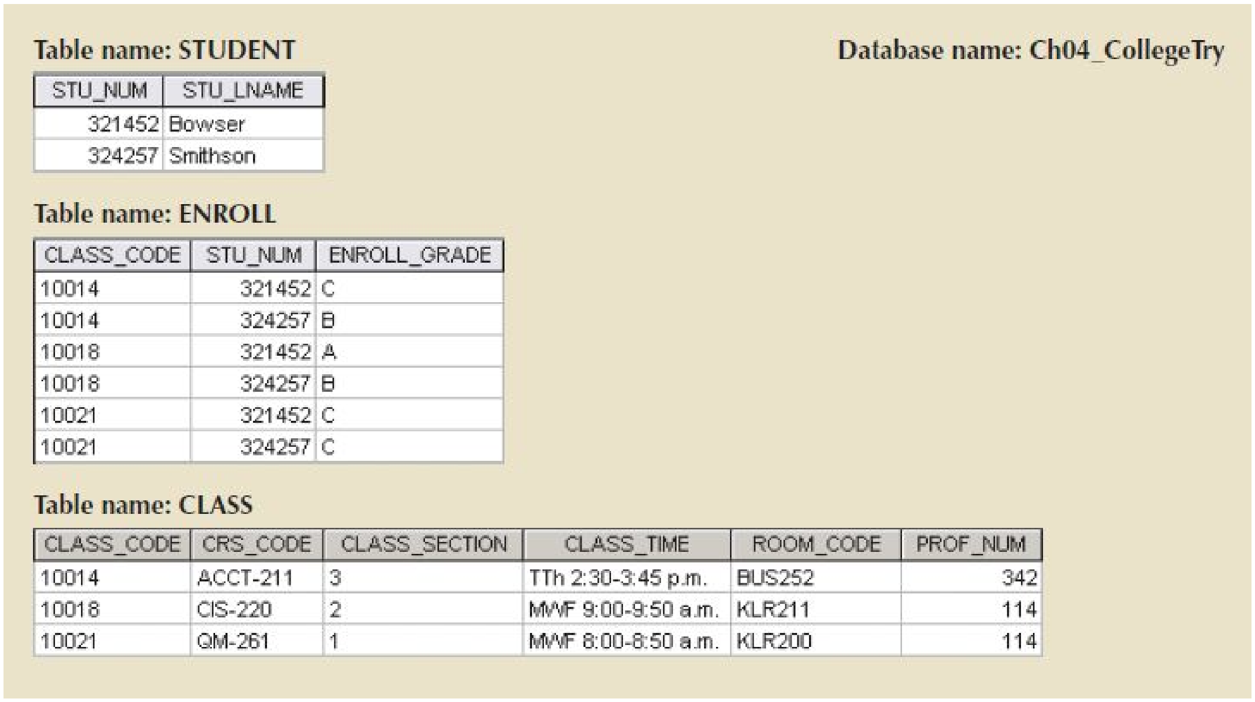


Table name: STUDENT

Database name: Ch04_CollegeTry

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

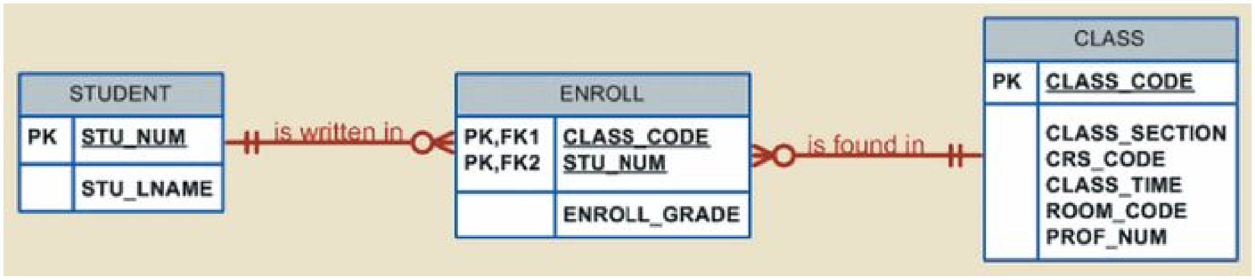
Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

- Note that the composite ENROLL entity is existence-dependent on the other two entities.

