Study Notes for DB Design and Management Exam 1 (Chapters 1-2-3)

Chapter 1

Glossary Table

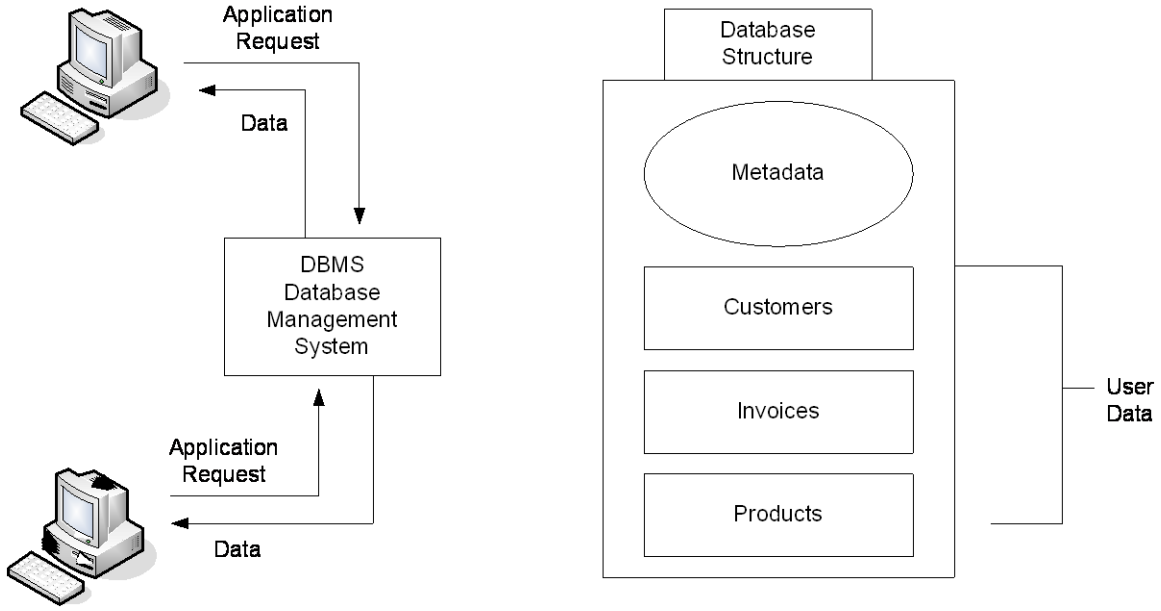
data—Raw facts; that is, facts that have not yet been processed to reveal their meaning to the end user.
field—A character or group of characters (alphabetic or numeric) that defines a characteristic of a person, place or thing. For example, a person's social security, address, phone, bank balance, and so on all constitute fields.
record—A collection of related (logically connected) fields.
file—A named collection of related records.
information—Facts (data) that are arranged in meaningful patterns. Information consists of *transformed* data and facilitates decision making.
data management—A process that focuses on data collection, storage and retrieval. Common data management functions include addition, deletion, modification and listing.
database—A computer structure that houses a collection of related data. A database contains two types of data: end user data (raw facts) and metadata. The metadata consist of data about data, that is, the data characteristics and relationships.
metadata—Data about data; that is, data concerning data characteristics and relationships. **See *data dictionary***.
database management system (DBMS)—Software that serves as an intermediary between the user and the database. The DBMS translates user requests into the computer code that is required to fulfill those requests. A DBMS manages the data stored within the database.
query—A question or task asked by an end user of a database in the form of SQL code.
ad hoc queries—"Spur-of-the-moment" questions.
single-user DBMS—A database management system classification that depicts a DBMS that supports only one user at a time.
desktop database—A single-user database that runs on a personal computer.

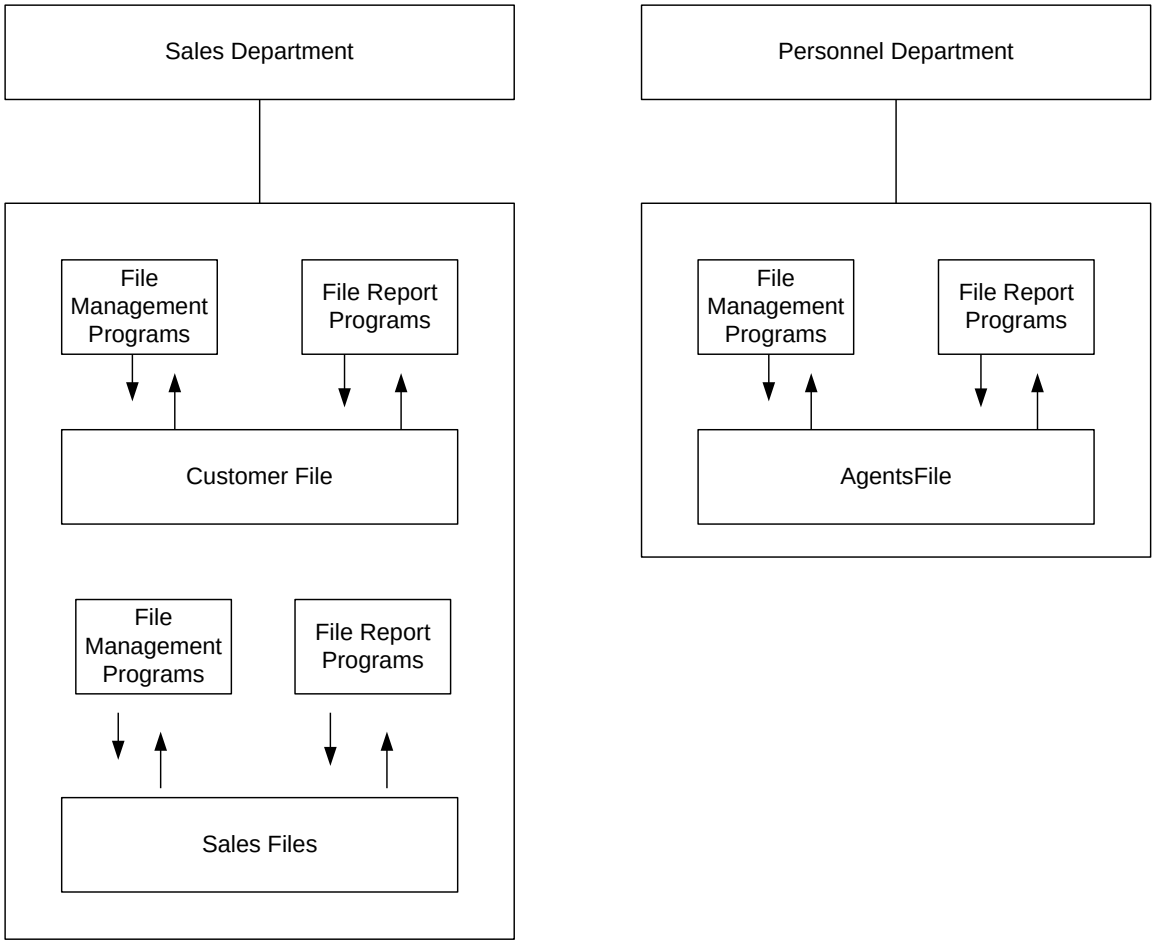
multiuser DBMS—A database management system that supports multiple concurrent users.
workgroup database—A multiuser database that supports a relatively small number of users (usually fewer than 50) or for a specific department in an organization.
enterprise database—The overall company data representation, which provides support for present and expected future needs.
centralized DBMS—A database management system that supports a database located at a single site.
distributed database management system (DDBMS)—A database management system that supports a database distributed across several different sites; governs the storage and processing of logically related data over interconnected computer systems in which both data and processing functions are distributed among several sites.
transactional database—A database designed to keep track of the day to day transactions of an organization. **See also *production database***.
production database—The main database designed to keep track of the day-to-day operations of a company. **See also *transaction database*.**
data warehouse database—A database that focuses primarily on the storage of data used to generate information required to make tactical or strategic decisions.
database design—The process that yields the description of the database structure. The database design process determines the database components. Database design is the second phase of the database life cycle.
redundant data—Duplicated data that are stored in more than one location.
logical design—A stage in the design phase that matches the conceptual design to the requirements of the selected DBMS an is, therefore, software-dependent. It is used to translate the conceptual design into the internal model for a selected database management system, such as DB2, SQL Server, Oracle, IMS, Informix, Access, and Ingress.
data processing (DP) specialist—A now obsolete position formed in the conversion from manual filing systems to computer filing systems; once filled by an employee who created and programmed the necessary file structures, wrote the software that managed the data in those structures, and designed the application programs that produced reports from the file data.

data processing (DP) manager—A DP specialist who evolved into the department supervisor. Roles include: managing the technical and human resources, supervising the senior programmers, program troubleshooting.
third-generation language (3GL)—A language that requires the programmer to specify both what must be done and how it is to be done. Examples include, COBOL, BASIC and FORTRAN.
fourth-generation language (4GL)—A nonprocedural language, such as SQL, that only requires the user to define what must be done; the details of *how* the user's commands are executed are handled by the 4GL.
structural dependence—A data characteristic that exists when a change in the database schema affects data access, thus requiring changes in all access programs.
data dependence—A data condition in which the data representation and manipulation are dependent on the physical data storage characteristics.
structural independence—A data characteristic that exists when changes in the database schema do not affect data access.
physical data format—The way in which the computer "sees" the data.
logical data format—The way in which a human being views data.
islands of information—Independent data pools created and managed by different organizational components. Typical of old-style file systems.
data redundancy—A condition that exists when the data environment contains redundant—unnecessarily duplicated— data.
data inconsistency—A condition in which different versions of the same data yield different (inconsistent) results.
data integrity—A condition in which given data always yield the same result. Data integrity is mandatory in any database.
data anomaly—A data abnormality that exists when inconsistent changes to the database have been made. Example: An employee moves, but the address change is only corrected in one file and not across all files in the database.
database system—An organization of components that define and regulate the collection, storage, management, and use of data in a database environment.

database administrator (DBA)—Person responsible for the planning, organization, control and monitoring of the centralized and shared corporate database. The DBA is the general manager of the database-administration department.
data dictionary—A DBMS component that stores metadata – data about data. Thus, the data dictionary contains the data definition as well as its characteristics and relationships. A data dictionary may also include data that are external to the DBMS. **See also *active data dictionary* and *passive data dictionary*.**
performance tuning—Activities that make the database perform more efficiently in terms of storage and access speed.
data independence—A condition that exists when data access is unaffected by changes in the physical data storage characteristics.
query language—A nonprocedural language that lets the user specify *what* is to be done without specifying *how* it is to be done. An example of a query language is SQL.
data definition language (DDL)—The language that allows a database administrator to define the database structure, schema, and subschema.
data manipulation language (DML)—The language (set of commands) that allows the end user to manipulate the data in the database (Select, Insert, Update, Delete).

