

DBMS

MODULE 1 NOTES

INTRODUCTION

A database is a collection of related data

By data, we mean known facts that can be recorded and that have implicit meaning.

- **Data:**
- Data refers to raw facts, observations, or symbols that have no inherent meaning or context.
- It can be in various forms, such as numbers, text, images, audio, or video.
- Data is typically unprocessed and lacks interpretation or organization.
- Data can be collected, recorded, and stored, but it requires further processing to become meaningful.
- **Information:**
- Information is the result of processing and organizing data to provide meaning, context, and value.
- It represents processed and structured data that conveys meaning and significance to the recipient.
- Information is relevant, useful, and provides insights or knowledge to support decision-making or understanding.
- It is typically presented in a human-readable format, such as reports, documents, charts, or visualizations.

A database has the following implicit properties:

- A database represents some aspect of the real world, sometimes called the **miniworld or the universe of discourse (UoD)**. **Changes to the miniworld** are reflected in the database.
- A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
- A database is designed, built, and populated with data for a specific purpose. It has an intended group of users

- a database has some source from which data is derived, and an audience that is actively interested in its contents.
- In order for a database to be accurate and reliable at all times, changes must be reflected in the database as soon as possible.
- A database can be of any size and complexity.
- Example social media company such as Facebook, which has more than a billion users.

Database Management System

- A database management system (DBMS) is a computerized system that enables users to create and maintain a database.
- The DBMS is a *general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.*
- ***Defining a database***
 - involves specifying the data types, structures, and constraints of the data to be stored in the database.
- ***Constructing the database***
 - is the process of storing the data on some storage medium that is controlled by the DBMS.
- ***Manipulating a database***
 - includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
- ***sharing a database***
 - allows multiple users and programs to access the database simultaneously.

DATABASE ENVIRONMENT

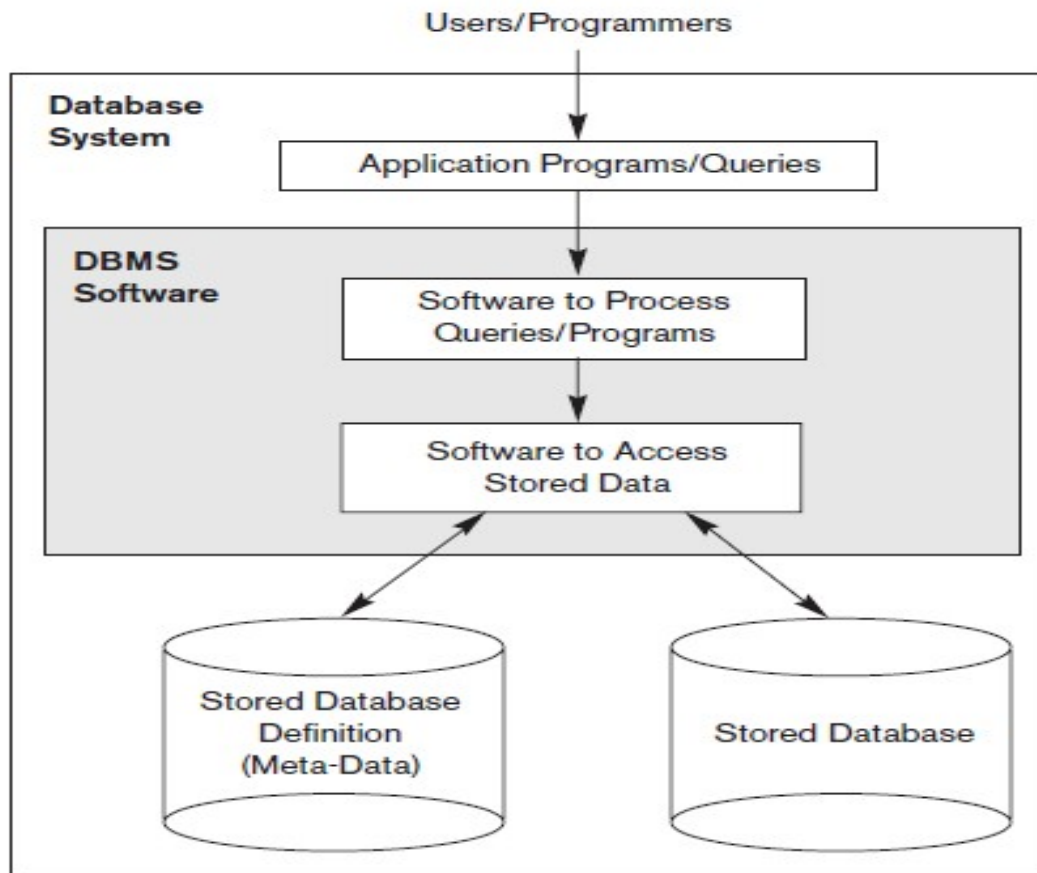


Figure 1.1
A simplified database
system environment.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Characteristics of the Database Approach

1. Self-describing nature of a database system
2. Insulation between programs and data, and data abstraction
3. Support of multiple views of the data
4. Sharing of data and multiuser transaction processing

1. Self-describing nature of a database system

- database system contains not only the database itself but also a complete definition or description of the database structure and constraints.
- This definition is stored in the DBMS catalog,
- which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.
- The information stored in the catalog is called meta-data

RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....
....
....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Figure 1.3

An example of a database catalog the database in Figure 1.2.

2. Insulation between Programs and Data, and Data Abstraction

- In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require *changing all programs that access that file*.
- *By contrast, DBMS access programs do not require* such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property **program-data independence**.

- object-oriented and object-relational systems users can define operations on data as part of the database definitions.
- An **operation (also called a *function or method*) is specified in two** parts. The *interface (or signature) of an operation includes the operation name and* the data types of its arguments (or parameters). The *implementation (or method) of the operation* is specified separately and can be changed without affecting the interface.

- User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed **program-operation independence**.
- The characteristic that allows program-data independence and program-operation independence is called **data abstraction**.

3. Support of Multiple Views of the Data

- A database typically has many types of users, each of whom may require a different perspective or **view of the database**.
- A view may be a subset of the database or it may contain **virtual data that is derived from the database files but is not explicitly stored**
- A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views.

4. Sharing of Data and Multiuser Transaction Processing

- A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time.
- The DBMS must include **concurrency control software to ensure that several users trying to update the same data** do so in a controlled manner so that the result of the updates is correct.
- Example: Airline reservation system. These types of applications are generally called **online transaction processing (OLTP) applications**.
- **A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.**

Advantages of Using the DBMS Approach

1. Controlling Redundancy
2. Restricting Unauthorized Access
3. Providing Persistent Storage for Program Objects
4. Providing Storage Structures and Search Techniques for Efficient Query Processing
5. Providing Backup and Recovery
6. Providing Multiple User Interfaces
7. Enforcing Integrity Constraints
8. Permitting Inferencing and Actions Using Rules and Triggers

Controlling Redundancy

- This redundancy in storing the same data multiple times

■ Employee Database

Ename	EID	Ephone no.	Address	Dept	Project name
Raju	201	9022009987	Bangalore	Sales	ABC
Ram	202	9900224455	Mumbai	Accounts	Xyz
Bhim	203	8855334466	Bangalore	Claims	MNO
Ravi	204	7700998866	Pune	Sales	PQR

If Ravi quits sales dept and joins claims dept, then to update this change

Ename	EID	Ephone no.	Address	Dept	Project name
Raju	201	9022009987	Bangalore	Sales	ABC
Ram	202	9900224455	Mumbai	Accounts	Xyz
Bhim	203	8855334466	Bangalore	Claims	MNO
Ravi	204	7700998866	Pune	Sales	PQR
Ravi	204	7700998866	Pune	claims	YYY

Ename	EID	Phoneno.	Address
Raju	201	9022009987	Bangalore
Ram	202	9900224455	Mumbai
Bhim	203	8855334466	Bangalore
Ravi	204	7700998866	Pune

EID	Dept Name	Project Name
201	Sales	ABC
202	Accounts	Xyz
203	Claims	MNO
204	Sales	PQR
204	Claims	YYY

REDUNDANCY

DRAWBACKS/PROBLEMS

- First, there is the need to perform a single logical update multiple times: once for each file where student data is recorded.
- This leads to *duplication of effort*. Second, *storage space is wasted when the same*
- data is stored repeatedly, and this problem may be serious for large databases.
- Third, files that represent the same data may become *inconsistent*

Restricting Unauthorized Access

- When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database.
- some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update.
- Hence, the type of access operation—retrieval or update—must also be controlled.
- Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database.
- DBMS provide a **security and authorization subsystem, which the DBA uses to create** accounts and to specify account restrictions.