- 

Developing an ER Diagram

- Building an ERD usually involves the following activities:
    - Create a detailed narrative of the organization's description of operations.
    - Identify the business rules based on the description of operations.
    - Identify the main entities and relationships from the business rules.
    - Develope the initial ERD.
    - Identify the attributes and primary keys that adequately describe the entities.
    - Revise and review the ERD.

Lesson 5 - Normalization

Dataset Tables and Normalization

- **Normalization**
    - A Process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies.
    - The normalization process involves assigning attributes to tables based on the concept of determination.
- **Denormalization**
    - Produces a lower normal form.
    - The price you pay for increased performance through denormalization is greater data redundancy.

The Need for Normalization

- When designing a new database structure based on the business requirements of the end users, the database designer will construct a data model using a technique such as Crow's Foot notation ERDs.
- After the initial design is complete, the designer can use normalization to analyze the relationships among the attributes within each entity and determine if the structure can be improved through normalization.
- Alternatively, database designers are often asked to modify existing data structures that can be in the form of flat files, spreadsheets, or older database structures. By analyzing relationships among the attributes in the data structure, the designer can use normalization to improve the existing data structure and create an appropriate database design.
- Data redundancy leads to wasted storage space. Even worse, data redundancy produces data anomalies.

The Normalization Process

- The objective of normalization is to ensure that each table conforms to the concept of well-formed relations — in other words, tables that have the following characteristics:
    - Each table represents a single subject.
    - No data item will be unnecessarily stored in more than one table.
    - All non-prime attributes in a table are dependent on the primary key — the entire primary key and nothing but the primary key. The reason for this requirement is to ensure that the data is uniquely identifiable by a primary key value.

    - Each table is void of insertion, update, or deletion anomalies, which ensures the integrity and consistency of the data.

Normal Form	**Characteristic**
1NF	Table format, no repeating groups, and PK identified.
2NF	1NF and no partial dependencies
3NF	2NF and no transitive dependencies
Boyce-Codd normal form	Every determinant is a candidate key (special case of 3NF)
4NF	3NF and no independent multivalued dependencies

- The normalization process works one relation at a time, identifying the dependencies on that relation and normalizing the relation.
- Two types of functional dependencies that are of special interest in normalization are:
    - **Partial dependency**
        - Exists when there is a functional dependence in which the determinant is only part of the primary.
        - For example, if (A,B) -> (C,D), B -> C, and (A,B) is the primary key, then the functional dependence B -> C is a partial dependency because only part of the primary key (B) is needed to determine the value of C.
    - **Transitive dependency**
        - Exists when there are functional dependencies such that X -> Y, Y -> Z, and X is the primary key. In that case, the dependency X -> Z is a transitive dependency because X determines the value of Z via Y.
        - Unlike partial dependencies, transitive dependencies are more difficult to identify among a set of data.
        - Fortunately, there is an effective way to identify transitive dependencies: they occur only when a functional dependence exists among nonprime attributes.

Conversion to First Normal Form

- **Repeating group**
    - A group of multiple entries of the same type can exist for any single key attribute occurrence.
- A relational table must not contain repeating groups, which must be eliminated by making sure that each row defines a single entity.
- Dependencies must be identified to diagnose the normal form.
- Normalization starts with a simple three-step procedure:

        - Step 1: Eliminate the Repeating Groups
            - Start by presenting the data in a tabular format, where each cell has a single value and there are no repeating groups. To eliminate the repeating groups, eliminate the nulls by making sure that each repeating group attribute contains an appropriate data value.
        - Step 2: Identify the Primary Key
            - Maintain a proper primary key that will uniquely identify any attribute value.
        - Step 3: Identify All Dependencies
            - Identify all dependencies, including partial and transitive dependencies.