The DBMS Manages the Interaction between the end user and the Database





A Simple File System – showing possible **Islands of Information**

Contents of an un-Normalized Customer file

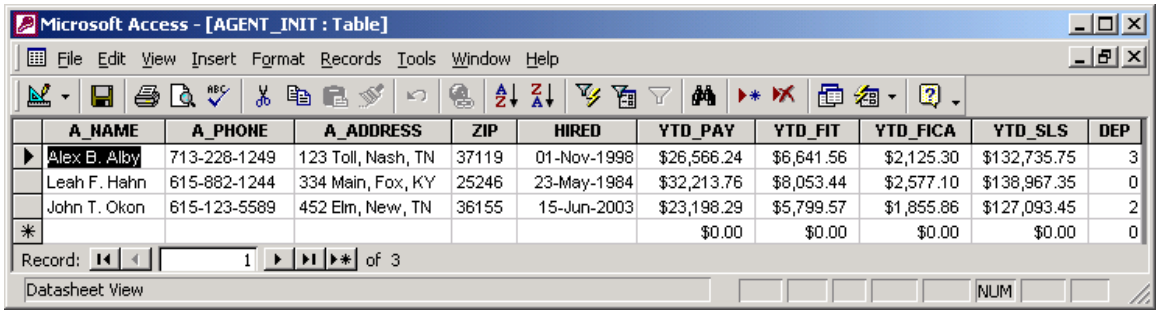
Microsoft Access - [CUSTOMER_INIT : Table]

File Edit View Insert Format Records Tools Window Help

C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	$100.00	05-Apr-200
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	$250.00	16-Jun-200
Kathy W. Smith	615-894-2285	125 Oak Ln, Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	$150.00	29-Jan-200
Paul F. Olowski	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	$300.00	14-Oct-200
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	$100.00	28-Dec-200
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	$850.00	22-Sep-200
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	$120.00	25-Mar-200
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	$250.00	17-Jul-200
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	$100.00	03-Dec-200
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	$500.00	14-Mar-200

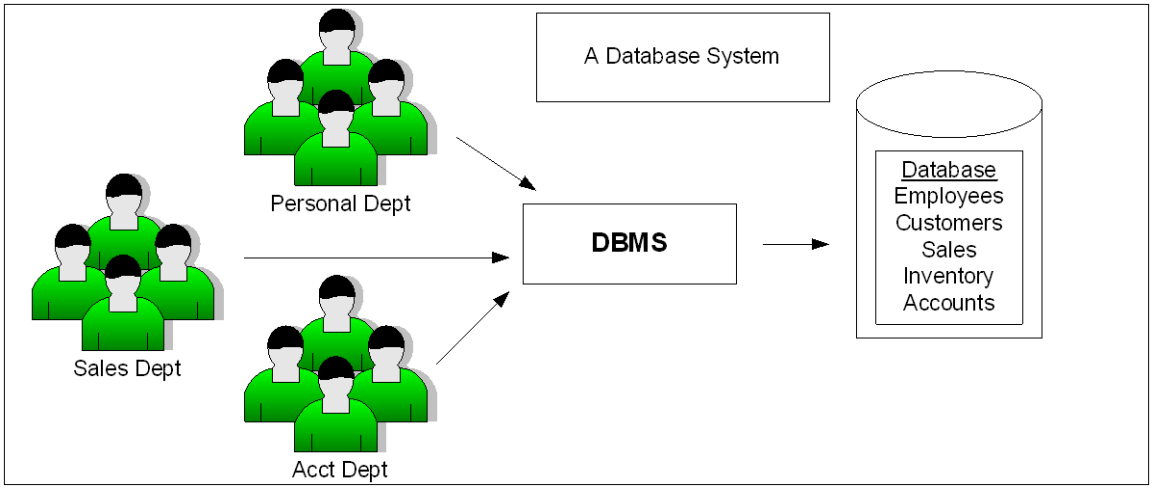
Record: 1 of 10

Datasheet View NUM



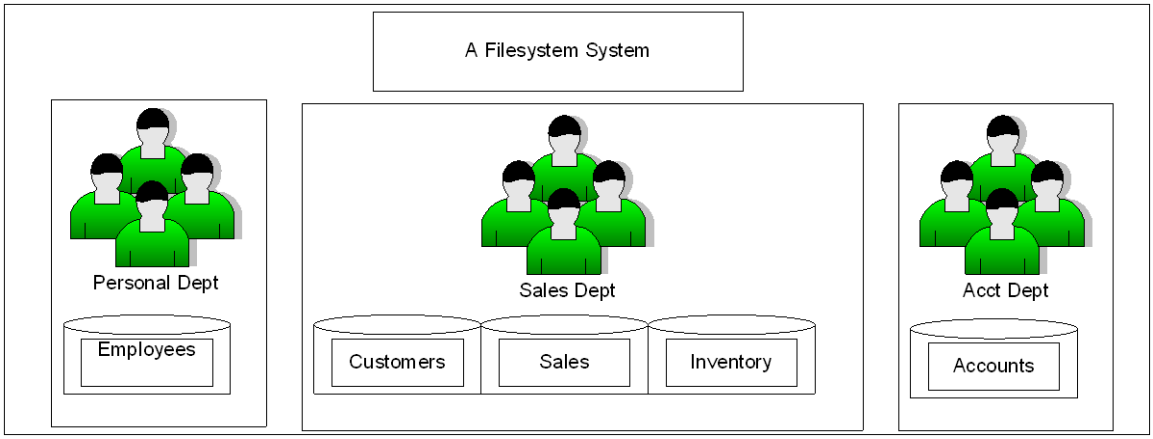
A_NAME	A_PHONE	A_ADDRESS	ZIP	HIRED	YTD_PAY	YTD_FIT	YTD_FICA	YTD_SLS	DEP
Alex B. Alby	713-228-1249	123 Toll, Nash, TN	37119	01-Nov-1998	$26,566.24	$6,641.56	$2,125.30	$132,735.75	3
Leah F. Hahn	615-882-1244	334 Main, Fox, KY	25246	23-May-1984	$32,213.76	$8,053.44	$2,577.10	$138,967.35	0
John T. Okon	615-123-5589	452 Elm, New, TN	36155	15-Jun-2003	$23,198.29	$5,799.57	$1,855.86	$127,093.45	2
					$0.00	$0.00	$0.00	$0.00	0