Providing Persistent Storage for Program Objects

- Object-oriented database systems are compatible with programming languages such as C++ and Java, and the DBMS software automatically performs any necessary conversions.
- Hence, a complex object in C++ can be stored permanently in an object-oriented DBMS. Such an object is said to be **persistent, since it survives the termination of program execution and can** later be directly retrieved by another program.

Providing Storage Structures and Search Techniques for Efficient Query Processing

- Database systems must provide capabilities for *efficiently executing queries and updates. Because the database is typically stored on disk,*
- *the DBMS must provide* specialized data structures and search techniques to speed up disk search for the desired records. Auxiliary files called **indexes are often used for this purpose.**
- Indexes are typically based on tree data structures or hash data structures that are suitably modified for disk search.

Providing Backup and Recovery

- A DBMS must provide facilities for recovering from hardware or software failures.
- The **backup and recovery subsystem of the DBMS is responsible for recovery.**
- **For** example, if the computer system fails in the middle of a complex update transaction,
- the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing.
- *Disk backup* is also necessary in case of a catastrophic disk failure.

Providing Multiple User Interfaces

- Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces.
- These include
 - Apps for mobile users,
 - Query languages for casual users, programming language interfaces for application programmers.
 - Menu-driven interfaces and natural language interfaces for standalone users.

History of database applications.

- The history of database applications is closely tied to the development of computer technology and the need for efficient data storage and retrieval. Here is a brief overview of the major milestones in the history of database applications:
- Early Database Systems (1960s-1970s):
 - During the 1960s, the first database management systems (DBMS) emerged. These systems were primarily used for scientific and military applications.
 - IBM's Information Management System (IMS), developed in the mid-1960s, was one of the first widely used database management systems. It employed a hierarchical data model.
- Relational Databases (1970s-1980s):
 - In 1970, Edgar Codd introduced the relational model, which formed the basis for the modern relational database management systems (RDBMS).
 - IBM's System R and the University of California, Berkeley's Ingres were among the early implementations of relational database systems.
 - In 1979, Oracle Corporation released Oracle Database, one of the most influential commercial relational database systems. It played a significant role in popularizing the use of relational databases.
- Client-Server Architecture (1980s-1990s):
 - The development of local area networks (LANs) and the client-server architecture in the 1980s led to significant advancements in database applications.
 - RDBMSs like Oracle, IBM's DB2, and Microsoft SQL Server became more widely adopted, allowing multiple users to access and manipulate data simultaneously.
 - Structured Query Language (SQL) became the standard language for interacting with relational databases.

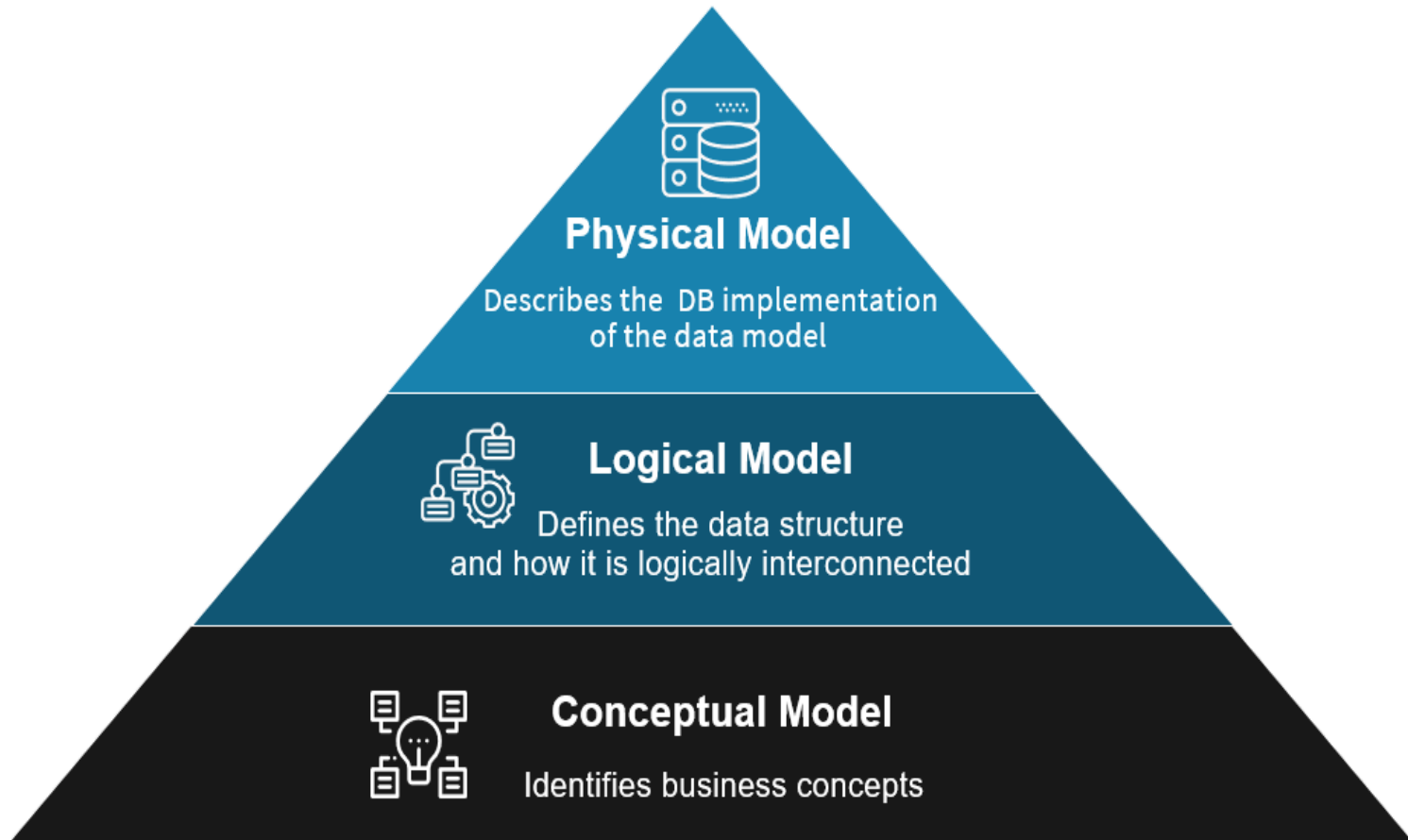
- **Object-Oriented Databases (1990s-2000s):**
 - Object-oriented databases (OODBMS) emerged as an alternative to relational databases, aiming to bridge the gap between object-oriented programming and data storage.
 - OODBMSs stored data in the form of objects, allowing for more natural representation of complex data structures.
 - However, the adoption of OODBMSs was limited due to the dominance of the relational model and the complexity of integrating object-oriented concepts with existing systems.
- **Web-Based Applications (2000s-2010s):**
 - The rise of the internet and the World Wide Web brought about significant changes in the way databases were used.
 - Web-based applications required databases to support high levels of concurrency and scalability to handle large user bases.
 - MySQL and PostgreSQL became popular open-source relational databases for web applications, while NoSQL databases like MongoDB and Cassandra emerged to address specific needs like scalability and flexibility.
- **Big Data and Cloud Computing (2010s-present):**
 - The explosion of data generated by various sources gave rise to the era of big data.
 - Distributed computing frameworks like Apache Hadoop and Apache Spark were developed to process and analyze vast amounts of data stored across multiple machines.
 - Cloud computing platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform offered scalable and managed database services, making it easier to deploy and manage databases in the cloud.
- **Modern Database Technologies:**
 - In recent years, new database technologies have gained prominence. These include graph databases for handling highly connected data, time-series databases for managing time-stamped data, and blockchain databases for decentralized and secure transactions.
 - Additionally, the advent of machine learning and artificial intelligence has led to the integration of these technologies with databases, enabling advanced analytics and data-driven decision-making.