Conversion to Second Normal Form

- Conversion to 2NF occurs only when the 1NF has a composite primary key. If the 1NF has a single-attribute primary, then the table is automatically in 2NF.
- Step 1: Make New Tables to Eliminate Partial Dependencies
    - For each component of the primary key that acts as a determinant in a partial dependency, create a new table with a copy of that component as the primary key.
    - While these components are placed in the new tables, it is important they also remain in the original table as well. The determinants must remain in the original table because they will be the foreign keys for the relationships needed to relate these new tables to the original table.
- Step 2: Reassign Corresponding Dependent Attributes
    - Determine attributes that are dependent in the partial dependencies. The partial dependencies are removed from the original table and placed in the new table with the dependency's determinant.

Conversion to Third Normal Form

- Step 1: Make New Tables to Eliminate Transitive Dependencies
    - For every transitive dependency, write a copy of its determinant as a primary key for a new table.
    - A **determinant** is any attribute whose value determines other values within a row.
    - It is important that the determinant remain in the original table to serve as a foreign key.
- Step 2: Reassign Corresponding Dependent Attributes
    - Identify the attributes that are dependent on each determinant in Step 1.
    - Place the dependent attributes in the new tables with their determinants and remove them from their original tables.
- If a table has multiple candidate keys and one of them is a composite key, the table can have partial dependencies based on this composite candidate key, even when the primary key chosen is a single attribute.

- In those cases, those dependencies would be perceived as transitive dependencies and would not be resolved until 3NF.

Improving the Design

- **Evaluate PK Assignments**
- **Evaluate Naming Conventions**
- **Refine Attribute Atomicity**
  - An **atomic attribute** is one that cannot be further subdivided.
- **Identify New Attributes**
- **Identify New Relationships**
- **Refine Primary Keys as Required for Data Granularity**
  - **Granularity** refers to the level of detail represented by the values stored in a table's row.
- **Maintain Historical Accuracy**
- **Evaluate Using Derived Attributes**

Lesson 6 - Fundamentals of SQL

Introduction to SQL

- SQL functions fit into two broad categories
  - It is a **data definition language**.
    - SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects.
  - It is a **data manipulation language**.
    - SQL includes commands to insert, update, delete, and retrieve data within the database tables.

Command or Option	**Description**
CREATE SCHEMA AUTHORIZATION	Creates a database schema.
CREATE TABLE	Creates a new table in the user's database schema.
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Validates data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows and columns from one or more tables
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table and its data

DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

- SQL is a nonprocedural language, you merely command what is to be done, you do not have to worry about how.

Command or Option	**Description**
INSERT	Inserts rows into a table
SELECT	Select attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute’s values in one or more table’s rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to its original values
=, <, >, <=, >=, <>, !=	Used in conditional expressions
AND/OR/NOT	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
Aggregate Functions	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given

	column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

SELECT Queries

Selecting Rows with Conditional Restrictions

- You can select partial table contents by placing restrictions on the rows to be included in the output.
- Use the **WHERE** clause to add conditional restrictions to the SELECT statement that limits the rows returned by the query.

    -
      ```
      SELECT    columnList
      FROM      tableList
      [WHERE    conditionList   ];
      ```
- **The SELECT statement retrieves all rows that matched the specific conditions.**
- For example:

    -
      ```
      SELECT    P_DESCRIPT, P_INDATE, P_PRICE, V_CODE
      FROM      PRODUCT
      WHERE     V_CODE = 21344;
      ```
- **Date comparison is software specific.**
    - WHERE   P_INDATE >= ‘20-Jan-2016’;
    - WHERE   P_INDATE >= ‘2016-01-20’
    - WHERE   P_INDATE >= #20-Jan-16#
- **Use computed columns and column aliases:**

    -
      ```
      SELECT    P_QOH * P_PRICE AS TOTVALUE
      FROM      PRODUCT;
      ```
- You could also use a computed column, an alias, and date arithmetic in a single query.

