DATABASE MANAGEMENT SYSTEM

SYLLABUS OF SEMESTER –II, M.C.A. (MASTER IN COMPUTER APPLICATION)

Course Code: MCT545 Course: Database Management Systems

L: 3 Hrs. P: 0 Hrs. Per week **Total Credits: 3**

Course Objectives

1.To describe how to design, manipulate and manage databases.

2. To Develop preliminary understandings and skills for designing a database information system.

3. To understand the concepts of SQL and PL/SQL, implement database systems in real world.

Course Outcomes

On successful completion of the course, students will be able to:

1. Recognize the context, phases and techniques for designing and building database information systems in business.

2. Design and implement a database schema, database objects for a given problem-domain, organize database entities, understand the principles of storage structures and apply various Normalization techniques.

3. Apply the concurrency control and recovery techniques to build application for real world problem and understand query processing techniques involved in query optimization.

Syllabus

UNIT -I

Introduction to Database Management Systems:

Introduction, Conventional File Processing System,

Components of DBMS, Advantages and Disadvantages, Three-level Architecture proposal for DBMS, Abstraction and Data Integration, Data Independence.

Data Models: Introduction, Types of Data Models, Entity-Relationship Model: E-R diagram, Reduction to relational schemas, Generalization, Specialization & Aggregation. The Relational Model: Keys, Relationship, Integrity rules, Relational Algebra.

UNIT –II

SQL, Intermediate SQL and Relational Database Design:

SQL: Overview of SQL, DDL, integrity constraints, DML, set operations, null values, aggregate functions, sub-queries.

Intermediate SQL: Joins, Views, Indexes, Abstract Data type.

UNIT –III

Advanced SQL: PL-SQL.

Relational Database Design: Functional Dependency, Normalization.

UNIT-IV

File Organization, Indexing and Hashing:

Introduction, Ordered indices, B-Tree and B+-Tree file organization, Static & Dynamic hashing.

Query Processing and Optimization: Query Processing: Overview, Measures of Query Cost, Selection Operation, Join Operation.

Query Optimization: Overview, Transformation of Relational Expressions, Cost-Based Optimization, Heuristic Optimization.

UNIT-V

Concurrency Control and Database Recovery:

Concept of Transaction, Serializability, locking protocols.

Deadlock Detection and Recovery, Log based Recovery, Recovery with concurrent transactions.

Text Books:

1.Database Systems Concepts: Silberschatz, Korth, Sudarshan, McGraw-Hill.
2.An Introduction to Database Systems: Bipin C. Desai, Galgotia.
3.SQL & PL/SQL using Oracle: Ivan Bayross, BPB Publications.

Reference Books:

1.Fundamental of Database Systems: Elmasri, Navathe, Somayajulu, Gupta Pearson Publications
2.Database Management System: Raghu Ramkrishan, Johannes, McGraw Hill
3.An Introduction to Database Systems: C.J.Date, Narosa

UNIT -I

- **Introduction to Database Management Systems:**

Introduction, Conventional File Processing System,

Components of DBMS, Advantages and Disadvantages, Three-level Architecture proposal for DBMS, Abstraction and Data Integration, Data Independence.

Data Models: Introduction, Types of Data Models, Entity-Relationship Model: E-R diagram, Reduction to relational schemas, Generalization, Specialization & Aggregation. The Relational Model: Keys, Relationship, Integrity rules, Relational Algebra.

Introduction

- When the internet started to be globally available, the data that was created every day started to pile up rapidly. The storage requirements went from gigabytes to exabytes, and then came cloud databases.
- Have you ever imagined how much data is generated when you go on a trip to the Maldives by flight?
- Make a blind guess! 50 GB or 100 GB?
- A minimum of 500 GB of data will be generated when you travel to the Maldives by plane.

- Well, we can use a database management system to do so.

Introduction

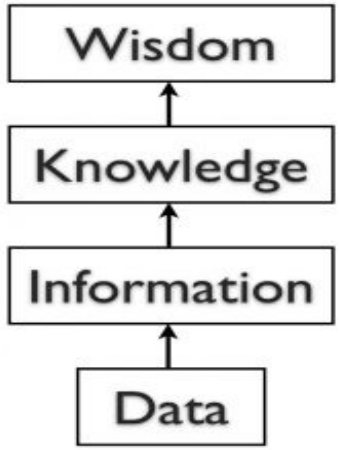
- All business activities deal with a lot of data.
- Examples:
  - ✔ Schools, colleges and universities store data about students, courses, trainers, etc.
  - ✔ Banks store data about their customers, transactions (deposits, withdrawals), loans, etc.

What is Data?

- Data is collection of known facts that can be recorded, that have implicit meaning and from which conclusion may be drawn
- In computer science, “data is anything in a form suitable for use with a computer.”
- Data is often distinguished from programs.
- A program is a set of instructions that detail a task for the computer to perform.
- In this sense, data is thus everything that is not program code.

Cont….

What is Data?



- Data are the raw bits and pieces of information with no context. If I told you, "15, 23, 14, 85," you would not have learned anything. But I would have given you data.

- Data can be quantitative or qualitative.
- Quantitative data is numeric, the result of a measurement, count, or some other mathematical calculation.
- Qualitative data is descriptive. "Ruby Red," the color of a 2013 Ford Focus, is an example of qualitative data.
- By itself, data is not that useful.
- To be useful, it needs to be given context.
- Returning to the example above, if I told you that "15, 23, 14, and 85″ are the numbers of students that had registered for upcoming classes, that would be information.
- By adding the context – that the numbers represent the count of students registering for specific classes – I have converted data into information.

What is Data?

- Once we have put our data into context, aggregated and analyzed it, we can use it to make decisions for our organization.
- We can say that this consumption of information produces knowledge. This knowledge can be used to make decisions, set policies, and even spark innovation.
- The final step up the information ladder is the step from knowledge (knowing a lot about a topic) to wisdom. We can say that someone has wisdom when they can combine their knowledge and experience to produce a deeper understanding of a topic. It often takes many years to develop wisdom on a particular topic, and requires patience.

Examples of Data

- Almost all software programs require data to do anything useful.
- For example, if you are editing a document in a word processor such as Microsoft Word, the document you are working on is the data.
- The word-processing software can manipulate the data: create a new document, duplicate a document, or modify a document.
- Some other examples of data are: an MP3 music file, a video file, a spreadsheet, a web page, and an e-book.
- In some cases, such as with an e-book, you may only have the ability to read the data.

What is Database?

- The database is a big container where data is stored in a structured format. We cannot store semi-structured or unstructured data in a database.

- A database is an organized collection of data that can be modified, retrieved, or updated. Data, DBMS, and applications associated with them together form the database concept.
- The data, stored in the database, is in the row and column format, which is called a table. Every website, which needs us to sign up, uses a database. There is no internet without databases.