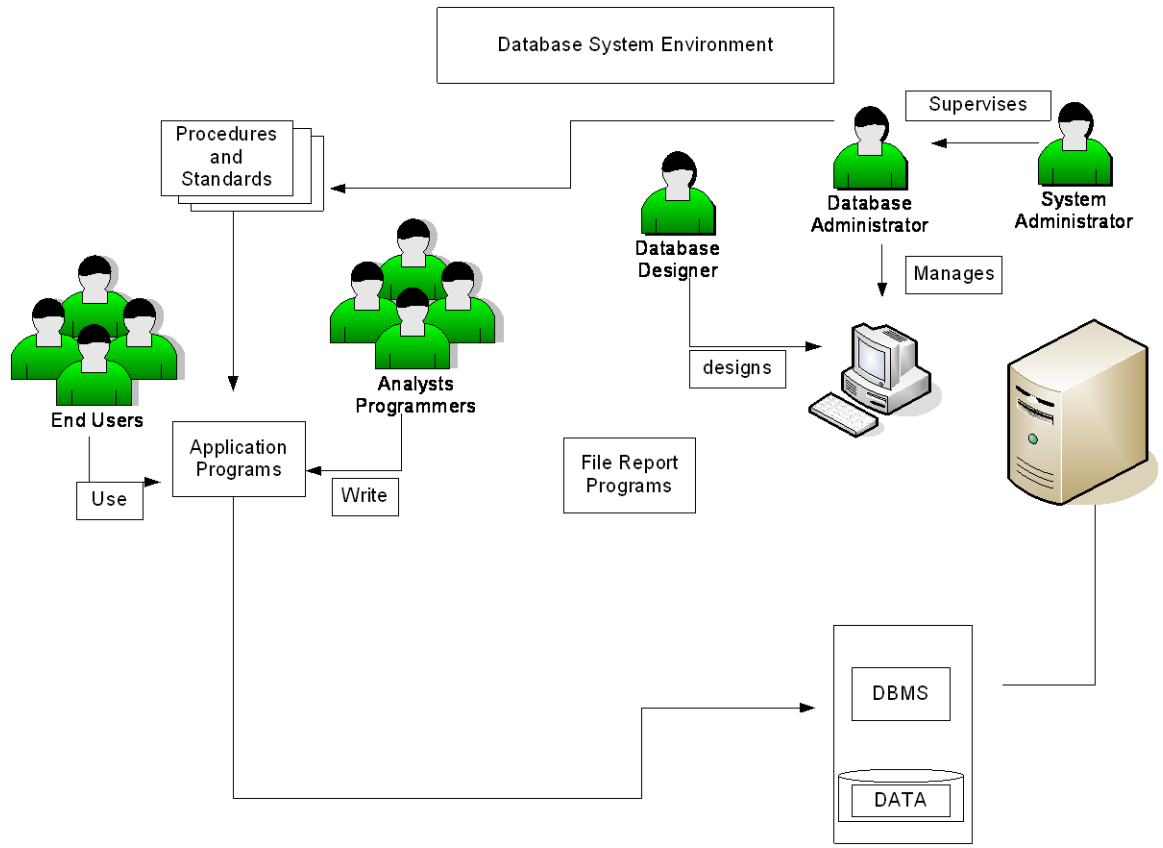
Contents of the Agents File



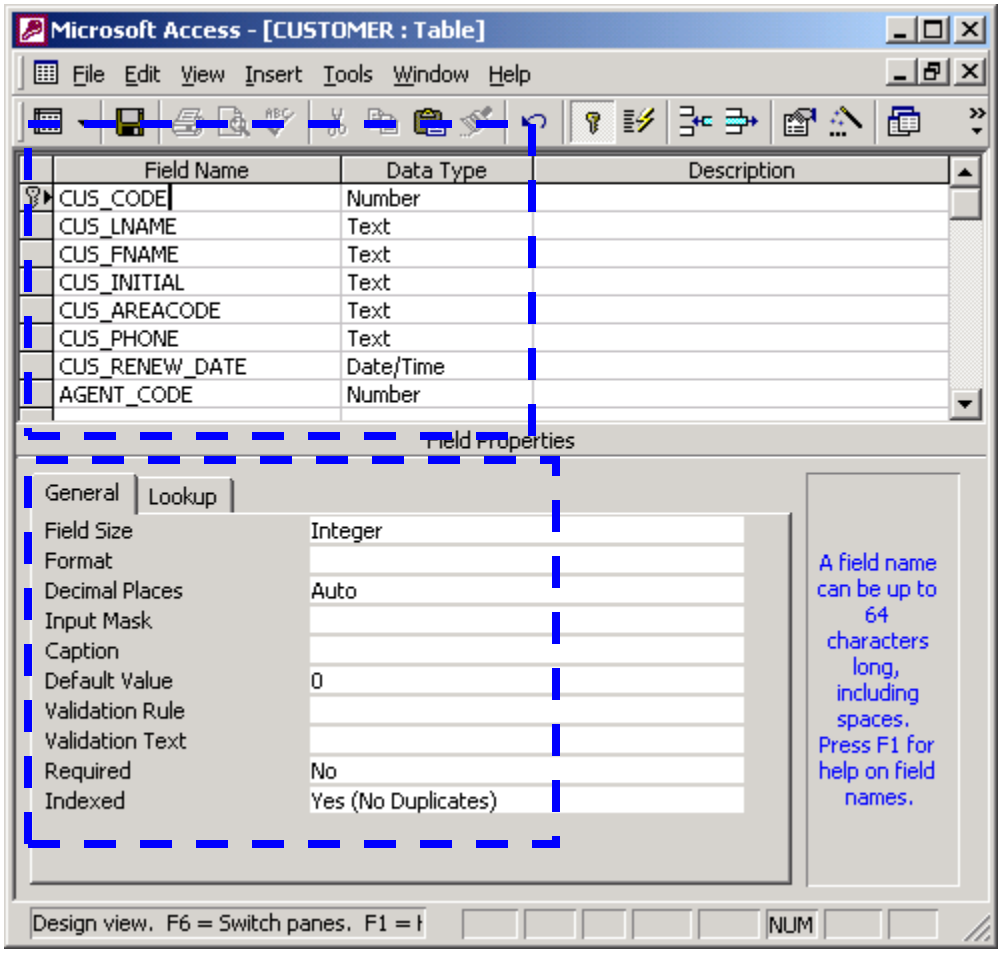
A Database System

Compared to




Database System Environment
Supervises
Procedures and Standards
Database Administrator
System Administrator
Database Designer
Manages
designs
End Users
Analysts Programmers
Application Programs
Use
Write
File Report Programs
DBMS
DATA

Microsoft Access Metadata



Key Notes to remember:

DBMS helps make data management much more efficient and effective. Because of this the following points are worth stressing:

- The DBMS helps create an environment in which end users have better access to more and better data.
- Wider access to well managed data promotes an integrated view of the organization's operations and a clearer view of the big picture.
- The probability of data inconsistency is greatly reduced in a properly designed database that is managed through a DBMS.
- A **DBMS** makes it possible to produce quick answers to AD HOC Queries!!

Database Design is the design of the DB structure that will be used to store and manage data – NOT the design of the DBMS software.

Proper DB design requires the DB designer to precisely identify the databases expected use.

A well designed database facilitates data management and becomes a valuable information generator.

A poorly designed database will most likely become a breeding ground for redundant data.

Redundant data is unnecessarily duplicated data.

Redundant data are often the source of difficult to trace information errors.

A poorly designed database tends to generate errors that are likely to lead to BAD decisions and BAD decisions can lead to the failure of an organization.

File systems – A way of managing data that are now largely obsolete. There are several good reasons for studying them in some detail.

- An understanding of the relatively simple characteristics of file systems makes the complexities of a DB design easier to understand.
- An awareness of the problems the plagued file systems can help you avoid such pitfalls with DBMS software.
- If you intend to convert an obsolete file system to a DB then knowledge of the file systems basic limitations will be useful.

Although the file systems method of organizing and managing data was a definite improvement over a manual system many problems and limitations became evident in this approach.

Understanding the shortcomings of the file system enables us to understand the development of the modern DB's.

3rd Generation Languages require the programmer to specify both WHAT must be done and HOW it is to be done. **Examples of this are COBOL, BASIC and FORTRAN.**

4th Generation Languages require the programmer to specify both WHAT must be done WITHOUT specifying HOW it is to be done. **Examples of this are SQL.**

Programming in a 3GL can be a time consuming, high skilled activity. This is because programmers must be familiar the physical file structure that is how and where the files are stored in the computer.

The need to write 3GL Programs to produce even the simplest reports makes **AD HOC Queries** impossible!!

DB Specialists and Managers are often harried and having numerous requests for reports means reports often take weeks and months.

Another problem with 3GL programming is that as the number of files in the system expands the System Administration becomes very difficult. Each file must have it's own file management system composed of programs that allow the user to do:

- Create the file structure
- Add data to the file
- Delete data from the file
- Modify the data contained in the file
- List the file contents

Making changes in an existing structure can be difficult in a file system environment. Changing just one field in the original file (ex: Customers file) requires a program that:

- Puts a new file structure into a special portion of the computers memory known as a buffer.
- Reads the record from the original file
- Transforms the original data to conform to the new structures storage requirements
- Deletes the original file
- Modifies all programs that use the file (ex: Customers file) to fit the revised file structure

To summarize the limitations of file system Data Management:

- It requires extensive programming
- System Administration can be complex and difficult
- It is difficult to make changes to existing structures
- Security features are likely to be inadequate

These limitations in turn lead to problems of structural and data dependency.

A change in any file's structure requires the modification of all programs using that file. Such modifications are required because the file system exhibits **Structural Dependency** that is access to a file is **dependent** on its structure.

Structural Independence exists when it is possible to make changes in the DB Structure **without affecting** the application programs ability to access the data.

Data Redundancy—A condition that exists when the data environment contains redundant—unnecessarily duplicated— data.

Uncontrolled **Data Redundancy** sets the stage for:

- **Data Inconsistency**—A condition in which different versions of the same data yield different (inconsistent) results.
- Data that display inconsistencies are also referred to data that lack data integrity.
- Data Anomilies develop when all the required changes in the redundant data are not made successfully. (Ex: Modification Anomalies, Insertion Anomalies, Deletion Anomalies)

Database System—An organization of components that define and regulate the collection, storage, management, and use of data in a database environment.

Database System is composed of the 5 major parts shown below:

1. **Hardware** – Refers to all the systems physical devices.
2. **Software** – Refers to the collection of programs used by the computers within the DB System. Examples are:
    - **Operating System Software**: DOS, MS, LINUX, UNIX…
    - **DBMS Software** manages the DB within the DB environment: Access, SQL Server, Oracle, UDB…
    - **Application Programs and Utility Software** are use to manipulate the data in the DBMS. They Generate Reports, tabulations and other information to facilitate decision making.

3. **People** – This component includes all users of the DB System. There are 5 types of users in a DB System:
    - **Systems Administrator (SYSADM)**—The person responsible for coordinating the activities of the data processing function. The systems administrator also manages the use of database software in a data processing (DP) department; solicits and evaluates database designs, coordinates the development of applications based on the data resource, and assigns the right to manage the database(s) to selected individuals. The systems administrator also coordinates the activities of all database administrators (DBAs) in a multidatabase operation.
    - **Database Administrator (DBA)**—Person responsible for the planning, organization, control and monitoring of the centralized and shared corporate database. The DBA is the general manager of the database-administration department.
    - **Systems Analysts and Programmers** – Design and implement the applications programs. They design and create the data entry screens, reports and procedures through which end users access data.

- **End User** – People who use the application programs to run the organizations daily operations.

4. **Procedures –** They are the instructions and rules that govern the design and use of the DB Systems.
5. **DATA –** The word DATA covers the collection of facts stored in the DB.

DB Systems must be cost effective as well as tactically and strategically effective.

DB levels of system complexity are dedicated by the organizations activities and the environment within which those activities take place.

DB Technology in use is likely to affect the selection of a DB system.

A DBMS performs several important functions that guarantee the integrity and the consistency of the data in the DB.

DBMS Functions:

1. **Data Dictionary Management** – The DBMS stores the definitions of the data elements and their relationships (metadata) in the **Data Dictionary**.
2. **Data Storage Management** – The DBMS creates and manages the complex structures required for data storage.
3. **Data Transformation and Presentation** – The DBMS transforms the entered data to conform to the data structures that are required to store the data. It relieves is of the chore of distinguishing between **Logical** and **Physical** format.
   Data Independence—A condition that exists when data access is unaffected by changes in the physical data storage characteristics. The DBMS formats the physically retrieved data to make it conform to the users logical expectations.
4. **Security Management** – The DBMS creates a security system that enforces user security and data privacy within the DB. Security rules determine which users can access the DB, which data items each user may access and which data operations (read, add, modify, delete…) the user may perform.
5. **Multi-User Access Control –** The DBMS creates the complex structures that allow multiple users access to the data. In order to provide data integrity and data consistency the DBMS uses sophisticated algorithms to ensure that multiple users can access the DB concurrently without compromising the integrity of the DB.
6. **Backup and Recovery Management –** The DBMS provides backup and data recovery procedures to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the **DBA** to perform routine and special backup and restore operations.
7. **Data Integrity Management –** The DBMS promotes and enforces integrity rules to eliminate data integrity problems. Thus minimizing data redundancy and

maximizing data consistency. The Data Relationships stored in the Data Dictionary are used to enforce data integrity.

8. **Database Access Languages and Application Programming Interfaces –** the DBMS provides data access through a query language.
    - A query Language is a Non Procedural Language (4GL)
    - The DBMS query language contains two components:
        - **Data Definition Language (DDL)**—The language that allows a database administrator to define the database structure, schema, and subschema.
        - **Data Manipulation Language (DML)**—The language (set of commands) that allows the end user to manipulate the data in the database (Select, Insert, Update, Delete).
    - The DBMS also provides data access to programmers via Procedural Languages (3GL) ex: COBAL, C, PASCAL, VB…

9. **Database Communication Interface –** Current generations of DBMS's provide communication interfaces designed to allow the DB to accept end-user request within a computer network environment. Example: The DBMS might provide communications functions to access the DB through the Internet using Web Browsers such as Netscape Navigator and Internet Explorer.
   In this environment communications can be accomplished in several ways:
    - End users can generate answers to queries by filling in screen forms through their preferred Browser.
    - The DBMS can automatically publish predefined reports on the Internet using a Web format that enables and Web user to browse it.
    - The DBMS can connect to the 3rd party to distribute information via email or other productivity applications such as Lotus Notes.

The DBMS allows chores to be done and performed without the tedious and time consuming programming required in the File Management System.

The role of the DB Specialist or the DB Manager changes from an emphasis on programming to a focus on the broader aspects of managing the organizations data resources and on administration of the complex software itself.

Because the File Systems DB Manager performs broader managerial functions in a database environment they might be promoted to **System Administrators – SYS DBAs**

The availability of a DBMS makes it possible to tackle far more sophisticated uses of the DATA resources, if the DB is designed to make use of that available power.