    -
      ```
      SELECT    P_CODE, P_INDATE, DATE() - 90 AS CUTDATE
      FROM      PRODUCT
      WHERE     P_INDATE <= DATE() - 90;
      ```
- **The BETWEEN Special Operator**

    -
      ```
      SELECT    *
      FROM      PRODUCT
      WHERE     P_PRICE BETWEEN 50.00 AND 100.00;
      ```

- **The IS NULL Special operator**

    -
      ```
      SELECT      P_CODE, P_DESCRIPT, V_CODE
      FROM        PRODUCT
      WHERE       V_CODE IS NULL;
      ```
- **The LIKE Special Operator**
    - Used in conjunction with wildcards to find patterns within string attributes. Standard SQL allows you to use the percent sign % and underscore _ wildcard characters to make matches when the entire string is not known:
        - % means any and all *following* or *preceding* characters are eligible.
            - ‘J%’ includes Johnson, Jones, Jernigan, July, J-231Q
            - ‘Jo%’ includes Johnson and Jones.
            - ‘%n’ includes Johnson and Jernigan
        - _ means any *one* character may be substituted for the underscore.
            - ‘_o_es’ includes Jones, Cones, Cokes, totes, and roles

    -
      ```
      SELECT      V_NAME, V_CONTACT
      FROM        VENDOR
      WHERE       V_CONTACT LIKE ‘Smith%’;
      ```
    - For case sensitive RDBMS, make a conversion to eliminate case sensitivity:

        -
          ```
          SELECT      V_NAME, V_CONTACT
          FROM        VENDOR
          WHERE       UPPER(V_CONTACT) LIKE ‘SMITH%’;
          ```
- **The IN Special Operator**
    - Uses a value list of the same data type to compare to the attribute.

        -
          ```
          SELECT      *
          FROM        PRODUCT
          WHERE       V_CODE IN (21344, 24288)
          ```
    - Can be used in conjunction with subqueries.
    - For example, suppose you want to list the V_CODE and V_NAME of only those vendors who provide products:

        -
          ```
          SELECT      V_CODE, V_NAME
          FROM        VENDOR
          WHERE       V_CODE IN (SELECT V_CODE FROM PRODUCT);
          ```
- **The EXISTS Special Operator**
    - If a subquery returns any rows, run the main query.

    -
      ```
      SELECT      *
      FROM        VENDOR
      WHERE       EXISTS (SELECT * FROM PRODUCT WHERE P_QOH <=
                  P_MIN);
      ```

Filtering SQL Query Results

Filtering using ORDER BY

- The **ORDER BY** clause is useful when the listing order is important to you.

    -
      ```
      SELECT      columnList
      FROM        tableLIST
      [WHERE      conditionLIST]
      [ORDER BY   columnList [ASC | DESC] ];
      ```
- Multilevel ordered sequence is known as a **cascading order sequence** and can be created easily by listing several attributes after the ORDER BY clause.

    -
      ```
      SELECT      EMP_LNAME, EMP_FNAME, EMP_INITIAL
      FROM        EMPLOYEE
      ORDER BY    EMP_LNAME, EMP_FNAME, EMP_INITIAL
      ```

Filtering using DISTINCT

- Filter out duplicates from a query.