- For instance, a college will have to keep the information about its students, including roll number, name, age, blood group, etc. The college will also need to keep the details of the professors and infrastructure. The details, which the college has, can be stored in a database named College, or if it is just the student details, then it can be named Students. All such details should be in a structured format, such as tables, in a hierarchy.

What is Database?

- "A database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated."

- "A database is a collection of data, typically describing the activities of one or more related organizations."

- "Database is a structured collection of records or data that is stored in a computer system."

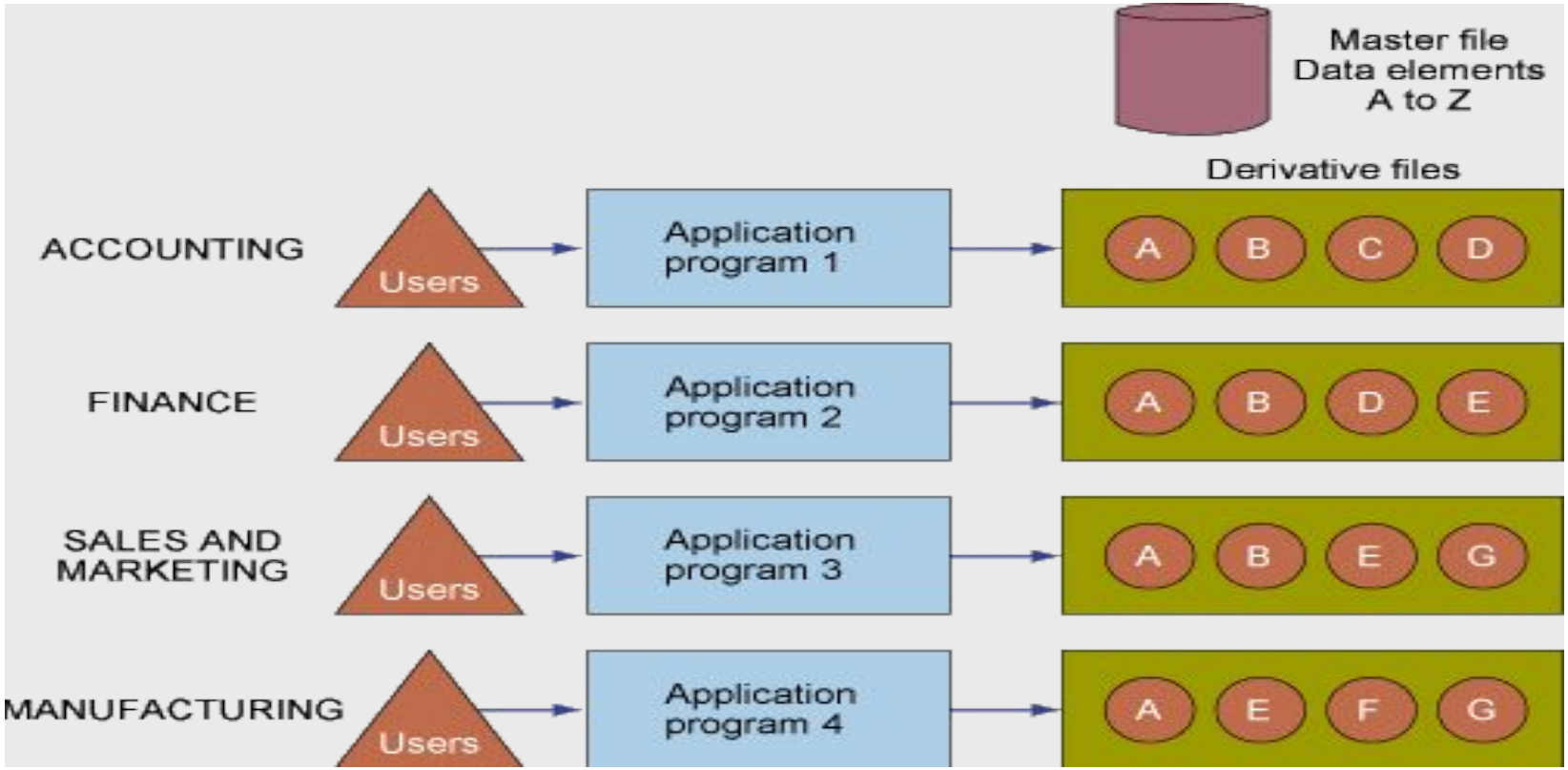
Cont…

Database Management System Examples

- Few examples of database are:
- **Oracle** – Oracle is a Relational Database Management System. It can be stored on-site or in the cloud. It uses enterprise-scale technology to offer a wide range of features to the users.
- **MySQL** – Used by platforms like Youtube, Twitter, and Facebook, MySQL is a Relational Database Management System. It is often integrated with open-source Content Management Systems (CMS).
- **SQL Server** – SQL Server is a Relational Database Management System developed by Microsoft. It was based on SQL, a query language that helps users in data query and database management.

Conventional File Processing System

- Data is stored in files
- Each file has a specific format
- Programs that use these files depend on knowledge about that format