OVERVIEW OF DATABASE LANGUAGES AND ARCHITECTURES

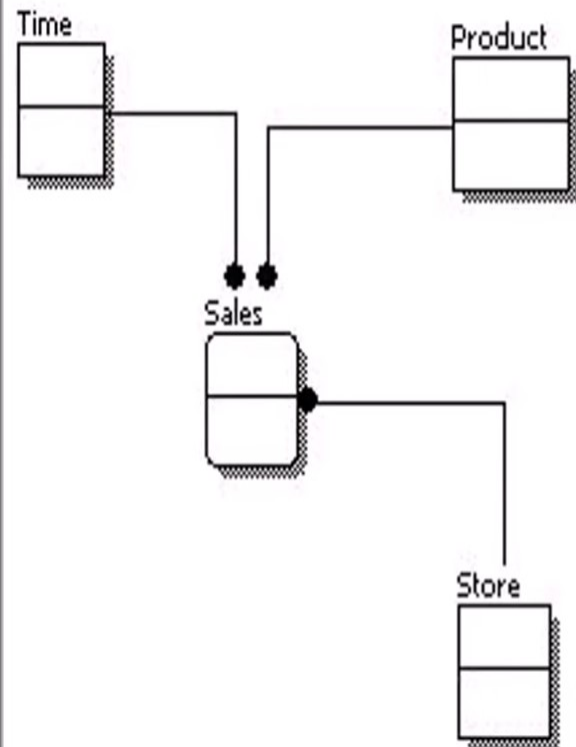
Data Models, Schemas, and Instances.

- A **data model**—a collection of concepts that can be used to describe the structure of a database
- By *structure of a database* we mean the data types, relationships, and constraints that apply to the data.
- Data modelling is a pictorial representation of a tables and their relationships.
- Data Modeling provides the necessary means to achieve abstraction.

Categories of Data Models

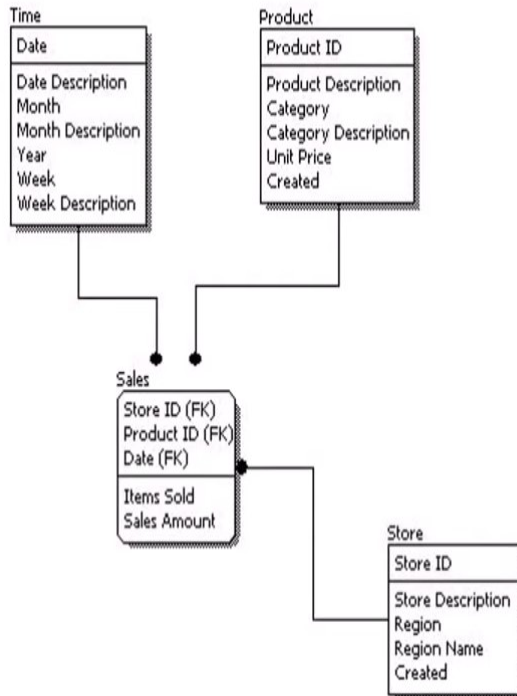


Conceptual Data Model



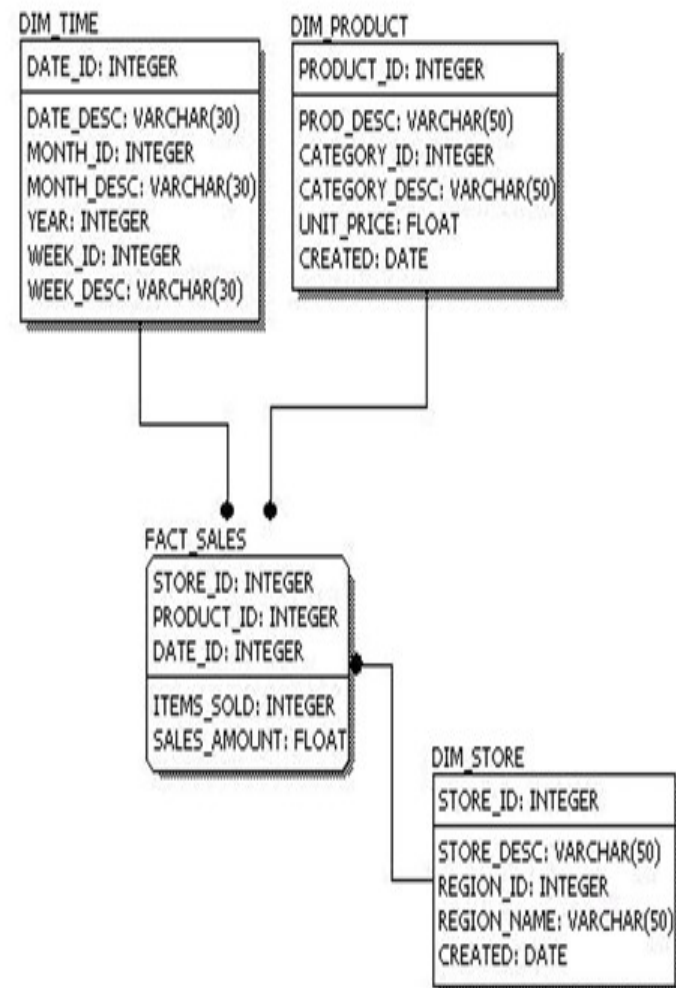
- Highly abstract
- Easily understood
- Easily enhanced
- Only “Entities” visible
- Abstract Relationships
- No software tool is required to define a Conceptual Data Model

Logical Data Model



- Presence of Attributes for each Entity
- Key Attributes
- Non-Key Attributes
- Primary Key – Foreign Key Relationships
- User Friendly Attribute names
- More detailed than Conceptual Model
- Database agnostic
- Bit more effort required to enhance, in comparison to Conceptual Model
- Data Modeling tools like ERWin or PowerDesigner can be used to create Logical Data Models. This can be automatically converted to a Physical Data Model with the help of these tools.

Physical Data Model



Entities referred to as Tables

Attributes referred to as Columns

Database compatible table names

Database compatible column names

Database specific data types

Difficult for users to understand

Significantly more effort required to enhance in comparison to Logical Model

Will include indexes, constraints, triggers & other DB objects

Difficult to port to a different database, once design is finalized.

Tools like ERWin and PowerDesigner can help in automatically porting over the Logical Data Model to Physical Data Models of different versions.

Data Modelling Tools

- There are several tools available for creating data models, depending on your specific requirements and preferences. Here are some popular tools commonly used for data modeling:
- **ER/Studio:** ER/Studio is a comprehensive data modeling tool that allows you to create and manage both logical and physical data models. It provides a user-friendly interface with features like entity-relationship diagrams, data dictionary, and data lineage tracking.
- **Lucidchart:** Lucidchart is a cloud-based diagramming tool that supports various types of diagrams, including data models. It offers an intuitive drag-and-drop interface, collaboration capabilities, and the ability to export your models in different formats.
- **Microsoft Visio:** Microsoft Visio is a widely used diagramming tool that includes features for creating data models. It provides a rich set of predefined shapes and templates, making it easy to create and modify data models. Visio also integrates well with other Microsoft tools like Excel and SQL Server.
- **PowerDesigner:** PowerDesigner by SAP is a robust data modeling and metadata management tool. It supports various modeling techniques, such as entity-relationship diagrams, UML diagrams, and business process models. PowerDesigner offers advanced features like impact analysis and integration with other SAP products.
- **Oracle SQL Developer Data Modeler:** This tool is specifically designed for data modeling in Oracle databases. It provides a graphical interface for creating and managing data models, along with features like version control, collaboration, and the ability to generate DDL scripts.
- **IBM InfoSphere Data Architect:** InfoSphere Data Architect is a powerful data modeling tool offered by IBM. It supports a wide range of database platforms and allows you to create logical, physical, and dimensional data models. It also integrates with other IBM data management solutions.
- **Toad Data Modeler:** Toad Data Modeler is a popular tool for database design and data modeling. It supports various database platforms and offers features like reverse engineering, forward engineering, and synchronization of models with databases.

database schema

- The description of a database is called the **database schema**, which is specified during

Figure 2.1

Schema diagram for
the database in
Figure 1.2.

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

- The actual data in a database may change quite frequently. For example, the student database changes every time we add a new student or enter a new grade.
- The data in the database at a particular moment in time is called a **database state** or **snapshot**. It is also called the *current* set of **occurrences** or **instances** in the database.
- Schema is sometimes called the **intension**, and a database state is called an **extension** of the schema.