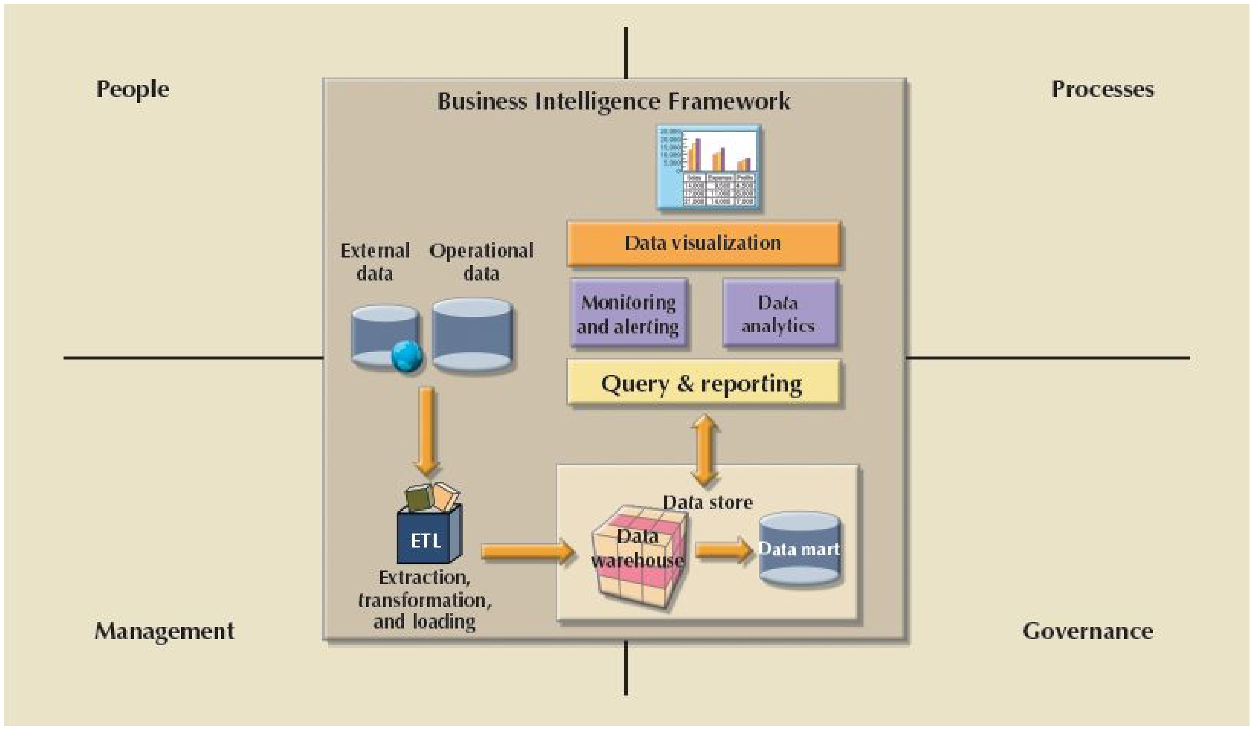
    -
      ```
      SELECT      DISTINCT Department
      FROM        Employee
      ORDER BY    Department
      ```

Lesson 7 - Business Intelligence

Business Intelligence

- **Business intelligence**
    - Describes a comprehensive, cohesive, and integrated set of tools and processes used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information to support business decision making.
    - This intelligence is based on learning and understanding the facts about the business environment.
    - BI is a framework that allows a business to transform data into information, information into knowledge, and knowledge into wisdom.
- Implementing BI an an organization involves capturing not only internal and external business data, but also the metadata, or knowledge about the data.
- BI requires a deep understanding and alignment of the business processes, business data, and information needs of users at all levels in an organization.
- In general, BI provides a framework for:
    - Collecting and storing operational data
    - Aggregating the operational data into decision support data
    - Analyzing decision support data to generate information
    - Presenting such information to the end user to support business decisions
    - Making business decisions, which in turn generate more data that is collected, stored, and so on.
    - Monitoring results to evaluate outcomes of the business decisions, which again provides more data to be collected, stored, and so on.
    - Predicting future behaviors and outcomes with a high degree of accuracy.

Business Intelligence Architecture



Component	Description
ETL Tools	Data **extraction, transformation, and loading (ETL)** tools collect filter integrate and aggregate internal and external data to be saved into a data store optimized for decision support.
Data store	The data store is optimized for decision support and is generally represented by a *data warehouse* or a *data mart*. The data is stored in structures optimized for data analysis and query speed.
Query and reporting	This component performs data selection and retrieval, and it is used by the data analyst to create queries that access the database and create the required reports.
Data visualization	This component presents data to the end user in a variety of meaningful and innovative ways. This tool helps the end user select the most appropriate presentation format.
Data monitoring and alerting	Allows real-time monitoring of business activities. The BI system will present the concise information in a single integrated view for the data analyst.
Data analytics	This component performs data analysis and data-mining tasks using

	the data in the data store. This tool advises the user as to which data analysis tool to select and how to build a reliable business data model.