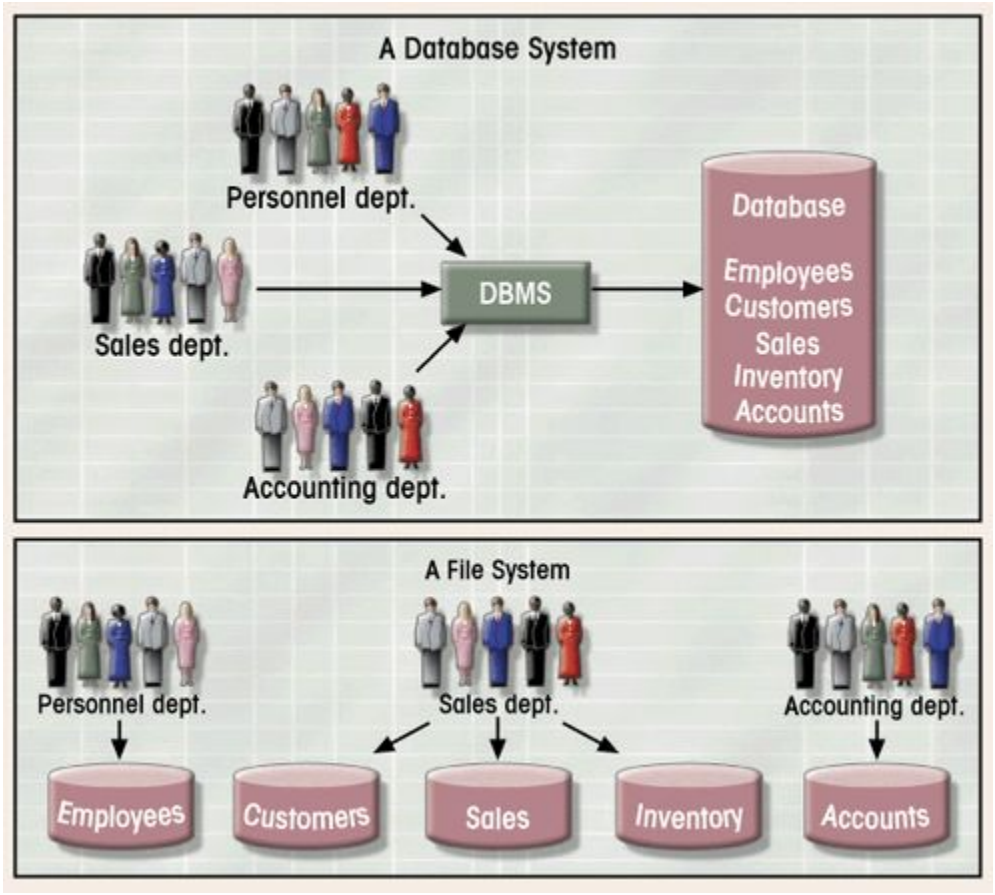


Conventional File Processing System

- **Data stored permanently in operating system files**
- **It is a simple disconnected system.**
- **Application creates format specific data**
- **Rigorous and adhoc programming**

Database vs. File Systems

Figure 1.6



17

What is Database Management System?

- **A Database Management System (DBMS), or simply a Database System (DBS) consist of :**
  - A collection of interrelated and persistent data (usually referred to as the database (DB)).
  - A set of application programs used to access, update and manage that data.
- **The goal of a DBMS is to provide an environment that is both convenient and efficient to use in :**
  - Retrieving information from the database.
  - Storing information into the database.

Cont….

What is Database Management System?

- **Databases are usually designed to manage large bodies of information. This involves**
  - Definition of structures for information storage (data modeling).
  - Provision of mechanisms for the manipulation of information (file and systems structure, query processing).
  - Providing for the safety of information in the database (crash recovery and security).
  - Concurrency control if the system is shared by users.

Count….

Goals of a Database Management System:

- To provide an efficient as well as a convenient environment for accessing data in a database
- Enforce information security: database security, concurrency control, crash recovery

It is a general purpose facility for:

- Defining database
- Constructing database
- Manipulating database

History of DBMS

- **1960 – First DBMS designed by Charles Bachman at GE. IBMs Information Management System (IMS)**
- **1970 – Codd introduced the RDBMS**
- **1980 – Relational model became popular and accepted as the main database paradigm. SQL, ANSI SQL, etc.**
- **1980 to 1990 – New data models, powerful query languages, etc. Popular vendors are Oracle, SQL Server, IBMs DB2, Informix, etc.**

DBMS Applications

- **Banking: (Customer, Accounts, Loans, Transactions)**
- **Airline / Railway: (Reservation)**
- **University: Student, Course Registration, Grades**
- **Telecommunication: Calls, Bills**
- **Human Resource: Employee, Salary, Leave**
- **Manufacturing: Inventory**

Drawbacks of Conventional File Processing System

In the early days, database applications were built directly on top of file systems

1. **Data redundancy and inconsistency**

- **Multiple file formats, duplication of information in different files**
- **Duplication means same data is held by different programs**
- **Wasted space and potentially different values and/or different formats for the same item**
- **Different functions collect the same information independently**
- **May have different meanings in different parts of the organisation**

Count…

Data Redundancy

Branch Relation

Branch_No	*BAddress*	*Tel_No*
B5	22 Deer Rd, London	0171-886-1212
B7	16 Argyll St, Aberdeen	01224-67125
B3	163 Main St, Glasgow	0141-339-2178

Staff Relation

Staff_No	*SName*	*SAddress*	*Position*	*Salary*	*Branch_No*
SL21	John White	19 Taylor St, London	Manager	30000	B5
SG37	Ann Beech	81 George St, Glasgow	Snr Asst	12000	B3
SG14	David Ford	63 Ashby St, Glasgow	Deputy	18000	B3
SA9	Mary Howe	2 Elm Pl, Aberdeen	Assistant	9000	B7
SG5	Susan Brand	5 Gt Western Rd, Glasgow	Manager	24000	B3
SL41	Julie Lee	28 Malvern St, Kilburn	Assistant	9000	B5

Staff_Branch relation has redundant data; the details of a branch are repeated for every member of staff.

In contrast, the branch information appears only once for each branch in the Branch relation and only the branch number (Branch_No) is repeated in the Staff relation, to represent where each member of staff is located.

Staff_Branch Relation

Staff_No	*SName*	*SAddress*	*Position*	*Salary*	*Branch_No*	*BAddress*	*Tel_No*
SL21	John White	19 Taylor St, London	Manager	30000	B5	22 Deer Rd, London	0171-886-1212
SG37	Ann Beech	81 George St, Glasgow	Snr Asst	12000	B3	163 Main St, Glasgow	0141-339-2178
SG14	David Ford	63 Ashby St, Glasgow	Deputy	18000	B3	163 Main St, Glasgow	0141-339-2178
SA9	Mary Howe	2 Elm Pl, Aberdeen	Assistant	9000	B7	16 Argyll St, Aberdeen	01224-67125
SG5	Susan Brand	5 Gt Western Rd, Glasgow	Manager	24000	B3	163 Main St, Glasgow	0141-339-2178
SL41	Julie Lee	28 Malvern St, Kilburn	Assistant	9000	B5	22 Deer Rd, London	0171-886-1212

Drawbacks of Conventional File Processing System

2. **Difficulty in accessing data**

- Need to write a new program to carry out each new task

3. **Data isolation**

– **Multiple files and formats**

Each program maintains its own set of data.

Users of one program may be unaware of potentially useful data held by other programs.

Drawbacks of Conventional File Processing System

4. **Integrity problems**
   - **Integrity constraints (e.g. account balance > 0) remain in program code rather than being stated explicitly**
   - **Hard to add new constraints or change existing ones**

5. **Atomicity of updates**

- **Atomicity means that all transactions must follow all or nothing rule.**
- **Failures may leave database in an inconsistent state with partial updates carried out**

Example: Transfer of funds from one account to another should either complete or not happen at all

Drawbacks of Conventional File Processing System

6. **Concurrent access by multiple users**

- **Concurrent access needed for performance**
- **Uncontrolled concurrent accesses can lead to inconsistencies**

Example: Two people reading a balance and updating it at the same time

– **Hard to get hands on information**

– **Different pieces of information in different files and different physical locations**

– **Since files in different locations can't be related, hence these files are hard to share or access in a timely manner**

– **Impossible for information to flow freely**

Drawbacks of Conventional File Processing System

7. **Security problems**

- **Hard to provide user access to some, but not all data**
- **There is little or no control and management of data**
- **Data could be disseminated all over the organisation without control**
- **Unable to find who is accessing the data and making changes.**