Summary Chap 1

- Information is derived from data, which are usually stored in DB.
- To implement a DB and manage its contents you need commercial software known as a DBMS.
- Database design defines the DB Structure.
- The DBMS stores the facts about the structure in the DB itself. The DB contains the data you have collected and the **"data about the data"** known as **Metadata**.
- Good Database design is important because even a good DBMS will perform poorly with a poorly designed DB.
- DB's were preceded by File Systems.
- Because File Systems lack a DBMS, file management becomes difficult as the file system grows.
- Each file requires it's own set of basic data management programs and because files are usually "owned" by those who commission them, the number of files tends to grow.
- Many of the files in a File System often contain redundant data, thus leading to data inconsistency, data anomalies and a lack of data integrity.
- Because each file can be used by many application programs a mature file system might have generated hundreds or even thousands of programs.
- Serious file system data management problems usually stem from data dependency. Access to any file is dependant on the data characteristics and storage formats therefore even a minor change in data structure with a file requires that all programs accessing that file must be modified too.
- DB management systems were developed to address the file systems inherent weaknesses. Rather than depositing data within independent files. A DBMS presents the DB to the End User as a single repository. This arrangement promotes data sharing thus at least potentially eliminating the **Islands of Information** problem.
- DBMS enforces data integrity, eliminates redundancy and promotes data security.

Chapter 2

Glossary Table

American National Standards Institute (ANSI)—The group that accepted the **DBTG** recommendations and augmented database standards in 1975 through its **SPARC** committee.
Attribute—A characteristic of an entity or object. An attribute has a name and a data type. See also *instance variable and entity*.
Business Rules—Narrative descriptions of policies, procedures, or principles within an organization. Examples: A pilot cannot be on duty for more than ten hours during a twenty-four hour period. A professor may teach up to four classes during any one semester.
Chen model—Entity relationship diagram described by Peter Chen, characterized by its use of diamonds to depict relationships and rectangles to depict entities.
class—A collection of like objects with shared structure (attributes) and behavior (methods). A class encapsulates the object's data representation and method's implementation. Classes are organized in a class hierarchy. See also *abstract data types*.
Class Hierarchy—The organization of classes in a hierarchical tree where each "parent" class is a *superclass* and each "child" class is a *subclass*.
Conceptual Model—An abstract view of the data as seen by high-level managers and database designers. Describes the main data objects, avoiding details.
CONFERENCE ON DATA SYSTEMS LANGUAGES (CODASYL)—A group originally formed to help standardize COBOL; its DBTG subgroup helped to develop database standards in the early 1970s.
Connectivity—Describes the classification of the relationship between entities. Classifications include 1:1, 1:M and M:N.
Crow's Foot Model—A notation of the entity relationship diagram using a three-pronged symbol to represent the "many" sides of the relationship.
Data Model—A representation, usually graphic, of a complex "real-world" data structure. Data models are used in the database design phase of the database life cycle. See also *database models*.

Database Task Group (Dbtg)—A CODASYL committee that helped develop database standards in the early 1970's. See also *CODASYL*.
Entity—Something about which you want to store data; typically a person, place, thing, concept or event. See also *attribute*.
Entity Instance—A term used in ER modeling to refer to a specific table row. Also known as an *entity occurrence*.
Entity occurrence—See *entity instance*.
Entity Relationship (ER) Diagram—A diagram that depicts an entity relationship model's entities, attributes and relations. It also displays connectivity and cardinality.
Entity Relationship (ER) Model—A data model developed by P. Chen in 1975. It describes relationships (1:1, 1:M, M:N) among entities at the conceptual level with the help of ER diagrams.
Extended Relational Data Model (ERDM)—A model that includes the object oriented model's best features in an inherently simpler relational database structural environment.
External Model—The application programmer's view of the data environment. Given its business-unit focus, it works with a data subset of the global database schema.
Generalized Update Access Method (GUAM)—Developed by North American Rockwell in the late 1960s to control file system redundancies; a precursor to database development.
Hardware Independence – Means that the model does not depend on the hardware used in the implementation of the model.
Hierarchic Sequence—See *preorder traversal*.
Hierarchical Database Model—No longer a major player in the current database market; important to know, however, because the basic concepts and characteristics form the basis for subsequent database development. This model is based on an "upside-down" tree structure in which each record is called a segment. The top record is the root segment. Each segment has a 1:M relationship to the segment directly below it. See also *child*, *parent*, *root* and *segment*.
Hierarchical Path—Ordered sequence of segments that must be accessed by the hierarchical DBMS in order to retrieve a given segment.
Hierarchical Structure—An ordered data arrangement that (logically) resembles an "upside-down" tree.

Information Management System (IMS)—A hierarchical database model based on GUAM and developed jointly by North American Rockwell and IBM. IMS became the hierarchical database standard in the 1970s.
Inheritance—In the object oriented data model, the ability of an object to inherit the data structure and methods of the classes above it in the class hierarchy. See also *class hierarchy* and *single inheritance*.
Internal Model—In database modeling, refers to a level of data abstraction that adapts the conceptual model to a specific DBMS model for implementation.
Many-To-Many (M:N or M:M) Relationships—One of three types of relationships (associations among two or more entities) in which one occurrence of an entity is associated with many occurrences of a related entity, and one occurrence of the related entity is associated with many occurrences of the first entity.
Member (Network Database)—One record type in network database terminology; a member record is equivalent to the hierarchical model's child.
Method—In the OO data model, a named set of instructions to perform an action. Methods represent real-world actions. Methods are invoked through messages.
Model—A description or analogy used to visualize something that cannot be directly observed; simplified abstractions of real-world events or conditions.
Network Model—A data model created to represent complex data relationships more effectively than the hierarchical model could, to improve database performance, and to impose a database standard.
Object—An abstract representation of a real-world entity that has a unique identity, embedded properties, and the ability to interact with other objects and with itself.
Object Oriented Data Model (OODM)—A data model based on object-oriented concepts. Provides support for new user-defined data types, inheritance, polymorphism, encapsulation, and so on.
Object Oriented Database Management System (OODBMS)—Software used to better manage data found within an object oriented database model.
Object / Relational Database Management System (O/RDBMS)—DBMS based on the ERDM

One-To-Many (1:M) Relationship—One of three types of relationships (associations among two or more entities) which are used by data models. In a 1:M relationship, one entity instance is associated with many instances of the related entity.
One-To-One (1:1) Relationship—One of three types of relationships (associations among two or more entities) which are used by data models. In a 1:1 relationship, one entity instance is associated with only one instance of the related entity. Owner (network database)—One record type in network database terminology; an owner record is equivalent to the hierarchical model's parent.
Physical Model—A model in which the physical characteristics (location, path, and so on) are described for the data. Both hardware- and software-dependent. See also *physical design*.
Pointer—A reference device that "points" to the location of data within the computer data-storage medium.
Preorder Traversal—A "left-list" path within a hierarchical database. The left-list refers to the order (always starting from the left of the hierarchical tree) in which segments are referenced. Also known as the *hierarchical sequence*.
Relation—In the relation model, a term used to describe a set of associated attributes. Normally, relations will become relational tables.
Relational Database Management System (RDBMS)—A collection of programs that manages the complexity of a relational database. The RDBMS software translates the user's logical requests (queries) into commands that physically locate and retrieve the requested data. A good RDBMS also creates and maintains a data dictionary (system catalog) to help provide data security, data integrity, concurrent access, easy access, and system administration to the data in the database through a query language (SQL) and application programs.
Relational Model—Developed by E. F. Codd (of IBM) in 1970, it represents a major breakthrough for users and designers because of its conceptual simplicity. The relational model produced an "automatic transmission" database to replace the "standard transmission" databases that preceded it.
Relational Schema—The description of the organization of a database as seen by the database administrator that shows connecting fields and the relationship type.
Relationship—An association between entities defined by a diamond-shaped symbol in an ER diagram. Its degree defines the number of related entities: unary, binary, ternary, or higher.
Root—The starting component of a hierarchical database tree

Schema—A group of database objects (tables, indexes, views, queries, etc.) that are related to each other. Usually a schema belongs to a single user or application.
Segment—The equivalent of a file record type in the hierarchical database model.
Semantic Data Model (SDM)—The first of a series of data models that more closely represented the real world, modeling both data and their relationships in a single structure known as an object. The **SDM**, published in 1981, was developed by M. Hammer and D. McLeod.
Set—In the network model, a description of a 1:M relationship between an owner record type and a member record type.
Software Independence—A property of any model or application that does not depend on the software used to implement it. See also *software-dependent*.
Standards Planning And Requirements Committee (SPARC)—The ANSI committee that augmented database standards in 1975.
Structured Query Language (SQL)—A powerful and flexible relational database software language composed of commands that enable users to create database and table structures, perform various types of data manipulation and data administration, and query the database to extract useful information.
Subschema—A subset of a schema. The subschema defines the portion of the database as "seen" by the application programs that use it. See also *schema*.
Table—A (conceptual) matrix comprising intersecting rows (entities) and columns (attributes) that represent an entity set in the relational model. Also called a *relation*.
Universal Database Servers—Another name for an object/relational product used in the object/relational database model.

A Data Model is the relatively simple representation, usually graphical, of complex real-world data structures. This is also known as the **Database Model**.

A good DB design is the foundation for good applications and a good DB design uses an appropriate data model as its foundation.

An **Entity** is something about which you want to store data; typically a person, place, thing, concept or event. But they may also be abstractions such as flight routes or musical concerts. (You would store this info in a table)

An **Attribute** is a characteristic of an entity or object. An attribute has a name and a data type. (This would be a field in a table)

A **Relationship** is an association between entities defined by a diamond-shaped symbol in an ER diagram. Its degree defines the number of related entities: unary, binary, ternary, or higher.
Data models use 3 types of Relationships:

- One to Many (1:M or N)
- Many to Many (M:N)
- One to One (1:1)

Business Rules are narrative descriptions of policies, procedures, or principles within an organization. Examples: A pilot cannot be on duty for more than ten hours during a twenty-four hour period. A professor may teach up to four classes during any one semester.