- **Master data management (MDM)**
    - A collection of concepts, techniques, and processes for the proper identification, definition, and management of data elements within an organization.
    - MDM's main goal is to provide a comprehensive and consistent definition of all data within an organization.
    - MDM ensures that all company resources that work with data have uniform and consistent views of the company's data.
- **Governance**
    - A method or process of government. in this case, BI provides a method for controlling and monitoring business health and for consistent decision making.
- **Key performance indicators (KPIs)**
    - Quantifiable numeric measurements that assess the company's effectiveness or success in reaching its strategic and operational goals.
- A modern BI system provides three distinctive reporting styles:
    - **Advanced reporting**
        - Insightful information about the organization in a variety of presentation formats. Furthermore, the reports provide interactive features that allow the end user to study the data from multiple points of view.
    - **Monitoring and alerting**
        - After a decision has been made, the BI system offers ways to monitor the decision's outcome.
    - **Advanced data analytics**
        - A BI system provides tools to help the end user discover relationships, patterns, and trends hidden within the organization's data.

Online Analytical Processing

Multidimensional Data Analysis Techniques

- The most distinctive characteristic of modern OLAP tools is their capacity for multidimensional analysis.
- Data is viewed as part of a multidimensional structure.
- Multidimensional data analysis techniques are augmented by the following functions:
    - **Advanced data presentation functions**
        - These functions include 3d graphics, pivot tables, crosstabs, data rotation, and three-dimensional cubes.
        - Such tools are compatible with desktop spreadsheets, statistical packages, and query and report packages.

- **Advanced data aggregation, consolidation, and classification functions**
    - These allow data analyst to create multiple data aggregation levels, slice and dice data, and drill down and roll up data across different dimensions and aggregation levels.
- **Advanced computational functions**
    - These include business-oriented variables such as market share, period comparisons, sales margins, product margins, and percentage changes.
- **Advanced data-modeling functions**
    - These provide support for what-if scenarios, variable assessment, contributions to outcome, linear programming, and predictive modeling tools. Predictive modeling allows the system to build advanced statistical models to predict future values with a high accuracy.

Advanced Database Support

- OLAP tools must have the following advanced data access features:
    - Access to many different kinds of DBMSs, flat files, and internal and external data sources.
    - Access to aggregated data warehouse data as well as to the detail data found in operational databases.
    - Advanced data navigation features such as drill-down and roll-up
    - Rapid and consistent query response times
    - The ability to map end-user requests, expressed in either business or model terms, to the appropriate data source and then to the proper data access language.
    - Support for very large databases.

OLAP Architecture

- GUI
- Analytical processing logic
- Data processing logic

Relational OLAP

- **Relational online analytical processing** provides OLAP functionality by using relational databases and familiar relational query tools to store an analyze multidimensional data.
- ROLAP adds the following extensions to traditional RDBMS tech:
    - Multidimensional data schema support within the RDBMS
    - Data access language and query performance optimized for multidimensional data
    - Support for very large databases

- The star scheme is designed to optimize data query operations rather than data update operations.
- ROLAP adds support for the star schema when familiar query tools are used.