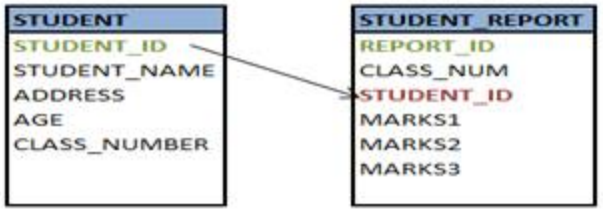
Advantages of DBMS

- **Efficient data access**
- **Concurrent access**
- **Redundancy can be reduced**
- **Inconsistency can be avoided**
- **Data can be shared**
- **Standards can be enforced**
- **Security restrictions can be applied**
- **Integrity can be maintained**
- **Data independence can be provided**
- **Backup and Recovery**
- **Reduced application development time**
- **Restricting unauthorized access to data.**

Advantages of DBMS

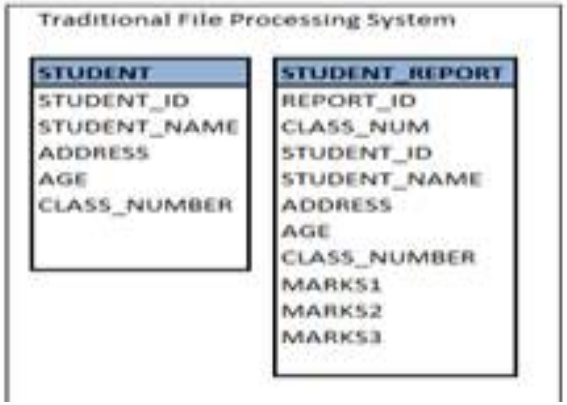
1. **Data Mapping and Access: -**

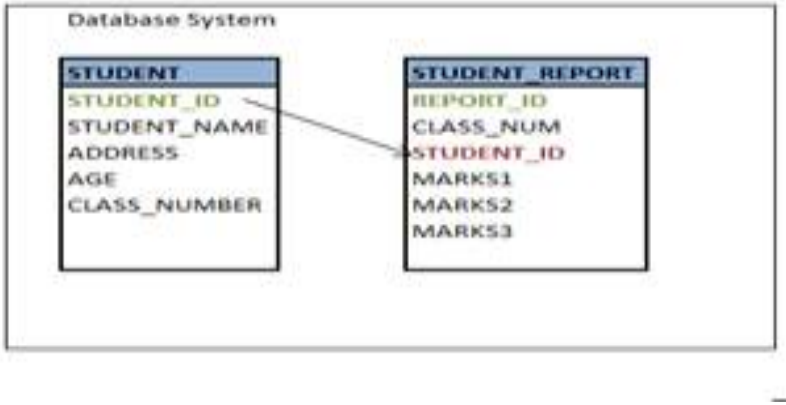
- **DBMS defines the way to map any two related tables by means of primary key –foreign key relationship.**
- **Primary key is the column in the table which is responsible for uniquely identifying each record in the table.**
- **Foreign key is the column in the table which is a primary key in other table and with which the entries in the current table are related to other table.**
- **For example, in the new database system, we can Student and Student_Report table as follows.**
- **STUDENT_ID is the unique using which we can identify each student and hence it forms a primary key in STUDENT table. Similarly, REPORT_ID is the primary key in the STUDENT_REPORT table. STUDENT_ID in the STUDENT_REPORT table is the foreign key. It links STUDENT and STUDENT_REPORT tables. Because of such mapping, it becomes easier for the programmer to search for related tables, join them, and fire the query as per the requirement.**
- **This reduces the time consumed while searching and mapping these tables. Even when there is large amount of data, the time consumed to retrieve, insert, update or delete is very less. Hence there is no data isolation in the system.**



2. Data Redundancy: -

- By the introduction of primary key in the table, data redundancy problem is reduced to great extent.
- As we saw, primary key is the unique column for each record, when there is a re-entry of same record, it does not allow saving such records.
- DBMS has strong designing techniques like normalization which makes sure the same copy of data is not stored in same table or in multiple tables.
- It makes sure all the information's are stored only once in the database tables.
- We can see the difference in the way data is being stored in the file and database system.
- Primary key, foreign keys are defined; unnecessary columns are removed from the STUDENT_REPORT table in the database system. These are missing in the file processing system.





3. Data Independence and Consistency: -

- **DBMS defines a standard to represent the data in the form of rows and columns.**
- **It also stores the information about the tables, columns, keys, storage space, used space, available space etc separately from the logical data. Hence they totally independent of the way they are stored and the data being stored.**
- **Any changes to the physical storage (like disks, tapes etc) or structure, does not harm the data being stored. Since DBMS defines each columns and rows at the beginning itself and controls the way data being entered, there is no affect on the programs or any other tables or data. Hence consistency of the data is also maintained.**
- **If there is a change in the address of any student, we just have to update it in the Student table.**
- **There is no other place his information is being stored. Hence it maintains the consistent data in the database. Suppose there is a new column addition to STUDENT table, say DOB.**
- **This will change the metadata to reflect additional column in the table structure.**
- **It will hardly affect the application unless until there is a new requirement to have transaction with DOB. Hence data independence is also assured in the database.**

- **4. Security: -**
- **DBMS allows different levels of access to different users based on their roles.**
- **In the school database, individual students will have access to their data alone, while their teachers will have access to all the students whom they are teaching and for the subjects that they are teaching.**
- **Class teacher will be able to see the reports of all the students in that class, but not other classes.**
- **Similarly, in a banking system, individual account holder will have Read-Only access to their account. While accountant can update, individual account details for each of their transaction.**
- **All these levels of security and access are not allowed in file system.**

- **5. Integrity: -**

- **DBMS allows having restrictions on individual columns. It would be defined while designing the table itself.**
- **If we want to enter salary of an employee within the range 10000 to 40000, we can impose this while designing the table by using CHECK constraint.**
- **When salary is entered, it will automatically check for the range specified.**
- **CREATE TABLE EMPLOYEE**
- **CONSTRAINT chk_salary CHECK (salary>10000 AND salary <40000)**

- **6. Atomicity: -**

- **DBMS makes sure either the transaction is fully complete or it is rolled back to the previous committed state.**
- **It does not allow the system to be in a partially committed state. In our example above, DBMS commits marks change transaction before calculating the total.**
- **If there is any crash or shutdown of the system, before committing the marks, then updated marks will be rolled back to the original marks. Hence it makes sure atomicity of the transaction is achieved.**

- **7. Concurrent Access: -**
- **DBMS provides access to multiple users to access the database at the same time.**
- **It has its own mechanism to have concurrency accesses and hence avoid any incorrect data in the system.**

Disadvantages

- It is bit complex.
- Since it supports multiple functionality to give the user the best, the underlying software has become complex.
- The designers and developers should have thorough knowledge about the software to get the most out of it.
- Because of its complexity and functionality, it uses large amount of memory.
- It also needs large memory to run efficiently.
- DBMS system works on the centralized system, i.e.; all the users from all over the world access this database.
- Hence any failure of the DBMS, will impact all the users.
- DBMS is generalized software, i.e.; it is written work on the entire systems rather specific one.
- Hence some of the application will run slow.