Business Rules are essential to DB Design for several reasons:

- They help standardize the companies view of data
- They constitute a communications 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 processors
- They allow the designer to develop appropriate relationship participation rules and constraints

---

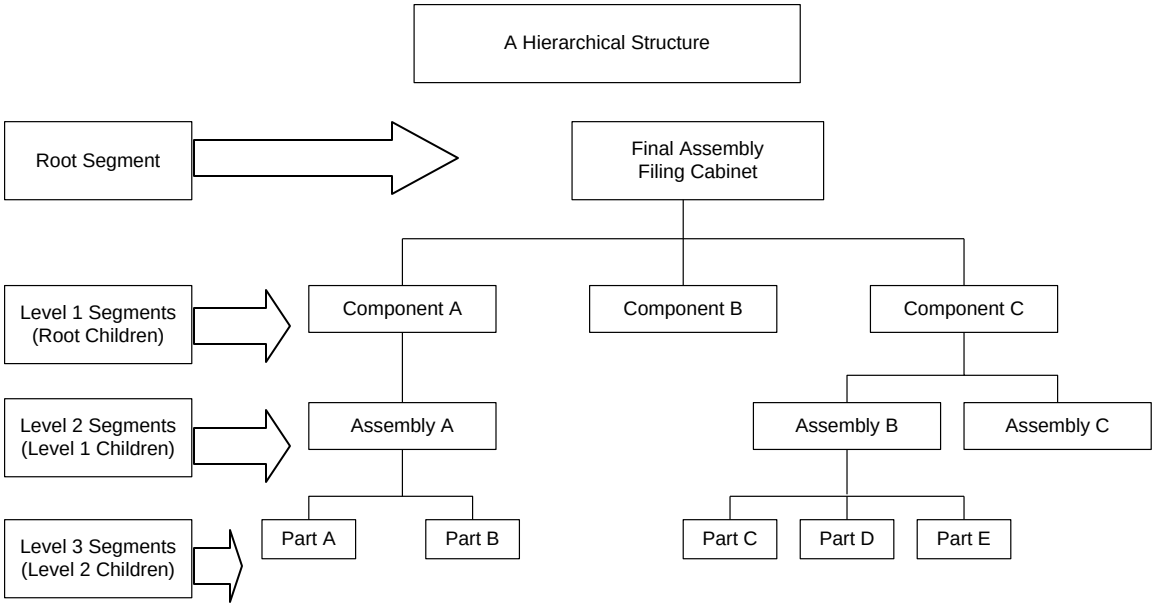
Hierarchical Model

North American Rockwell was a contractor for the Apollo project. They developed software known as **Generalized Update Access Method (GUAM)** in the late 1960s to control file system redundancies.
IBM then later joins Rockwell and they create **Information Management System (IMS),** which is a hierarchical database model.

IMS became the hierarchical database standard in the 1970s.

It is a **Hierarchical Model**, which is no longer a major player in the current database market. You should understand a few of its characteristics for these reasons:

- It's basic concept forms the basis for subsequent database development
- It's limitations lead to a different way of looking at DB Design
- Some of the basic concepts show up in current Data Models



A **segment** is the equivalent of a file systems record type.
The Hierarchical Database is a collection of records that logically organize to conform to the upside down tree structure shown here.

Within the Hierarchy the top layer (the **Root**) is perceived as the parent of the segment directly beneath it. For example the **Root** segment is the **parent** of **Level 1** segments, which in turn are the **parents** of the **Level 2** segments.

In short:

- Each Parent can have many Children
- Each Child has only One Parent.

It is therefore easy to trace the DB components and the 1:M relationships among them.

This ordered sequencing of segments tracing the Hierarchical structure is called the **Hierarchical Path.**

For example the path to the segment labeled “Part D” can be traced this way:

Final Assembly ➔ Component A ➔ Assembly A ➔ Part A➔ Part B ➔ Component B ➔ Component C ➔ Assembly B ➔ Part C ➔ Part D

Note that the path traces all segements from the root starting at the left-most segment. This left-list path is known as the **Preorder Traversal** or the **Hierarchical Sequence**.

The Network Model

Network Model—A data model created to represent complex data relationships more effectively than the hierarchical model could, to improve database performance, and to impose a database standard.

CONFERENCE ON DATA SYSTEMS LANGUAGES (CODASYL)—A group originally formed to help standardize COBOL; its DBTG subgroup helped to develop database standards in the early 1970s.

American National Standards Institute (ANSI)—The group that accepted the **DBTG** recommendations and augmented database standards in 1975 through its **SPARC** committee.

Database Task Group (Dbtg)—A CODASYL committee that helped develop database standards in the early 1970's. See also *CODASYL*.

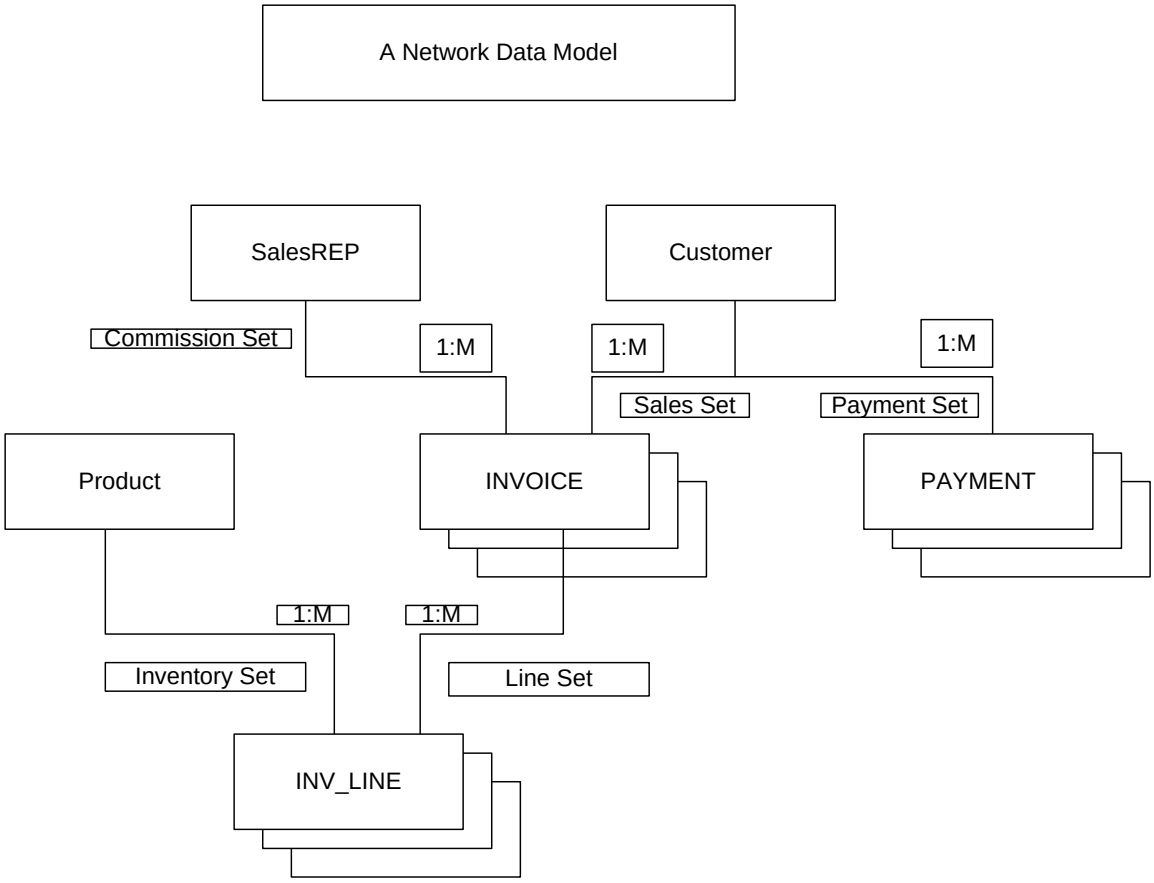
The **DBTG** was charged to define standard specifications for an environment that would facilitate DB creation and manipulation.

The final DBTG report contained specifications for the three crucial DB components:

1. **The Network Schema**, the conceptual organization of the entire DB as viewed by the DB Admin. The Schema includes a definition of the DB name, the record type for each record, and the components that make up those records.
2. **The Sub-Schema**, which defines the portion of the DB "seen" by the application programs that actually produce the desired information from the data contained within the DB. The existence of sub-schema definitions allows all application programs to simply invoke the sub-schema required to access the appropriate files.
3. A **Data Management Language (DML)** to define the data characteristics and the data structure in order to manipulate the data.

To produce the desired standardization for each of the three components the DBTG specified 3 distinct data management language components:

1. A **Schema Data Definition Language (DDL),** which enables the DB Admin to define components that will be used.
2. A **Sub-Schema DDL,** which allows application programs to define the database components that will be used.
3. A **DML** to manipulate the DB records.



In Network DB Terminology:

Set = Relationship

1. **Owner Record -** the equivalent of the **Hierarchical Models Parent**
2. **Member Record** – the equivalent of the **Hierarchical Models Child**

The Relational Model

First developed by E.F. Codd (of IBM) in 1970 and is considered by many the Automatic Transmission of DB's.

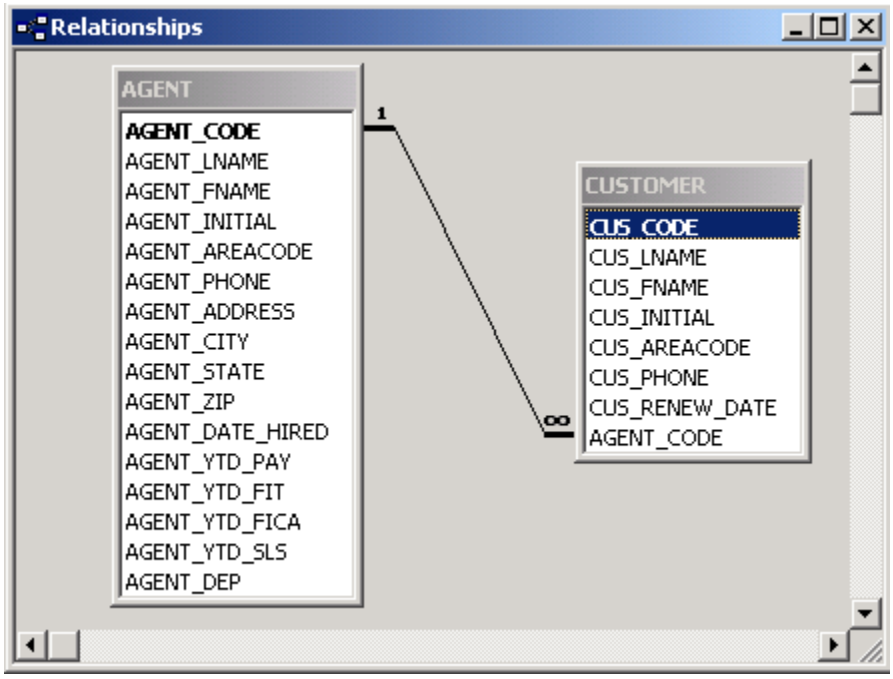
The Relational data model is implemented through a very sophisticated **Relational DBMS.**

The most important advantage of the **RDBMS** is its ability to let the user/designer operate in a human logical environment. The **RDBMS** manages all of the complex physical details. Thus the **Relational DB** is perceived by the user to be a collection of tables in which the data are stored.

Table – Matrix consisting of a series of row/column intersections. Tables are also called Relations and are related to each other by sharing a common entity characteristic.

Example:

The Customer table might contain a sales agents number that is also contained in the Agents file.



Relational Schema – A visual representation of the relational DB's entities, the attributes within those entities and the relationships between those entities.

For most Relational DB Software the query language is **Structured Query Language (SQL)**.