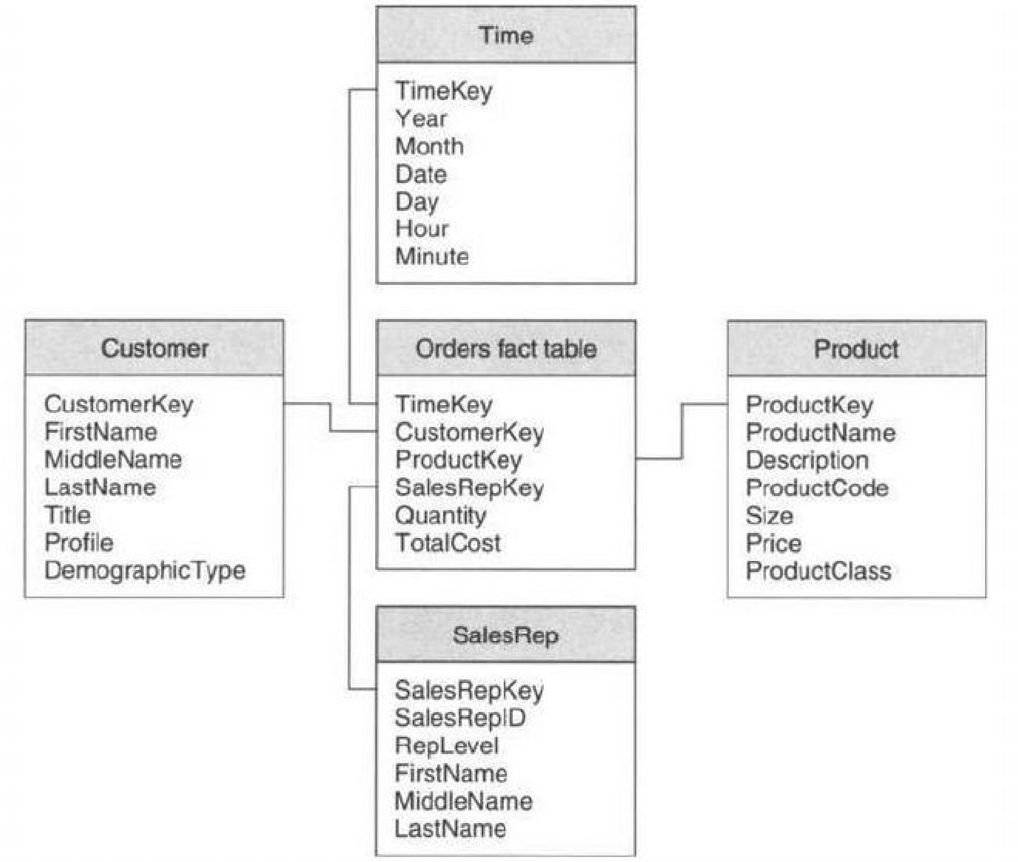
Supplementary Ch 1 - Business Intelligence and Information Exploitation

Why Business Intelligence?

- **Increased profitability**
- **Decreased costs**
- **Improved customer relationship management**
- **Decreased risk**

Supplementary Ch 6 - Data Warehouses, Online Analytical Processing, Metadata

- Dimensional modeling captures the basic unit of representation as a single multikeyed entry in a slender fact table, with each key exploiting the relational model to refer to the different dimensions associated with those facts.
- The representation of a **dimensional model** is straightforward:
    - A fact table contains a key whose components are keys to individual dimension tables, along with some specific pieces of information relevant to the fact.
    - This data is typically numeric so it is amenable to aggregate functions.



    -
    - Each fact represents the total quantity of a product sold to a specific customer at a particular point-of-sales location at a particular point of time.
    - This model contains four dimensions:
        - product, customer, pos location, and time.
    - This model layout is referred to as a **star join layout** or **star schema**.

The Data Warehouse

- A centralized repository of information
- Arranged around the relevant subject areas important to the corporation as a whole.
- A queryable source of data for an enterprise.
- Used for analysis and not for transaction processing.
- The data in a data warehouse is nonvolatile.
- The target location for integrating data from multiple sources, both internal and external to an enterprise.
- Information is loaded into the data warehouse after a number of preprocessing steps that include extracting data from the various data sources:
    - data profiling, data cleansing, and transformations.
- The data is then reformulated into dimensional form and loaded into the target warehouse. These processes compose what is referred to as the data warehouse back end.