Components of DBMS

DBMS Users

- Naive user
- Online user
- Application Programmer
- Sophisticated user
- Specialized user
- DBA

DBMS USERS

- **Naive users**
  - They are not aware about dbms.
  - Little knowledge of computers.
  - Interact with database through application programs, forms interface(end user).
  - Users are instructed at each step

  Eg. Reservation Clerks of Airline, Railway, Hotel, etc.
  Clerks at receiving station of Courier service, Insurance agencies, etc.

- **Online users**
  - Users may communicate with the database directly via terminal or indirectly via interface & app. Programs.
  - Little bit knowledge of DBMS & can perform DML operation, some are naive and require help menu

DBMS USERS

- **Application programmers**
  - professional programmers who are responsible for developing application programs or user interfaces utilized by the naïve and online users. They interact with system through DML calls
  - write programs. Make use of RAD (rapid application development) tools to develop forms & reports for naïve users
- **Sophisticated users**
  - use query language to retrieve data.
  - E.g. analyst
  - Use Online Analytical Processing or Data Mining tools
- **Specialized users**
  - write special programs for DBMS, Expert systems,
  - Handle audio,video data
  - use AI tools

DBMS USERS

- **DBA**
  - **Coordinates all the activities of the database system;**
  - **Schema definition**
    - Write & creates database schema
    - Three level schema definition / modification as per changes, to improve performance , Mapping between views
  - **Performance tuning**
    - File organization & access-method definition / modification
  - **Security**
    - Authentication, Granting authorization for data access to users
  - **Routine maintenance**
    - backup, recovery
    - space management
    - monitoring jobs running on database

DBMS facilities

- DDL - create, alter, drop, truncate
- DML – insert, delete, update, select

  – Procedural DML

    require user to specify what data & how to get that data

  – Non procedural (declarative) – require user to specify what data needed without specifying how to get that data. DML component of SQL is nonprocedural.

  - QUERY – statement in query language used to retrieve data

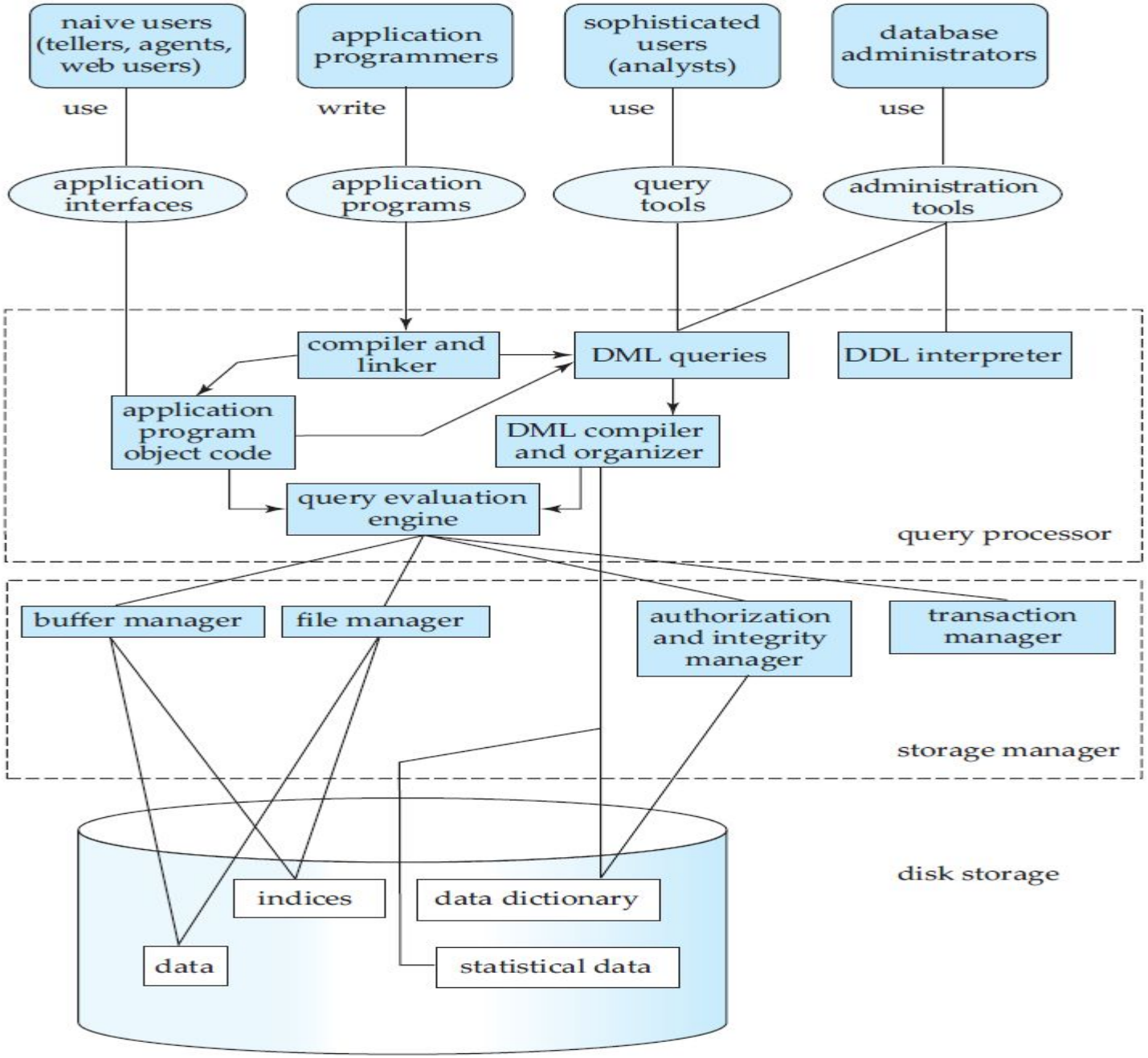


Figure 1.5 System structure.

Data Storage and Querying

- Components of a DBMS can be broadly divided into the **storage manager and the query processor components.**
- The storage manager is important because databases typically require a large amount of storage space.
- Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data.
- A gigabyte is approximately 1000 megabytes (actually 1024) (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes).
- **Since the main memory of computers cannot store this much information, the information is stored on disks.**
- **Data are moved between disk storage and main memory as needed.**

Query processor

- Since the **movement of data to and from disk is slow** relative to the speed of the central processing unit, it is imperative that the **database system structure the data so as to minimize the need to move data between disk and main memory.**
- The query processor is important because it helps the database system to simplify and facilitate access to data.
- The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system.
- It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

Query processor
Components

- **DDL interpreter: it interprets DDL statements and records the definitions** in the data dictionary.
- **DML compiler: it translates DML statements in a query language into an** evaluation plan consisting of low-level instructions that the query evaluation engine understands. A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization; that is, it picks the lowest cost evaluation plan from** among the alternatives.
- **Query evaluation engine: it executes low-level instructions generated** by the DML compiler.