Three-Schema Architecture and Data Independence

- **The Three-Schema Architecture:**
- The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:
 - **internal schema,**
 - **conceptual schema**
 - **external schemas**

Three-Schema Architecture

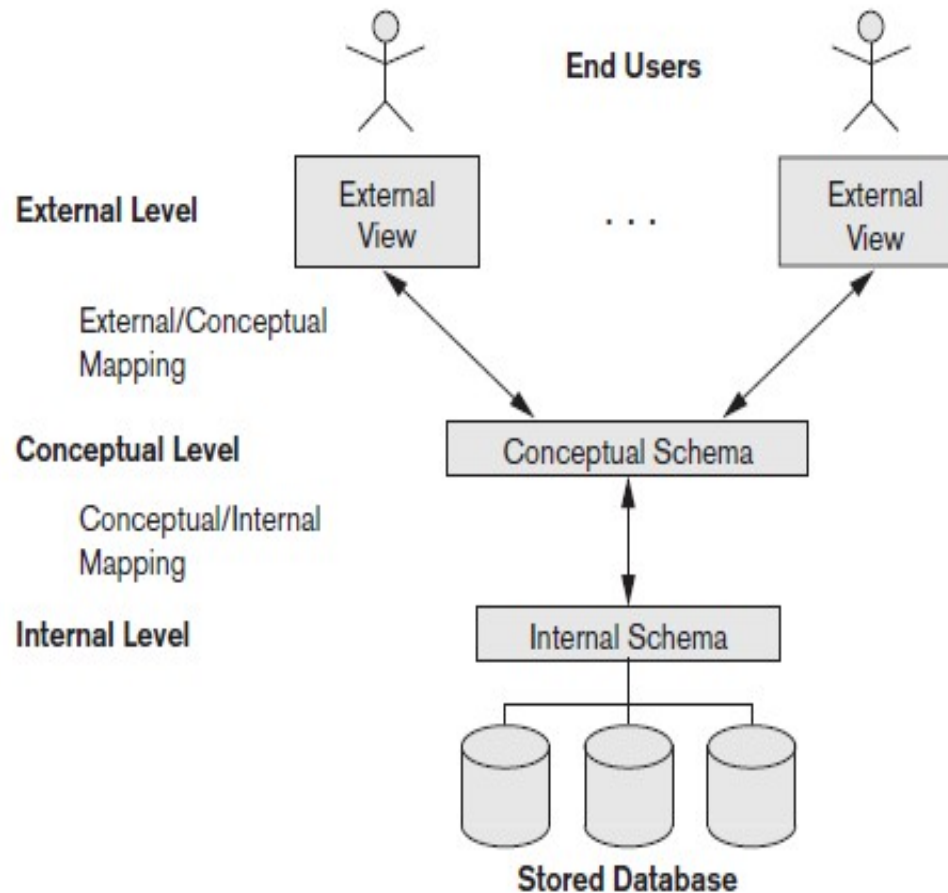
- **Internal schema,**
- describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

- **Conceptual schema**
- The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

- **External schemas**
- Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Three-Schema Architecture

Figure 2.2
The three-schema architecture.



Three-Schema Architecture

- In the three-schema architecture, each user group refers to its own external schema. Hence,
- The DBMS must transform a request specified on an external schema into a request against the conceptual schema,
- and then into a request on the internal schema for processing over the stored database.
- The processes of transforming requests and results between levels are called **mappings**.

Data Independence

- The three-schema architecture can be used to further explain the concept of **data independence**
- DATA INDEPENDENCE: can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
-

DATA INDEPENDENCE

- We can define two types of data independence:
- **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs.
- We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). In the last case, external schemas that refer only to the remaining data should not be affected.

- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema.
- Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update.

Database Languages and Interfaces

DBMS Languages

- Data definition language (DDL):
 - storage definition language (SDL)
 - View definition language (VDL)
 - data manipulation language (DML)
 - record-at-a-time
 - set-at-a-time
- Structured Query Language (SQL)

Database languages

- **Data definition language (DDL)**
- In many DBMSs where no strict separation of levels is maintained, one language, called the **data definition language (DDL)**, is used by the DBA and by database designers to define both schemas.
- The DBMS will have a DDL compiler whose function is to process DDL statements in order to identify descriptions of the schema constructs and to store the schema description in the DBMS catalog.

Database languages

- **Storage definition language (SDL),**
- In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. Another language, the **storage definition language (SDL), is used to specify the internal schema.**

Database languages

- **View definition language (VDL)**
- For a true three-schema architecture, we would need a third language, the **view definition language (VDL)**, to specify user **views and their mappings to the conceptual schema**,

Database languages

- **data manipulation language(DML)**
- Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database. Typical manipulations include retrieval, insertion, deletion, and modification of the data. The DBMS provides a set of operations or a language called the **data manipulation language (DML) for these purposes.**

Database languages

- **Structured Query Language (SQL)**
- In current DBMSs, the preceding types of languages are usually *not considered distinct languages; rather, a comprehensive integrated language is used that includes* constructs for conceptual schema definition, view definition, and data manipulation all comes under one language called SQL
- SQL represents a combination of DDL, VDL, and DML, as well as statements for constraint specification, schema evolution, and many other features.
- Storage definition is typically kept separate, since it is used for defining physical storage structures to fine-tune the performance of the database system, which is usually done by the DBA staff.

Database languages

- A low level or procedural DML *must be embedded in a general-purpose programming* language. This type of DML typically retrieves individual records or objects from the database and processes each separately. Low-level DMLs are also called **record-at-a-time DMLs**
- High-level DMLs, such as SQL, can specify and retrieve many records in a single DML statement; therefore, they are called **set-at-a-time or set-oriented DMLs**.

DBMS Interfaces

- **Menu-based Interfaces for Web Clients or Browsing.** These interfaces present the user with lists of options (called menus) that lead the user through the formulation of a request.
- Menus do away with the need to memorize the specific commands and syntax of a query language; rather, the query is composed step-by-step by picking options from a menu that is displayed by the system.
- Pull-down menus are a very popular technique in Web-based user interfaces.

■ Apps for Mobile Devices.

■ These interfaces present mobile users with access to their data. For example, banking, reservations, and insurance companies, among many others, provide apps that allow users to access their data through a mobile phone or mobile device. The apps have built-in programmed interfaces that typically allow users to login using their account name and password;

- **Forms-based Interfaces.** A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.
- **Graphical User Interfaces.** A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms

- **Natural Language Interfaces.** These interfaces accept requests written in English or some other language and attempt to *understand them*. A *natural language* interface usually has its own *schema*, which is similar to the *database conceptual* schema, as well as a dictionary of important words. The natural language interface refers to the words in its schema, as well as to the set of standard words in its dictionary, that are used to interpret the request.
- **Keyword-based Database Search.** These are somewhat similar to Web search engines, which accept strings of natural language (like English or Spanish) words and match them with documents at specific sites (for local search engines) or Web pages on the Web at large (for engines like Google or Ask).

- **Speech Input and Output.**
- Applications with limited vocabularies, such as inquiries for telephone directory, flight arrival/departure, and credit card account information, are allowing speech for input and output to enable customers to access this information. The speech input is detected using a library of predefined words
- **Interfaces for Parametric Users.** Parametric users, such as bank tellers, often have a small set of operations that they must perform repeatedly. For example, a teller is able to use single function keys to invoke routine and repetitive transactions such as account deposits or withdrawals, or balance inquiries

■ **Interfaces for the DBA.** Most database systems contain privileged commands that can be used only by the DBA staff. These include commands for

- Creating accounts,
- Setting system parameters,
- Granting account authorization,
- changing a schema,
- Reorganizing the storage structures of a database.

The Database System Environment

2.4.1 DBMS Component Modules:

- Figure 2.3 illustrates, in a simplified form, the typical DBMS components. The figure is divided into two parts.
- **The top part:** of the figure refers to the various users of the database environment and their interfaces.
- **The lower part:** shows the internal modules of the DBMS responsible for storage of data and processing of transactions.