Entity Relational Model

Entity Relationship (ER) Model—A data model developed by P. Chen in 1975. It describes relationships (1:1, 1:M, M:N) among entities at the conceptual level with the help of ER diagrams.

Entity Relationship (ER) Diagram—A diagram that depicts an entity relationship model's entities, attributes and relations. It also displays connectivity and cardinality.

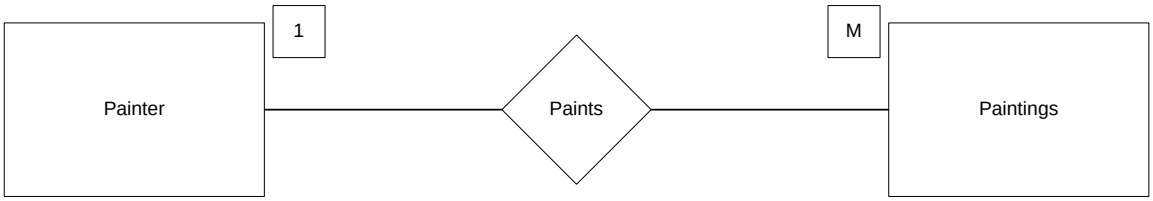
An **Entity** is represented in the **ERD** by a **Rectangle**. Each **Row** in the **Relational Table** is known as an **Entity Instance or Entity Occurrence**.

A collection of Entities is known as an Entity Set.
Example: an Agent file has 3 agents (entities) in the agent Entity Set.

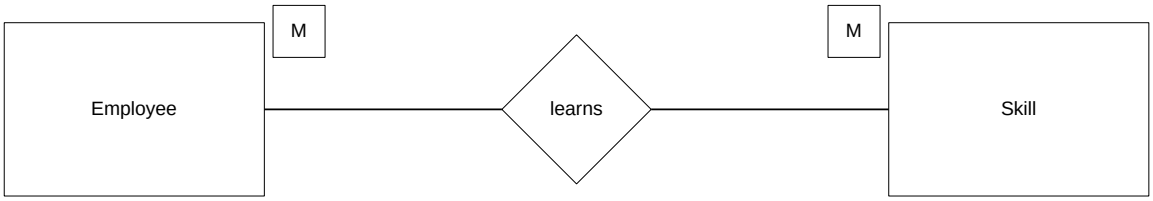
ERD Models use the term **Connectivity** to label the types of **Relationships**.

The Basic CHEN ERD

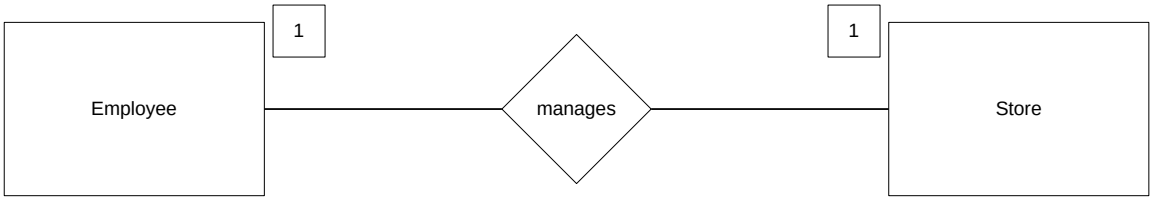
One to Many (1:M) Relationship
A painter can paint many paintings.
Each painting is painted by one painter



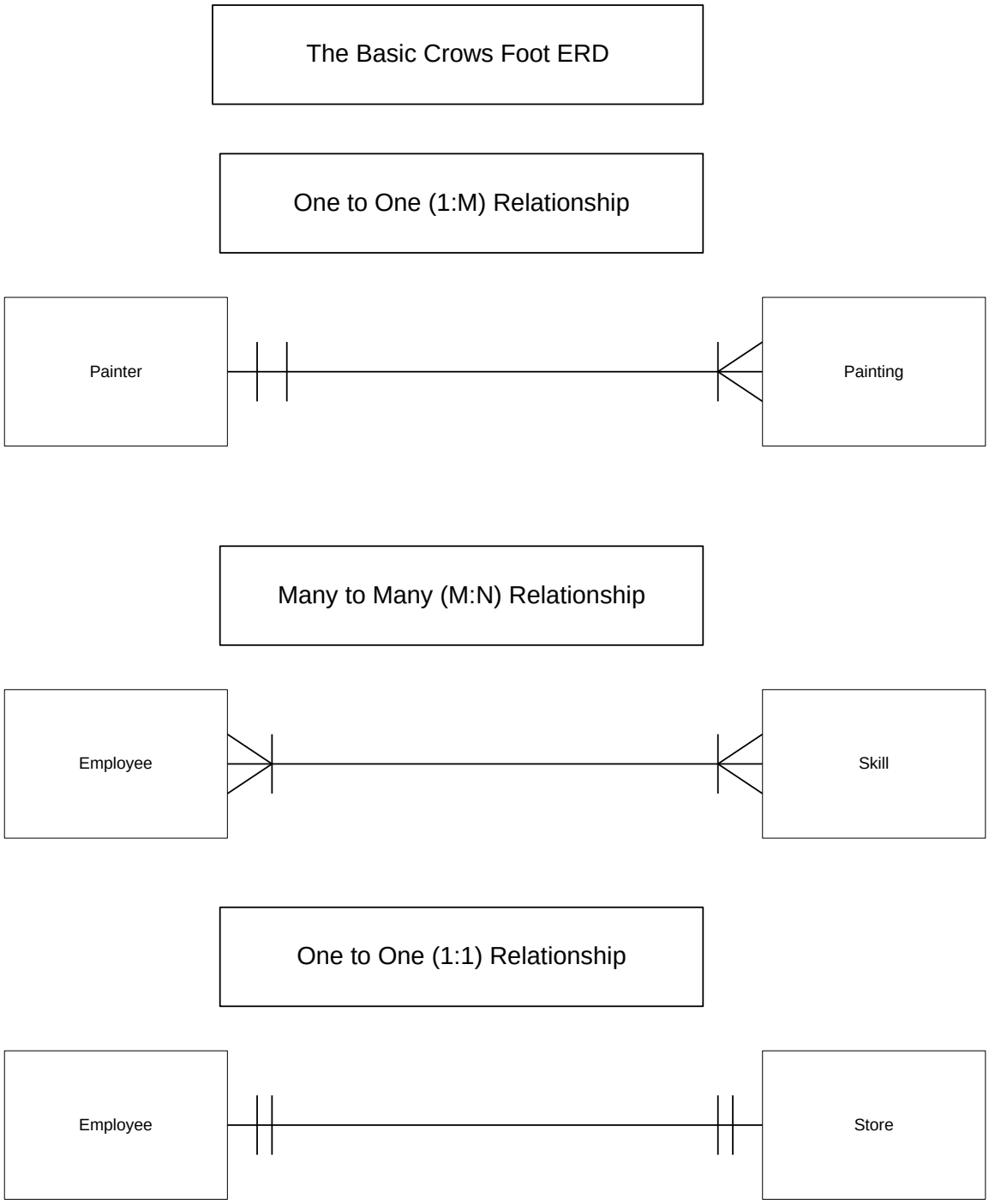
Many to Many (M:N) Relationship
An Employee can learn Many Skills
Each Skill is learned by many employees



One to One (1:1) Relationship
An Employee manages One Store
Each Store is managed by One Employee



A more current version of the **ERD** is the **Crow's Foot Model** derived from the 3 prong symbol used to represent the **Many** side of the relationship. There is NO diamond to indicate the relationship.



Chen and the Crows Foot are the basic ERD's used.

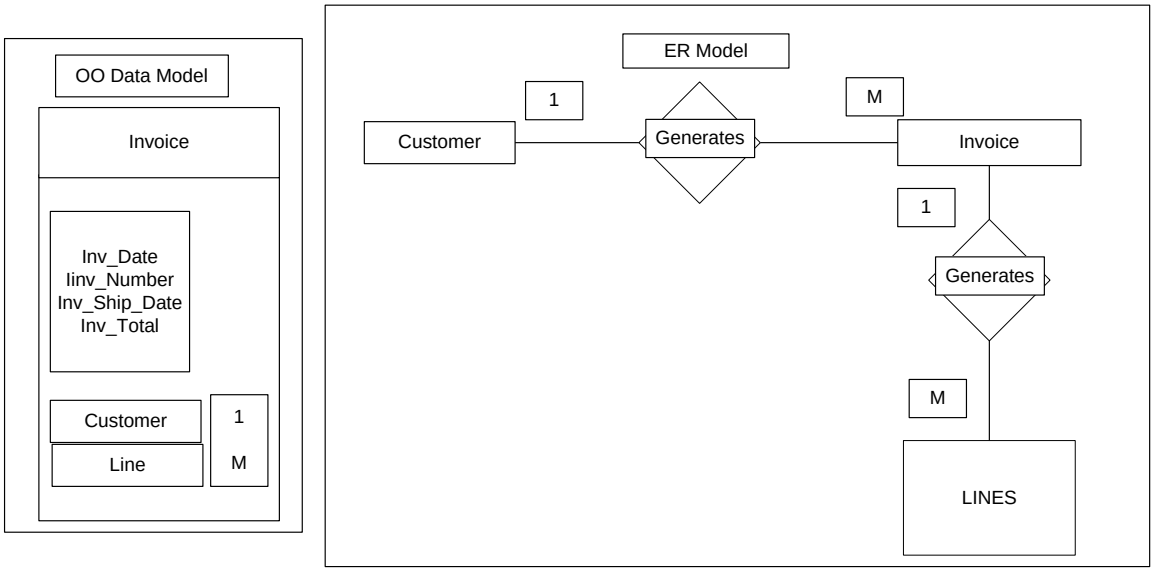
The Object Oriented Model

The **Semantic Data Model (SDM**) modeled both data and their relationships in a single structure known as an **Object**.

Object Oriented Data Model (OODM)—A data model based on object-oriented concepts. Provides support for new user-defined data types, inheritance, polymorphism, encapsulation, and so on.

In turn the **Object Oriented Database Management System (OODBMS)**—Software used to better manage data found within an object oriented database model.

An **OODM** reflects a very different way to define and use **Entities**. Like the relational models entity, an object described by it's factual content. BUT quite unlike an entity, an Object includes information about relationships between the facts within the object, as well as well as its information about its relationship with other objects.



Example here shows an OODM model and the same Model as a Chen ER model.