Storage Manager

- The ***storage manager*** *is the component of a database system that provides the* **interface** between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the interaction with the file manager.
- The raw data are stored on the disk using the file system provided by the operating system.
- The storage manager translates the various DML statements into low-level file-system commands.
- Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.
- The storage manager components include:
- **Authorization and integrity manager:** it tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager:** it ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

Storage Manager

- **File manager:** it manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager:** it is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.
- The storage manager implements several data structures as part of the physical system implementation:
- **Data files,** which store the database itself.
- **Data dictionary,** which stores metadata about the structure **of** the database, in particular the schema of the database.
- **Indices,** which can provide fast access to data items.
- Like the index in this textbook, a database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the *instructor* record with a particular *ID, or all instructor records with a particular name.*
- **Hashing** is an alternative to indexing that is faster in some but not all cases.

Transaction manager

- A **transaction is a collection of operations that performs a single logical** function in a database application. Each transaction is a unit of both atomicity and consistency.

- Ensuring the atomicity and durability properties is the responsibility of the database system itself—specifically, of the **recovery manager. In the absence of** failures, all transactions complete successfully, and atomicity is achieved easily.

- The database system must therefore perform **failure recovery,** that is, detect system failures and restore the database to the state that existed prior to the occurrence of the failure.
- Finally, when several transactions update the database concurrently, the consistency of data may no longer be preserved, even though each individual transaction is correct.

- It is the responsibility of the **concurrency-control manager to control** the interaction among the concurrent transactions, to ensure the consistency of the database.

- The **transaction manager consists of the concurrency-control** Manager and the recovery manager.

Three-level Architecture proposal for DBMS

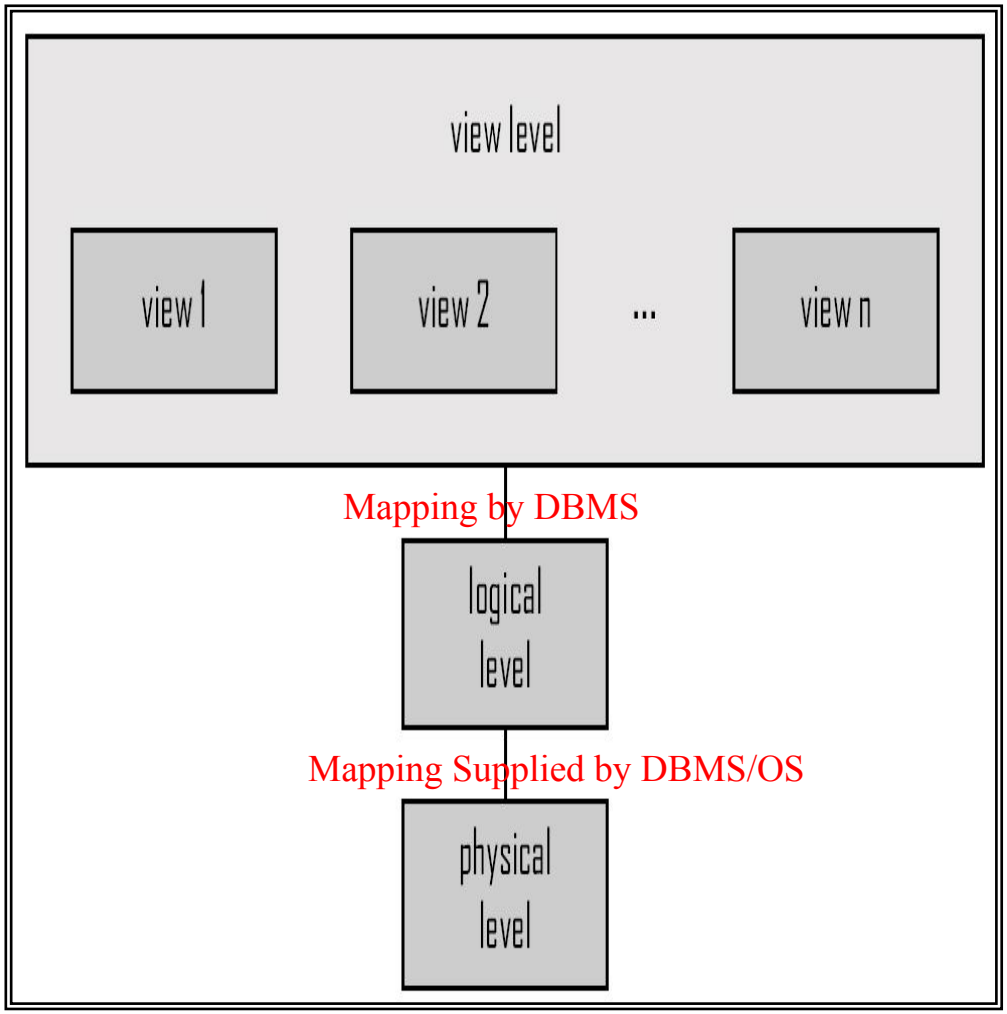
- Most commercial databases are based on three level architecture model called ANSI/SPARC (American National Standards Institute / Standard Planning and Requirements Committee).
- Architecture is divided into Internal, Conceptual and External Level

Three-level Architecture proposal

- Requirement via user or Application
- In consultation with DBA
- DML

- What?
- For Enterprise
- By DBA
- DDL

- How?
- For optimization
- Low level programmer /DBA
- SDL



Customer_Loan

Cust_ID :
Loan_no :
Amount :

Create table customer_loan(cust_id Number(4)

loan_no Number(4)

amount Number(7,2))

Cust_id type=byte(4),offset=0
Loan_no type byte(4),offset 4
Amount type byte(7), offset 8

Three-level Architecture proposal for DBMS

- **External / View Level**
  - **Highest Level**
  - **Many users are not concerned with all the information in the database. Instead, they need portion of database**
  - **Depending upon number of applications, system may provide number of views for the same database.**
- **Conceptual / Logical Level**
  - **Middle Level**
  - **Describes data in the database and relationship among data**
  - **DBA decide what information must be kept in database**
- **Internal / Physical Level**
  - **Lowest Level**
  - **Describes how data is stored in terms of bytes, its consecutive storage locations and access methods**
  - **DBA is aware about certain details**

Data Abstraction

- **Efficiency & Convenience are the two main goals of DBMS**
- **Efficiency require to use complex data structures while convenience require simplicity**
- **All users are not computer trained so Complex data structures made simple through abstraction for top level users**
- **DBMS allow to hide this complexity (how data is stored and maintained) from end users through three levels of abstraction, to simplify users interaction with the system.**
- **Security is the major concern.**

Data Abstraction

- Data Abstraction refers to the process of hiding irrelevant details from the user.
- For efficient retrieval of complex data, database need a good data structures to represent data.
- Developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system
- **Example:** If we want to access any mail from our Gmail then we don't know where that data is physically stored i.e is the data present in India or USA or what data model has been used to store that data? We are not concerned about these things.
- We are only concerned with our email.
- So, information like these i.e. location of data and data models are irrelevant to us and in data abstraction, we do this only. Apart from the location of data and data models, there are other factors that we don't care of. We hide the unnecessary data from the user and this process of hiding unwanted data is called Data Abstraction.