The Database System Environment

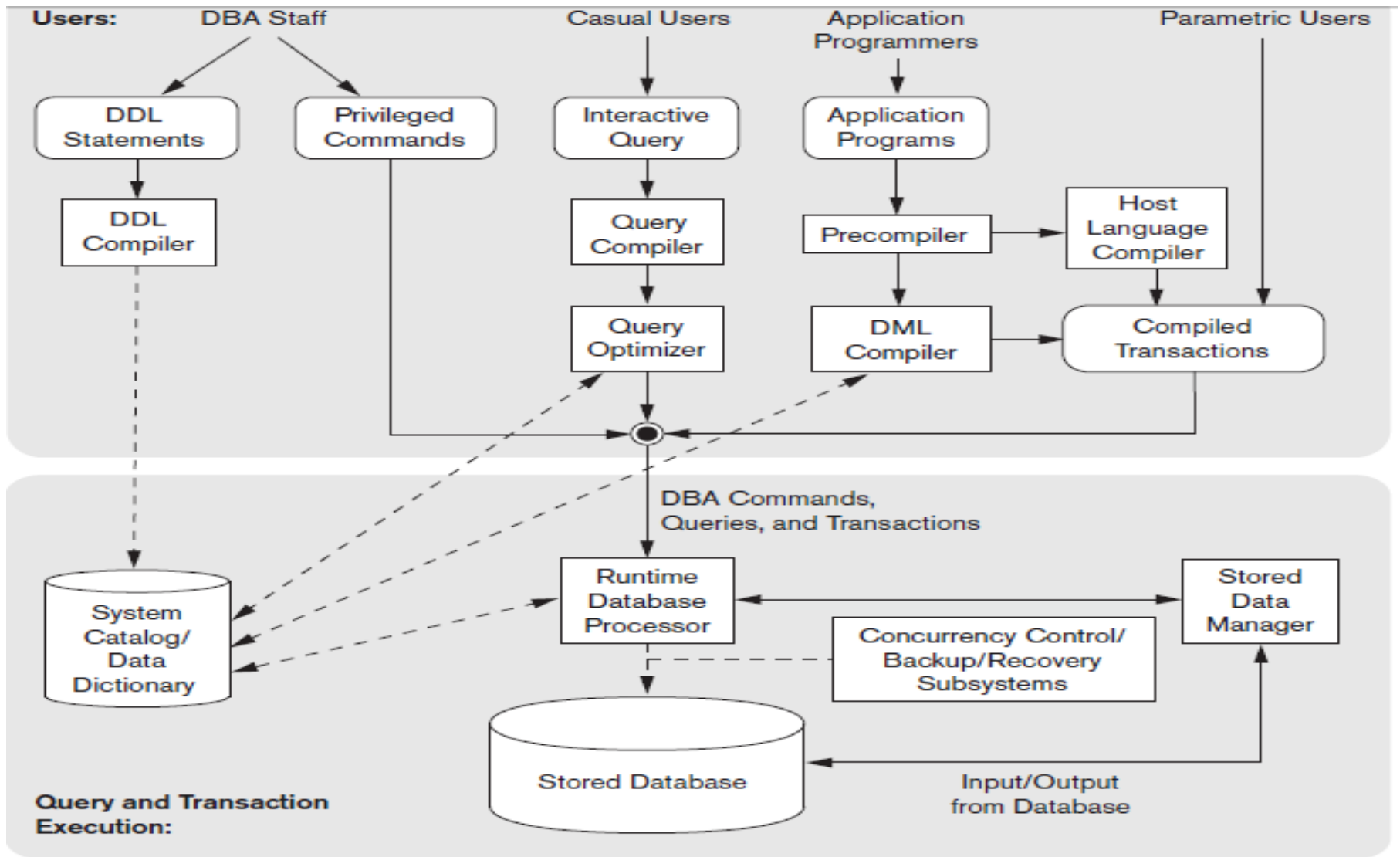


Figure 2.3

The Database System Environment

- The database and the DBMS catalog are usually stored on disk.
- Access to the disk is controlled primarily by the **operating system (OS), which schedules disk** read/write.
- Many DBMSs have their own **buffer management module to schedule** disk read/write, because management of buffer storage has a considerable effect on performance. Reducing disk read/write improves performance considerably.
- A higher-level **stored data manager module of the DBMS controls access** to DBMS information that is stored on disk

The Database System Environment

- **Top part:** It shows interfaces for the DBA staff, casual users who work with interactive interfaces to formulate queries, application programmers who create programs using some host programming languages, and parametric users who do data entry work by supplying parameters to predefined transactions.
- The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.
- The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file,

The Database System Environment

- Casual users and persons with occasional need for information from the database interact using the **interactive query interface in Figure**.
- These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a **query compiler that compiles** them into an internal form. This internal query is subjected to query optimization
- the **query optimizer is** concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of efficient search algorithms during execution.

The Database System Environment

- Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a precompiler. The **precompiler extracts DML commands** from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access. The rest of the program is sent to the host language compiler

- In the lower part of Figure 2.3, the runtime database processor executes
 - (1) the privileged commands,
 - (2) the executable query plans, and
 - (3) the canned transactions with runtime parameters.
- It works with the **system catalog and may update it** with statistics.
- It also works with the **stored data manager, which in turn uses basic** operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory.
- The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory.

The Database System Environment

■ 2.4.2 Database System Utilities

- **Loading.** A loading utility is used to load existing data files—such as text files or sequential files—into the database.
- **Backup.** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database in case of catastrophic disk failure.
- **Database storage reorganization.** This utility can be used to reorganize a set of database files into different file organizations and create new access paths to improve performance.
- **Performance monitoring.** Such a utility monitors database usage and provides statistics to the DBA.

Entity Types, Entity Sets, Attributes, and Keys

■ Entity

- An entity, which is a *thing or object in the real world with an independent existence*.
- An entity may be an object with a physical existence (for example, a particular person, car, house, or employee)
- or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

■ Attribute

- Each entity has **attributes**—the particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job. A particular entity will have a value for each of its attributes. The attribute values that describe each entity become a major part of the data stored in the database.

- **Composite versus Simple (Atomic) Attributes.** Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.
- For example, the Address attribute of the EMPLOYEE entity shown
- in Figure 3.3 can be subdivided into Street_address, City, State, and Zip,³ with the values '2311 Kirby', 'Houston', 'Texas', and '77001'. Attributes that are not divisible are called simple or atomic attributes.
- **Stored versus Derived Attributes.** In some cases, two (or more) attribute values are related—for example, the Age and Birth date attributes of a person.
- For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's Birth_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute.

■ **NULL Values.**

- In some cases, a particular entity may not have an applicable value
- for an attribute. For example, the Apartment_number attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes
- For such situations, a special value called NULL is created.

- **Complex Attributes.**
- composite and multivalued attributes can be nested arbitrarily. We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called **complex attributes**.
- **For example, if a person** can have more than one residence and each residence can have a single address and multiple phones, an attribute Address_phone for a person can be specified as
- {Address_phone({Phone(Area_code,Phone_number)},Address(Street_address
(Number,Street,Apartment_number),City,State,Zip))}

Entity Types and Entity Sets.

- A database usually contains groups of entities that are similar.
- For example, a company employing hundreds of employees may want to store similar information concerning each of the employees.
- These employee entities share the same attributes, but each entity has its own value(s) for each attribute. An entity type defines a collection (or set) of entities that have the same attributes.

Entity Type Name:

EMPLOYEE

COMPANY

Name, Age, Salary

Name, Headquarters, President

Entity Set:
(Extension)

e_1 •

(John Smith, 55, 80k)

e_2 •

(Fred Brown, 40, 30K)

e_3 •

(Judy Clark, 25, 20K)

•
•
•

c_1 •

(Sunco Oil, Houston, John Smith)

c_2 •

(Fast Computer, Dallas, Bob King)

•
•
•

Figure 3.6

Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

ROLES

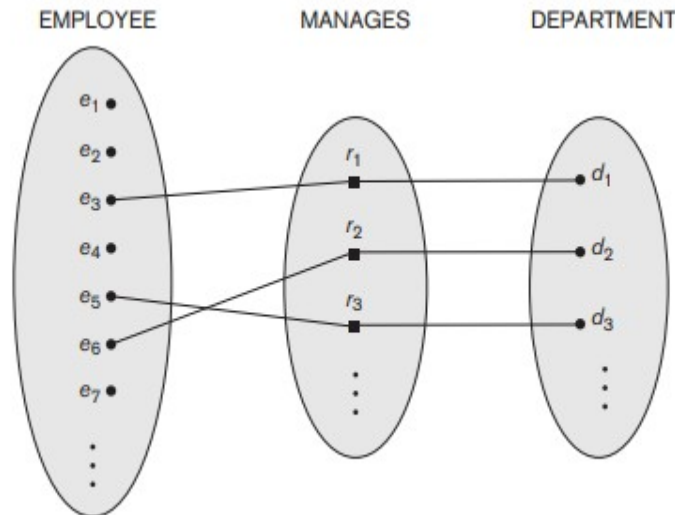
- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The role name signifies the role that a participating entity from the entity type plays in each relation-ship instance.
- For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.