The Data Mart

- A subject oriented data repository, similar in structure to the enterprise data warehouse, but holds the data necessary for decision support and BI needs of a specific department or group within the organization.
- Data marts are more for formalized reporting and directed drill-down.
- Centered on specific goals and decision support of a specific department, so the amount of data is much smaller.

Online Analytical Processing

- Differs from OLTP (online transaction processing)
- Presenting data sourced from a warehouse or mart in a way that allows the consumer to view comparative metrics across multiple dimensions.

The Importance of Metadata

- Metadata encapsulates both the logical and physical business knowledge required to transform disparate data sets into a coherent warehouse.
- Captures the structure and meaning of the data being fed into the warehouse.
- Provides a road map for deriving an information audit trail.
- Can capture differences associated with how data is manipulated over time.
- Provides the means for tracing the evolution of information as a way to validate and verify results derived from an analytical process.

Technical Metadata

- Describes the structure of information.

Business Metadata

- Incorporates metadata that describes the structure of data as perceived by business clients.
- Descriptions of the methods for accessing data for client analytical applications.
- Business meanings for tables and their attributes.
- Data ownership characteristics and responsibilities
- Data domains and mappings between those domains, for validation
- Business rules that describe constraints or directives associated data within a record or between records as joined.

Supplementary Ch 10 - Information Integration

ETL: Extract, Transform, Load

- Get the data from the source location
- Map the data from its original form into a data model that is suitable for manipulation at the staging area
- Validate and clean the data
- Apply any transformations to the data that are required before the data sets are loaded into the repository
- Map the data from its staging area model to its loading model
- Move the data set to the repository
- Load the data into the warehouse

Staging Architecture

- Typically a combination of a hardware platform and appropriate management software we refer to as the **staging area**

Transformation

- **data type conversion** — includes parsing strings and transforming them into the proper representational form.
- **data cleansing** — the rules uncovered through profiling can be applied, along with the directed actions that can be used to correct data that is known to be incorrect.
- **integration** — exploiting the discovery table and foreign keys for representing linkage between different tables, along with the generation of alternate keys that are independent of any systemic business rules, mapping keys from one system to another,

archiving data domains and codes mapped into those data domains, and maintaining the metadata.
- **Referential integrity checking** — in relation to the foreign key relationships exposed through profiling or as documented through interaction with subject matter experts, this component checks that any referential integrity constraints are not violated.
- **Derivations** — any transformations based on business rules, new calculations, string manipulations, etc. that need to be applied as the data moves from source to target are applied.
- **Denormalization and renormalization** — data that is in normalized form when it comes from the source system needs to be broken out into a denormalized form. Data sourced from join extractions may be denormalized and may need to be renormalized before it is forwarded to the warehouse.
- **Aggregation** — any aggregate information used for populating summaries or any cube dimensions can be performed at the staging area
- **Audit information** — calculate some auditing information to make sure what you have is what you wanted.
- **Null conversion** — transform different nulls from disparate systems

Loading

- The loading component of ETL is centered on moving the transformed data into the warehouse. The critical issues include the following:
    - **Target dependencies** — such as where and on how many machines the repository lives, and the specifics of loading data into the platform
    - **Refresh volume and frequency** — such as whether the data warehouse is toe be loaded on an incremental basis, whether data is forwarded to the repository as a result of triggered transaction events, or whether all data is periodically loaded in the warehouse in the form of a full refresh.

Supplementary Ch 14 - Data Mining

Six Basic Tasks of Data Mining

Classification

- Involves examining the attributes of a particular object and assigning it to a defined class.

Estimation

- A process of assigning some continuously valued numeric value to an object.

Prediction

- Attempt to classify objects according to some expected future behavior.

Affinity Grouping

- A process of evaluating relationships or associations between data elements that demonstrate some kind of affinity between objects.

Clustering

- The task of taking a large collection of objects and dividing them into smaller groups of objects that exhibit some similarity.

Description

- The process of trying to characterize what has been discovered or trying to explain the results of the data mining process.