Levels of abstraction

- **Physical level:**
  - describes complex low level data struc
  - Describes how data is actually stored.
  - System level programmer decides schema but to some extent DBA is aware about this
  - Suppose we need to store the details of an employee. Blocks of storage and the amount of memory used for these purposes is kept hidden from the user.

Cust_id type=byte(4),offset=0
Loan_no type=byte(4),offset 4
Amount type=byte(7), offset 8

cust_id Number(4);
loan_no Number(4);
Amount Number(7,2);

- **Logical level:**
  - Complexity at physical level is reduced
  - Describes what data are stored in database, and the relationships among the data.
  - Contain information of entire database
  - DBA decides schema
  - but to some extent programmer know this

Levels of abstraction

- **View level**
  - Reduce complexity at conceptual level
  - application programs hide details of data types.
  - All information in the database stored at conceptual level is not needed for the application
  - System provide many views of the same database as per the users need
  - Users can just view the data and interact with the database
  - storage and implementation details are hidden from them.

Cust_ID : 101
Loan_no : 1011
Amount : 50,000.00

Data Independence

- The main purpose of **data abstractio**n is to achieve data independence in order to save time and cost required when the database is modified or altered.
-
- **Data Independence** means users and data should not directly interact with each other.

- The user should be at a different level and the data should be present at some other level.

- We have namely two levels of data independence arising from these levels of abstraction :

Data Independence

- Achieved through the use of 3 levels of abstraction

> The ability to modify a schema definition in one level without affecting a schema definition in higher level is called data independence

mainly two kinds of database independence

a) Logical data independence
b) Physical data independence

Logical data independence

✓ The ability to change logical schema without changing external schema or application program is called logical data independence

Ex:

faculty(fid: string , fname: string , office : integer , sal: real)

Divided into

fac_public (fid: string , fname: string , office : integer)

fac_private(fid: string, sal: real)

but the user who query will get the same result.

Physical data independence

✓ The ability to change physical schema without changing logical schema is called Physical data independence

Changes in physical schema may include

- ✓ Using new storage device
- ✓ Using different data structure
- ✓ Switching from one access method to another
- ✓ using different storage structures
- ✓ modifying indexes etc.

Data Model

- A data model in software engg. is an abstract model.
- It describes how data are represented and accessed.
- Defines data elements and relationships among data elements.
- Typical applications of data models include database models, design of information systems, and enabling exchange of data.
- A data model is the medium which project team members from different backgrounds and with different levels of experience can communicate with one another.
- Data models describe structured data for storage in data management systems such as relational databases.

Types of data models

- **Object Based Logical Model**
- **Record Based Logical Model**

Object Based Logical Model

Entity-Relationship Model (E-R Model) is a widely known **object based logical model**.

- They are used to describe data at the conceptual and the view level.
- The E-R Model is based on the perception of the real world that consists of a collection of basic objects called entities, and of relationships among these objects.

Record Based Logical Model

- They are used to describe data at the conceptual and the view level.
- They are used both to specify the overall logical structure of the database and to provide a higher level description of the implementation.
- There are three widely accepted record based logical models.

Record Based Logical Model

- **Hierarchical Data Model**
- **Network Data Model**
- **Relational Data Model**

Hierarchical Data Model

- The hierarchical data model organizes data in a tree structure.
- There is a hierarchy of parent and child data segments.
- This structure implies that a record can have repeating information, generally in the child data segments.
- Data is represented by a collection of records (record types).
- A record type is the equivalent of a table in the relational model, and with the individual records being the equivalent of rows.
- To create links between these record types, the hierarchical model uses parent-child relationships.

Hierarchical Data Model

- In a hierarchical database the parent -child relationship is one to many.
- This restricts a child segment to having only one parent segment .
- Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.
- *Example: Consider the banking system. Figure 1-20 shows the hierarchical* representation of Customer_Details and Customer_Loan records from
- Customer_Details and Customer_Loan files respectively.
- Note: Loan (Loan_No: 1011) is shown as taken jointly by Mike A. Smith and Graham S. Smith to explain the difference between hierarchical and network Model.