The cardinality ratio

- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in. For example, in the WORKS_FOR binary relationship type, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to (that is, employs) any number of employees (N),⁹ but an employee can be related to (work for) at most one department.

- An example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department.
- —at any point in time—an employee can manage at most one department and a department can have at most one manager.

Figure 3.12
A 1:1 relationship,
MANAGES.



Participation constraint

- The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- There are two types of participation constraints
 - —total
 - partial—

Participation constraint-Total Participation

- If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance.
- Thus, the participation of EMPLOYEE in WORKS_FOR is called total participation, meaning that every entity in the total set of employee entities must be related to a department entity via WORKS_FOR.
- Total participation is also called existence dependency

Participation constraint-Partial Participation

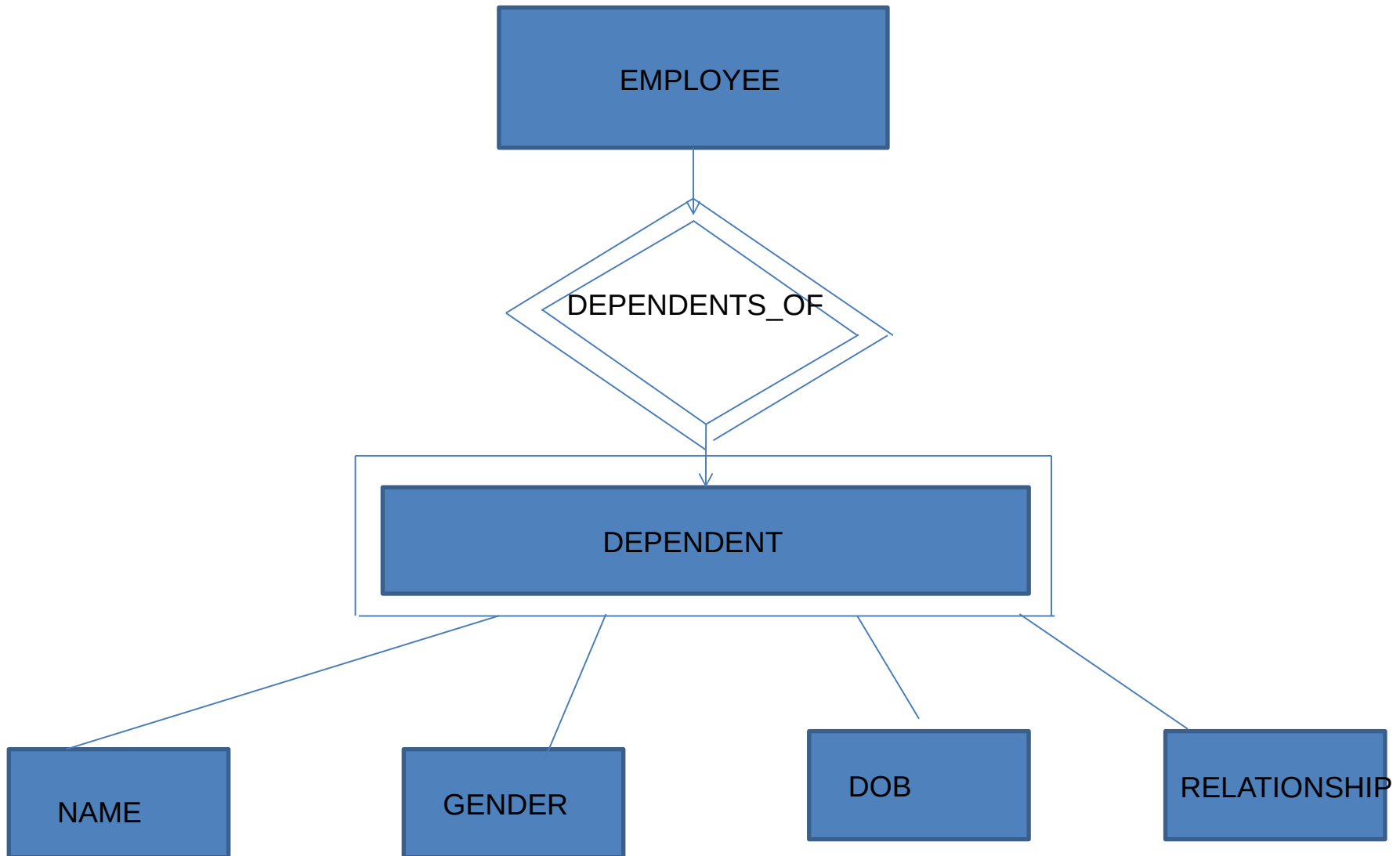
- we do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is partial, meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.

structural constraints

- We will refer to the cardinality ratio and participation constraints, taken together, as the structural constraints of a relationship type.

Weak Entity Types

- Entity types that do not have key attributes of their own are called weak entity types



weak entity

- Consider the entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee
- In our example, the attributes of DEPENDENT are Name (the first name of the dependent), Birth_date, Gender, and Relationship (to the employee). Two dependents of two distinct employees may, by chance, have the same values for Name, Birth_date, Gender, and Relationship, but they are still distinct entities. They are identified as distinct entities only after determining the particular employee entity to which each dependent is related.
- Each employee entity is said to own the dependent entities that are related to it.

weak entity

- Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.
- We call this other entity type the identifying or owner entity type, and we call the relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type. A weak entity type always has a total participation constraint (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.

ER Diagrams

3.7 ER Diagrams, Naming Conventions, and Design Issues


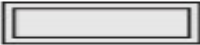









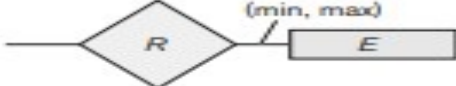
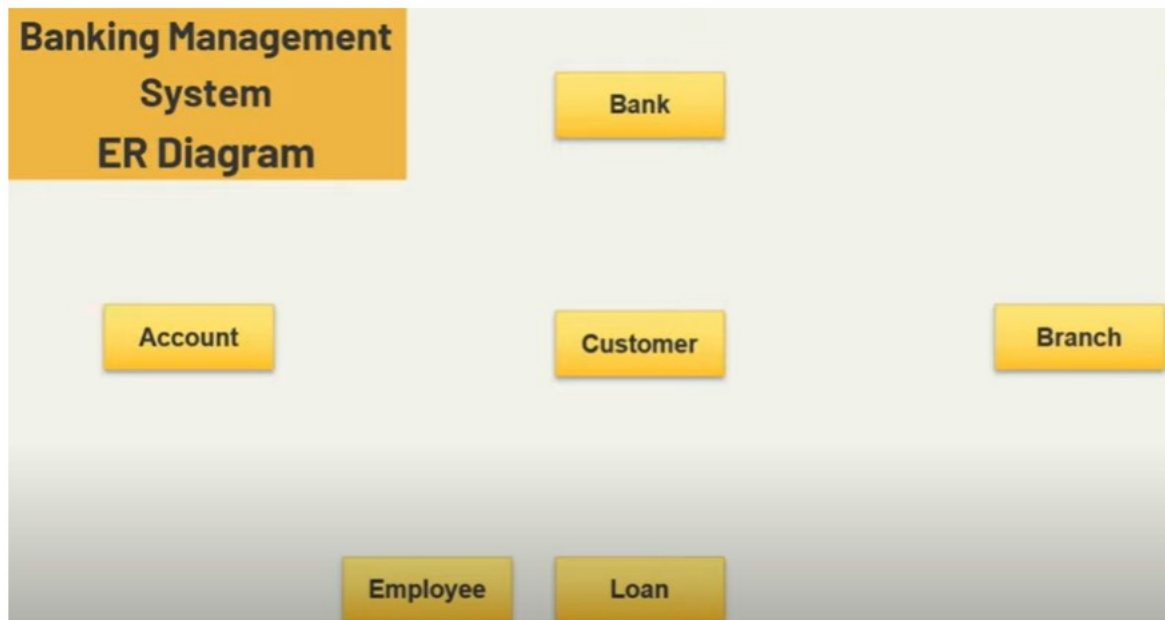
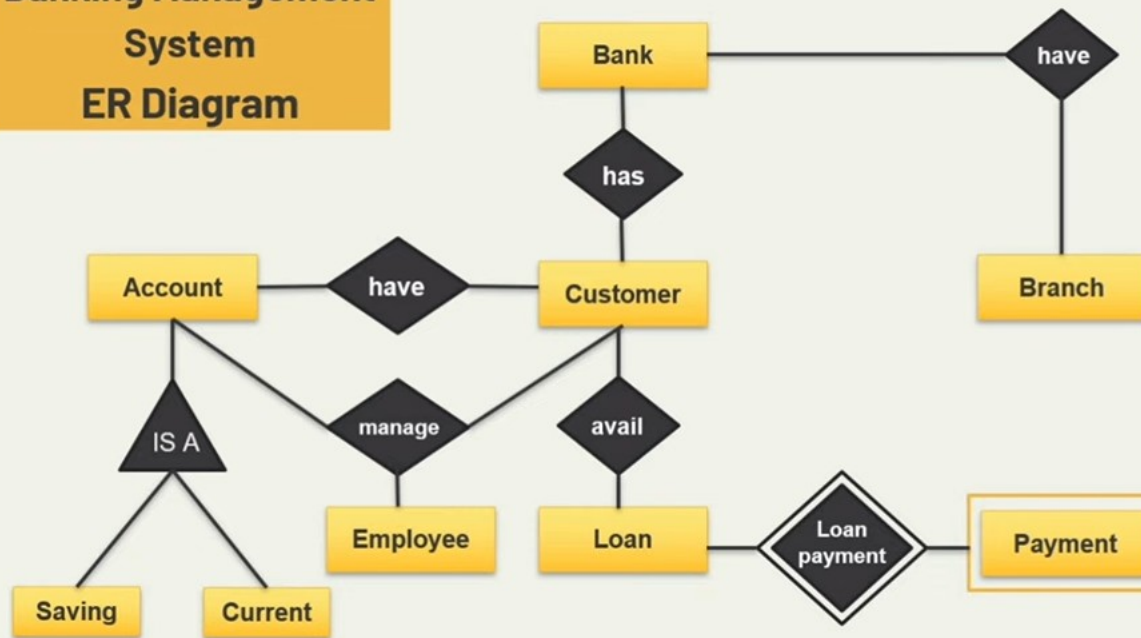
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