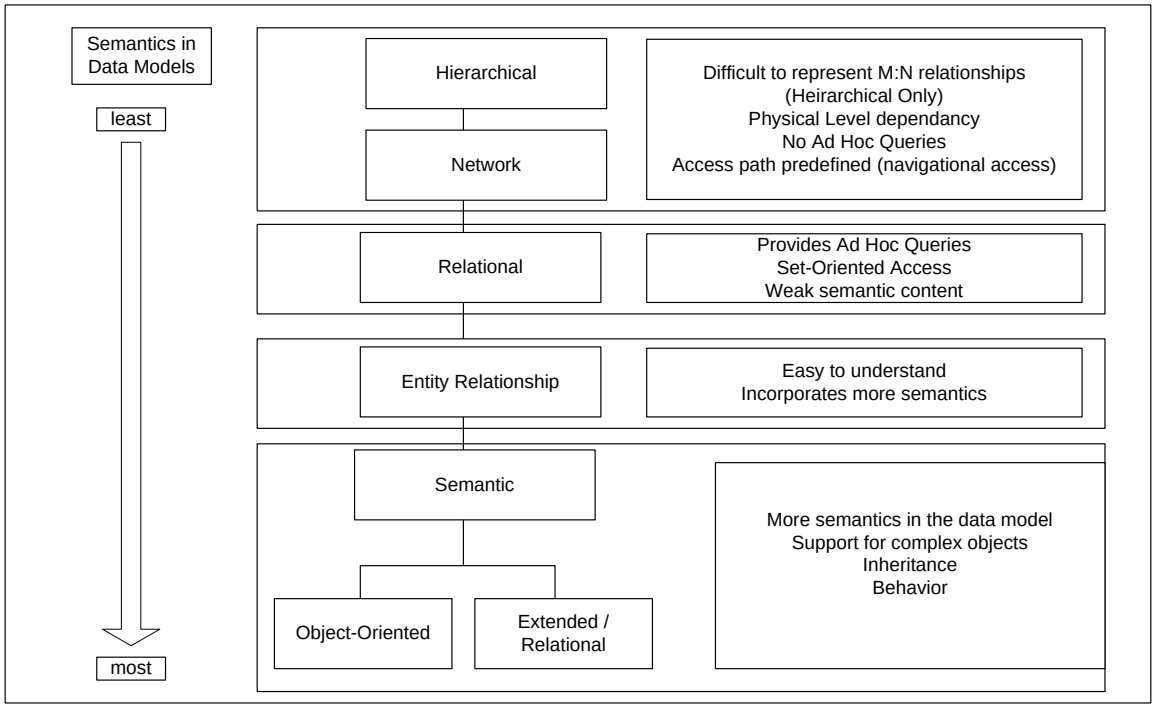
An Object Oriented Data Model is based on the following components.

- An Object is an abstraction of a real world entity. An Object may be considered equivalent to an ER Models Entity. Ex: An Object Class of **PERSON**.
- Object Instance is a particular real world occurrence of an Object. Ex: Brianna Sheridan is an **Instance** of the object class **PERSON.**
- **Objects** that share similar characteristics are grouped in **Classes.** A Class is a collection of similar objects with shared structure (attributes) and behaviour (methods). Ex: A **Class's method** represents a real world action such as:
    - Finding a persons name
    - Changing a persons name

Inheritance is the ability of an **Object** within the class hierarchy to **inherit** the **attributes and methods** of the **classes** above it.

Example: We create 2 Classes **Customer** and **Employee** as subclasses from the class **Person.** In this case **Customer** and **Employee** will inherit all attributes and methods from **Person.**

The Development of Data Models



Chapter 3

Glossary Table

abstract data types (ADT)—Data types that describe a set of similar objects with shared and encapsulated data representation and methods. An abstract data type is generally used to describe complex objects. See also *class*.
Attribute Domain—See *domain*.
Bridge Entity—See *composite entity*.
Candidate Key—See *key*.
Composite Entity—An entity designed to transform an M:N relationship into two 1:M relationships. The composite entity's primary key comprises at least the primary keys of the entities that it connects. Also known as a *bridge entity*.
Data Dictionary—A DBMS component that stores metadata – data about data. Thus, the data dictionary contains the data definition as well as its characteristics and relationships. A data dictionary may also include data that are external to the DBMS. See also *active data dictionary* and *passive data dictionary*.
Determination—The role of the key. In the context of a database table, the statement "A determines B" indicates that knowing the value of attribute A means that you can look up (determine) the value of attribute B.
Domain—Used to organize and describe an attribute's set of possible values.
Entity Integrity—Property of a relational table that guarantees that each entity have a unique value in a primary key and that there are no null values in the primary key.
equiJOIN—A join operator that links tables based on an equality condition that compares specified columns of the tables.
Flags—Special codes implemented by designers to prevent nulls by bringing attention to the absence of a value in a table.
Foreign Key—See *key*.
Full Functional Dependence—A condition in which an attribute is functionally dependent on a composite key but not on any subset of that composite key.

Homonyms—Indicates the use of the same name to label different attributes; should generally be avoided. Some relational software automatically checks for homonyms and either alerts the user to their existence or automatically makes the appropriate adjustments. See also *synonym*.
Index—An ordered array composed of index key values and row ID values (pointers). Indexes are generally used to speed up data retrieval.
Join Columns—Term used to refer to the columns used to join two tables. The join columns generally share similar values.
Key—An entity identifier based on the concept of functional dependence; may be classified as follows: ***Superkey:*** An attribute (or combination of attributes) that uniquely identifies each entity in the table. ***Candidate key:*** A minimal superkey, that is, one that does not contain a subset of attributes that is itself a superkey. ***Primary key:*** A candidate key selected as a unique entity identifier. ***Secondary key:*** A key that is used strictly for data retrieval purposes. For example, a customer is not likely to know his or her customer number (primary key) but the combination of last name, first name, initial and telephone number is likely to make a match to the appropriate table row. ***Foreign key:*** An attribute (or combination of attributes) in one table whose values must match the primary key in another table or whose values must be null.
Key Attributes—The attribute(s) that form a primary key. See also *prime attribute*.
Left Outer Join—In a pair of tables to be joined, a left outer join yields all the rows in the left table, including those that have no matching values in the other table. For example, a left outer join of Customer with Agent will yield all the Customer rows, including the ones that do not have a matching Agent row. See also *outer join*.
Linking Table—A table that implements a composite entity. See also *composite entity*.
Natural Join—A relational operation that links tables by selecting only the rows with common values in their common attribute(s).
NULL—The absence of an attribute value. Note: a null is not a blank.
Primary Key—See *key*.

Referential Integrity—A condition by which a dependent table's foreign key must have either a null entry or a matching entry in the related table. Even though an attribute may not have a *corresponding* attribute, it will be impossible to have an invalid entry.
Relational Algebra—A set of mathematical principles that form the basis of the manipulation of relational table contents; comprises eight main functions: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE.
Right Outer Join—In a pair of tables to be joined, a right outer join yields all the rows in the right table, including the ones with no matching values in the other table. For example, a right outer join of CUSTOMER with AGENT will yield all the agent rows, including the ones that do not have a matching CUSTOMER row. See also *outer join*.
Secondary Key—A key that is used strictly for data retrieval purposes. For example, a customer is not likely to know his or her customer number (the primary key) but the combination of last name, first name, initial, and telephone number is likely to make a match to the appropriate table row.
Superkey—See *key*.
Synonym—The use of different names to identify the same object, such as an entity, an attribute, or a relationship; should (generally) be avoided. See also *homonym*.
System Catalog—A detailed system data dictionary that describes all objects in the database.
Theta Join—A join operator that links tables using an inequality comparison operator (<, >, <=, >=) in the join condition.
Tuple—A table row.
Union-Compatible—Two or more tables are union compatible if they share the same column names and the columns have compatible data types or domains.
Unique Index—An index in which the index key can only have one pointer value (row) associated with it.

Relational Database Model

Characteristics of a Relational Table

1. A table is perceived as a two-dimensional structure composed of rows and columns
2. Each table row (**tuple**) represents a single entity occurrence within the entity set
3. Each table column represents an attribute and each column has a distinct name
4. Each row/column intersection represents a single data value
5. All values in a column must conform to a single data format
6. Each column has a specific range known as an **Attribute Domain**
7. The order of the rows/columns is immaterial to the DBMS
8. Each table must have an attribute or a combination of attributes that uniquely identifies each row.

Some older DBMSs might impose the following naming constraints:

1. Table names are restricted to 8 characters
2. Column (attribute) names are limited to 10 characters
3. Column names cannot begin with a digit

Although various DBMSs can support different data types most support at lest the following:

1. Numeric
2. Character
3. Date
4. Logical (yes/no)

A **Key** consists of one or more attributes that determine other attributes.

Determination is the role of the key. In the context of a database table, the statement "A determines B" indicates that knowing the value of attribute A means that you can look up (**determine**) the value of attribute B.

Full Functional Dependence is a condition in which an attribute is functionally dependent on a composite key but not on any subset of that composite key.
The attribute B is functionally dependent on A *if* A determines B.

A **Composite Key** is composed of more than one attribute (fields)

An attribute (fields) that is part of a key is known as a **Key Attribute**

A **Superkey** is an attribute (or combination of attributes) that uniquely identifies each entity in the table.

Candidate key: **A minimal Superkey**, that is, one that does not contain a subset of attributes that is itself a Superkey. **NO REDUNDANCIES in the key!**

Primary key: A candidate key selected as a unique entity identifier.

Secondary key: A key that is used strictly for **Search Capabilities**. For example, a customer is not likely to know his or her customer number (primary key) but the combination of last name, first name, initial and telephone number is likely to make a match to the appropriate table row.

Foreign key*:* An attribute (or combination of attributes) in one table whose values must match the primary key in another table or whose values must be null.

Within a table each Primary Key value must be unique to ensure that each row is uniquely identified by the Primary Key. When this is the case the table is said to have **Entity Integrity**.

To maintain **Entity Integrity** a null value is not permitted in the Primary Key.

Referential Integrity—A condition by which a dependent table's foreign key must have either a null entry or a matching entry in the related table. Even though an attribute may not have a *corresponding* attribute, it will be impossible to have an invalid entry.

Integrity Rules

Entity Integrity	**Description**
Requirement	All Primary Key entries are unique and no part of a primary key may be null
Purpose	Guarantees that each entity will have a unique identity and ensures that foreign key values can properly reference Primary Key Values
Example	No Invoice can have duplicate Numbers nor can they be NULL
Referential Integrity	**Description**
Requirement	A foreign Key may have either a NULL entry (as long as it is not part of THAT tables Primary Key) or an entry that matches the Primary Keys value.
Purpose	Makes it possible for an attribute NOT to have a corresponding value but it will make it impossible to have an invalid entry
Example	A customer might not yet have an assigned sales rep number but it will be impossible to have an invalid sales rep number

Relational Algebra—A set of mathematical principles that form the basis of the manipulation of relational table contents; comprises eight main functions: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE. **Produced through SQL.**

1. **UNION** – must be Union-Compatible meaning all the columns (fields) are the same in each table. This merges all rows into one output.

2. **INTERSECT** – yields only the rows that appears in both tables. Must be Union-Compatible

F_Name	**INTERSECT**	**F_NAME**	**Yields**	**F_NAME**
George		Jane		Jane
Jane		William		Jorge
Elaine		Jorge		
Wilfred		Dennis		
Jorge				

3. **DIFFERENCE** – Does the opposite from above and give the records NOT found in the other table.