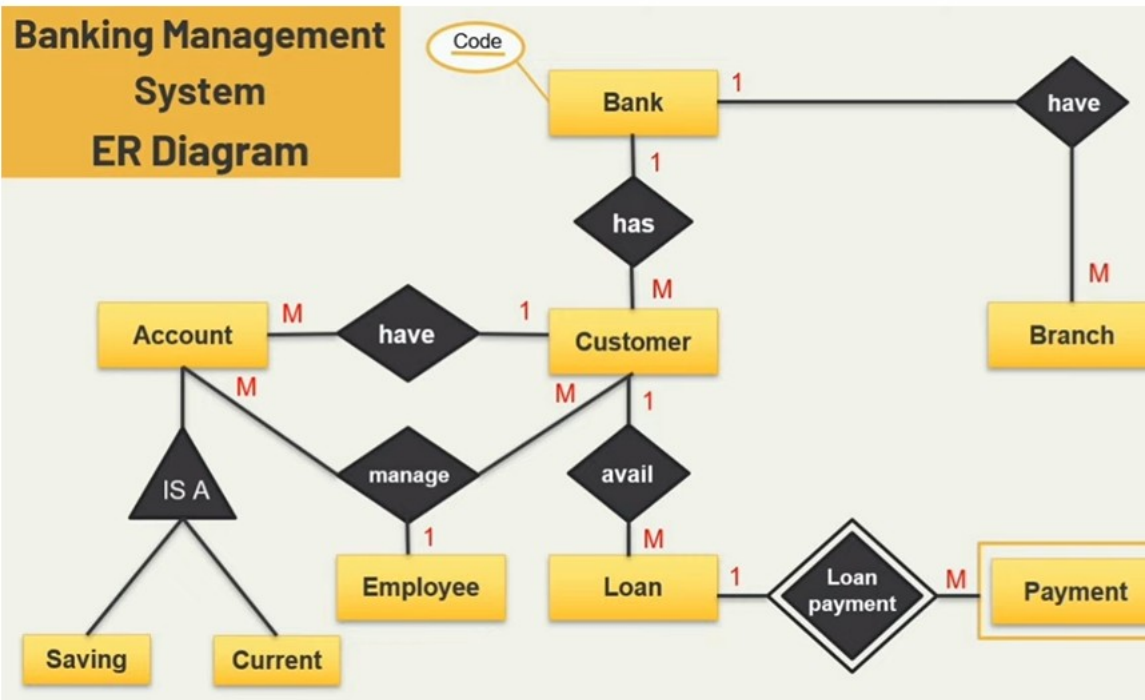
Figure 3.
Summary of
notation for
diagrams.



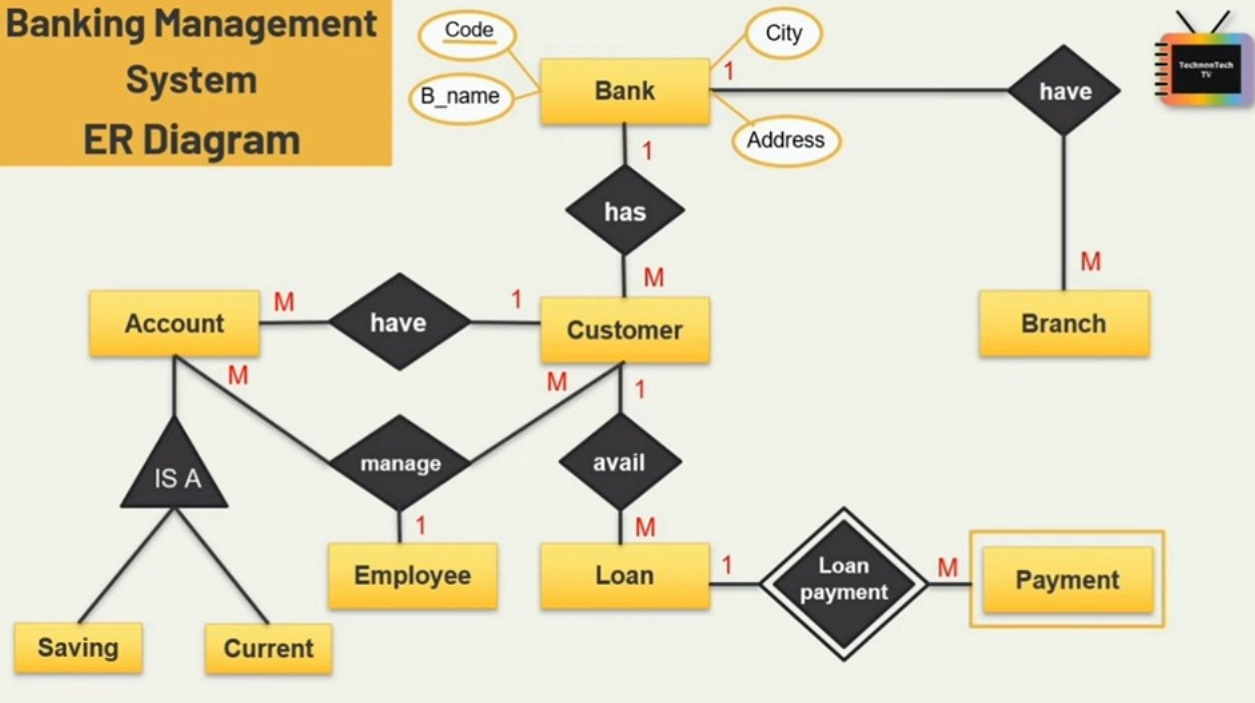
Banking Management System ER Diagram



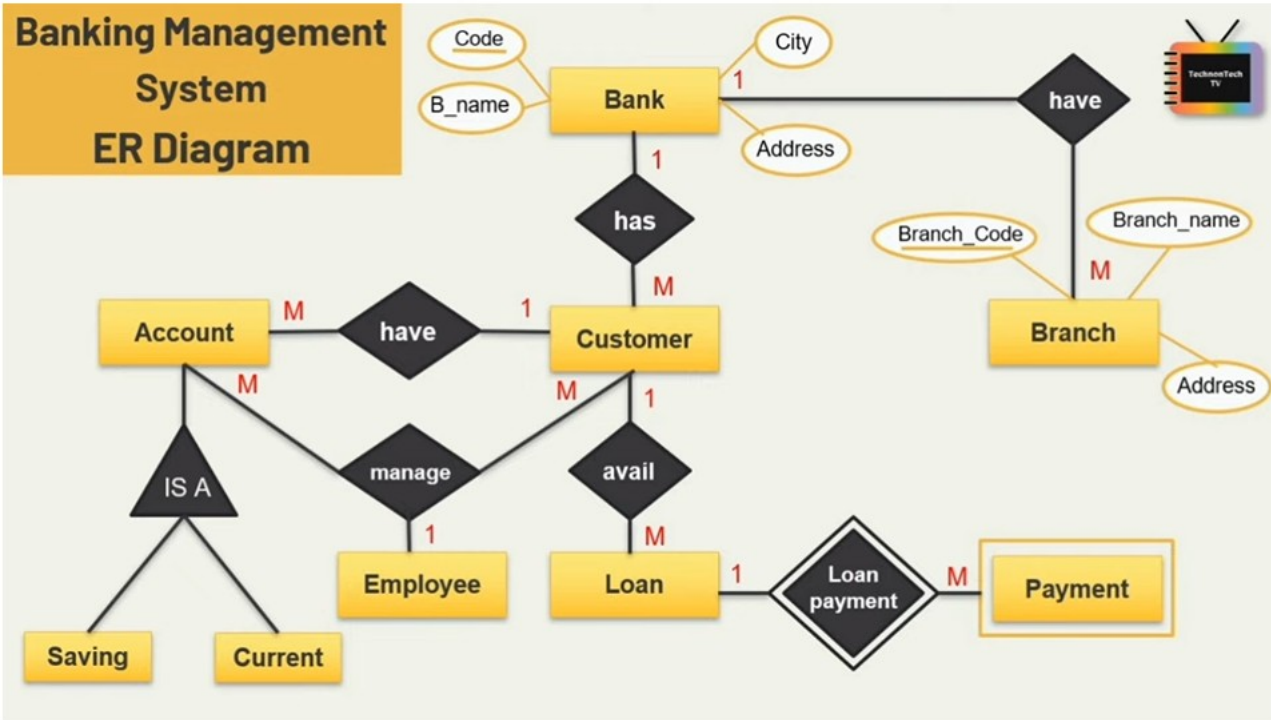
Banking Management System ER Diagram



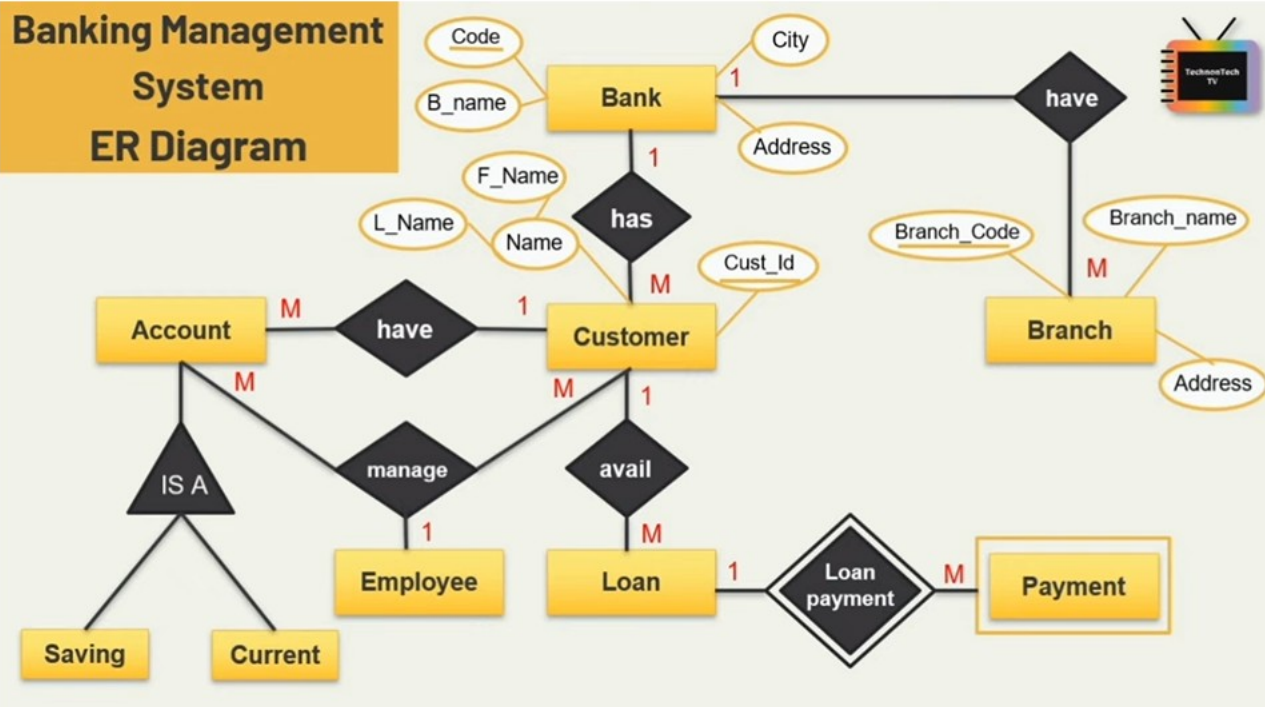
Banking Management System ER Diagram

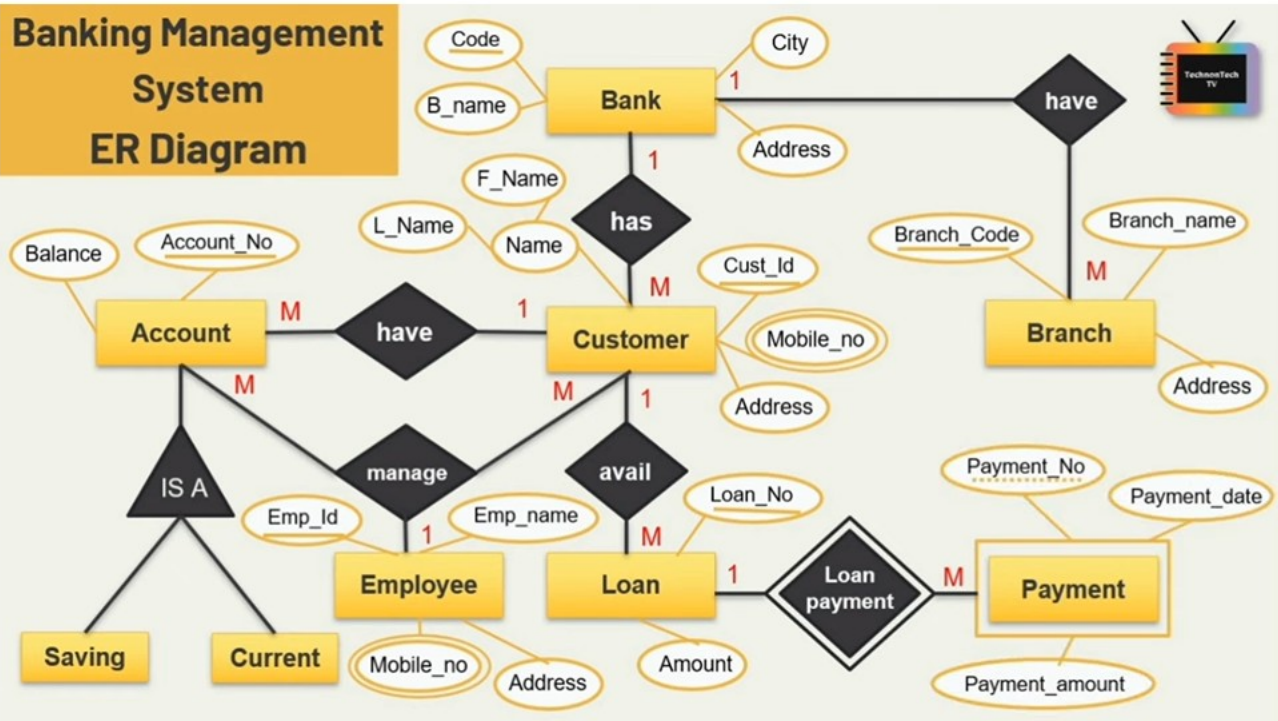


Banking Management System ER Diagram



Banking Management System ER Diagram





ER Diagram For Airline Reservation System