4. **PRODUCT** – Yields all possible pairs that the both could produce. So 10 records in one table and 10 records in another table would produce 100 records.

5. **SELECT** – yields values for all rows in a table

6. **PROJECT** - yields values for selected rows chosen

7. **JOIN** – Allows us to combine information from 2 or more tables

    - **Natural Join** links tables by selecting only the rows with common values in their attributes.
        - EquiJoin means the Where condition must "=" a value
        - ThetaJoin means the Where condition must be ">" or "<" a value

    - **Left Outer Join** yields all the rows in the LEFT table including all those that do not have a matching value in the RIGHT table
    - **Right Outer Join** yields all the rows in the RIGHT table including all those that do not have a matching value in the LEFT table
    - **INNER JOIN** returns all rows from both tables where there is a match. If there are rows in Employees that do not have matches in Orders, those rows will not be listed.

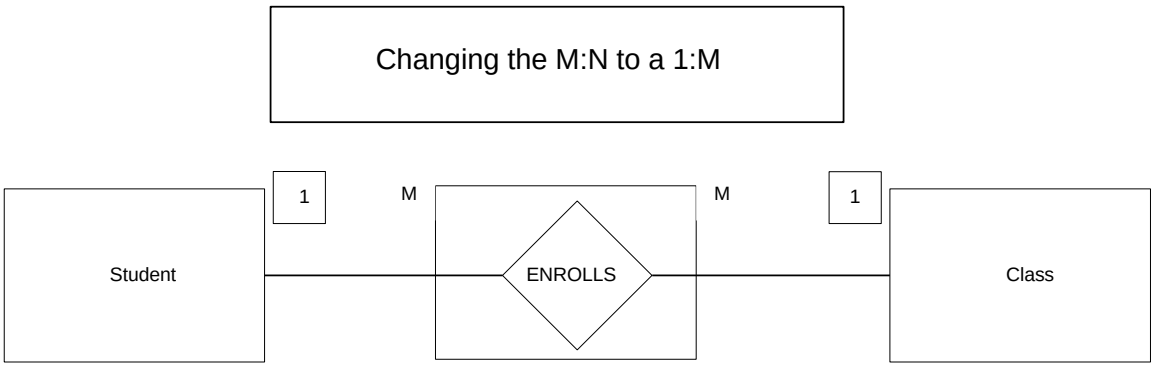
8. **DIVIDE** – Goes something like this

CODE and LOC DIVIDED by CODE = LOC and then it is the only value that the CODE s had in common.

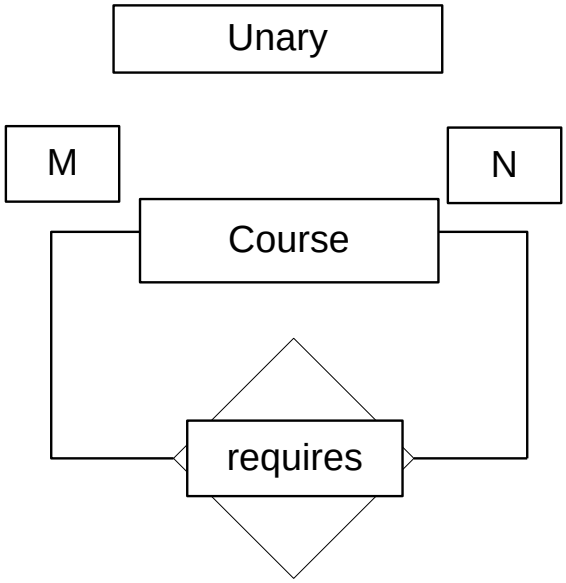
System Catalogs contains **metadata**

Homonyms—Indicates the use of the same name to label different attributes; should generally be avoided. Some relational software automatically checks for homonyms and either alerts the user to their existence or automatically makes the appropriate adjustments. See also *synonym*

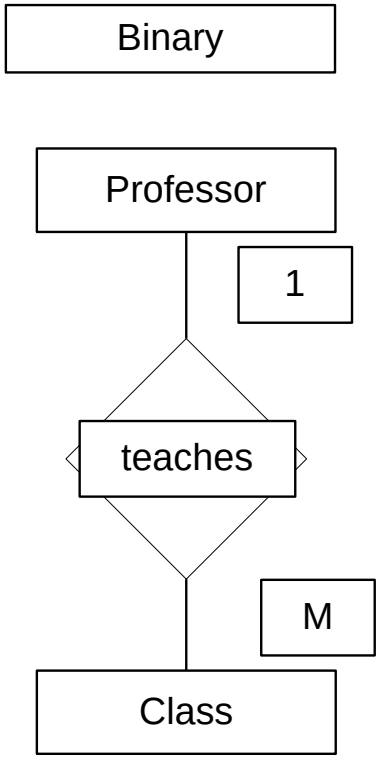
Synonym—The use of different names to identify the same object, such as an entity, an attribute, or a relationship; should (generally) be avoided. See also *homonym*.



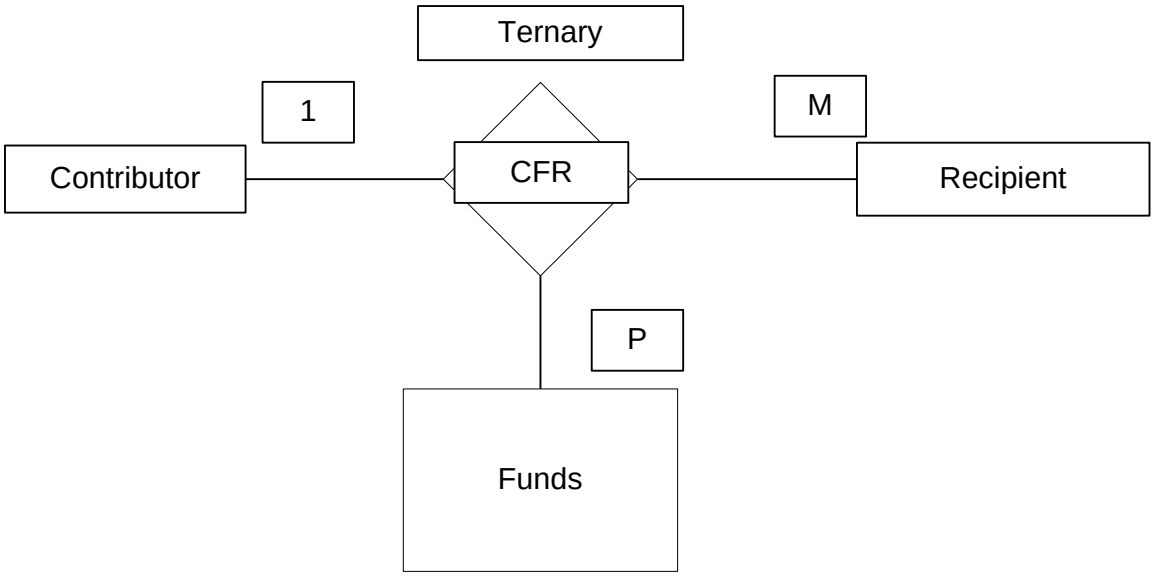
Unary Relationship exists when an association is maintained with a single entity.



A **Binary Relationship** exists when an association is maintained between two entities.



A **Ternary Relationship** exists when an association is maintained between three entities.

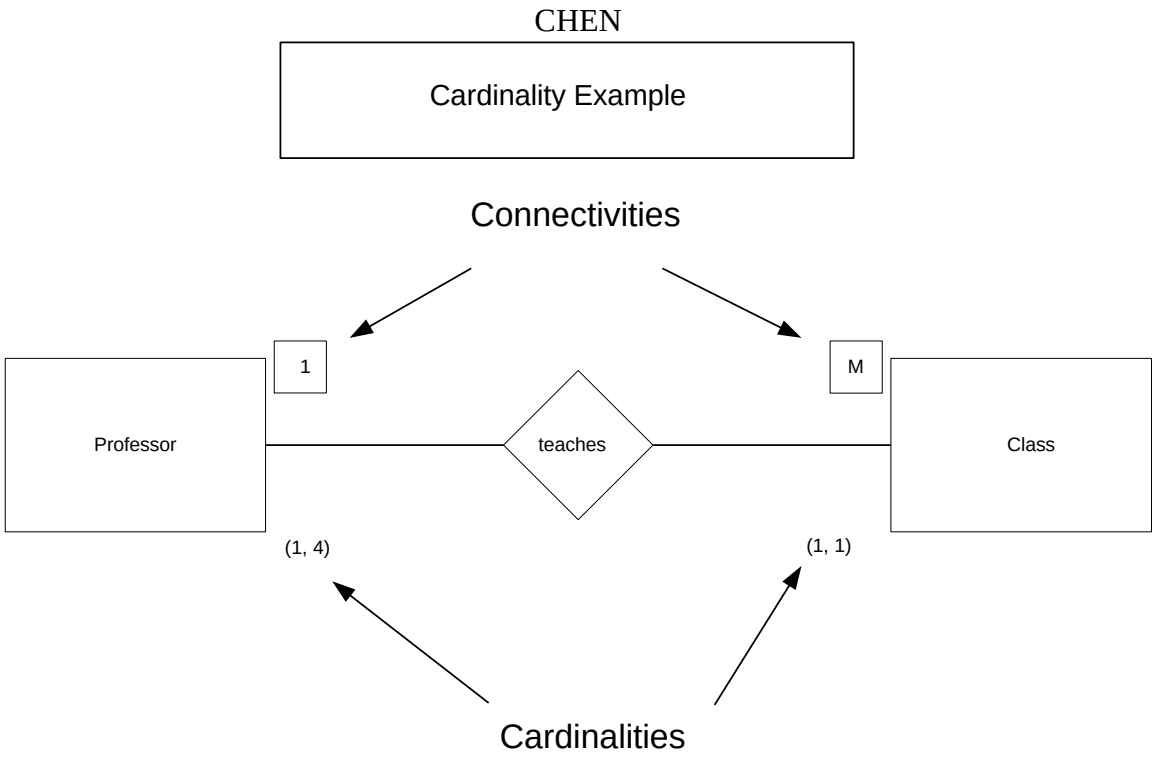


An **Index** is an orderly arrangement used to logically access rows in a table.

The **Index Key** is in effect the index's reference point.

A **Unique Index** is an index in which the index key can have only one pointer value (row) associated with it.

Cardinalities express the specific number of entity occurrences associated with one occurrence of the related entity.



Example: The Cardinality (1,4) written next to the professor indicates that the professor tables foreign key value occurs at least once and no more than four times in the class table. If the Cardinality had been written (1,N) then there would be no upper limit to the amount of classes that a professor might teach.

Cardinality
Crows Foot Model