Hierarchical Data Model

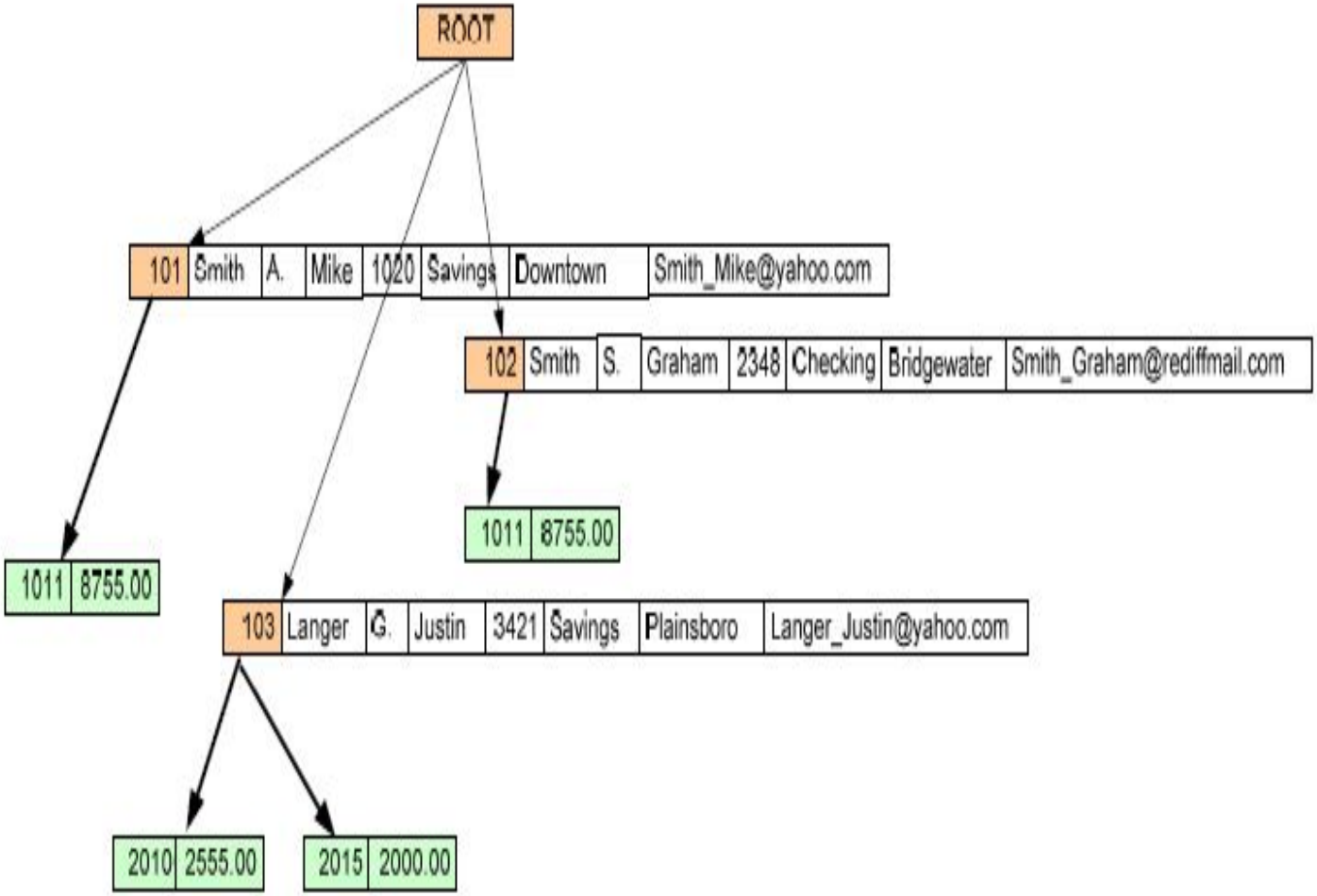


Figure 1-20: Hierarchical Model

Hierarchical Data Model

- **Advantages**
  - Conceptual simplicity
  - Database security
  - Data independence
  - Database integrity
  - Efficiency

Disadvantages

- Complex implementation
- Difficult to manage
- Lacks structural independence
- Complex applications programming and use
- Implementation limitations
- Lack of standards

Network Data Model

- The network model permitted the modeling of many-to-many relationships in data.
- In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model.
- The first network DBMS was implemented by Honeywell in 1964-65 (IDS System).
- Adopted heavily due to the support by CODASYL (Conference on Data Systems Languages) (CODASYL - DBTG report of 1971).
- Later implemented in a large variety of systems - IDMS (Cullinet - now Computer Associates), DMS 1100 (Unisys), IMAGE (H.P. (Hewlett-Packard)), VAX -DBMS (Digital Equipment Corp., next COMPAQ, now H.P.).
- Represent complex data relationships more effectively
- Improve database performance
- Impose a database standard
- It is accepted by American National Standards Institute (ANSI) as standard cobol specifications.
- Database Task Group (DBTG) was created to define standard specifications for database to facilitate creation and manipulation of data.

Network Data Model

- Data in the network model is represented by a collection of records and the relationships among data are represented by links (pointers).

- The records in the database are organized as collect ions of graphs.
- *Example: IDMS.*
- *Example: Refer t o Figure 1-21. Assume that loan (Loan_No:1011) is taken jointly by two customers (Mike A. Smith and Graham S. Smith).*
- In the hierarchical model (Figure 1-20), the loan information has to be repeated for each customer individually because it does not permit many t o many relationship.
- The parent -child relationship is one to many.
- However in the network model, because it allows many to many relationship, the loan information is stored only once and both the customers can refer to it .

Network Data Model

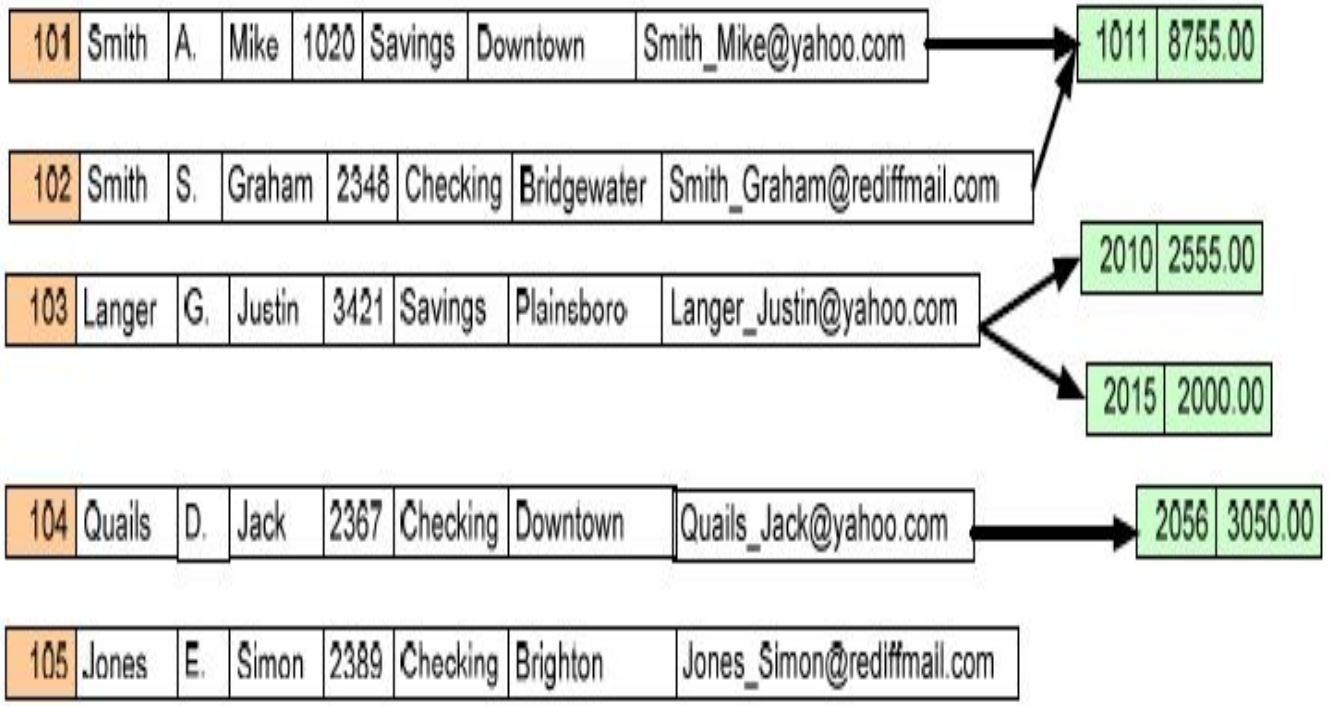


Figure 1-21: Network Model

The Network Data Model

- Advantages
  - Conceptual simplicity
  - Handles more relationship types
  - Data access flexibility
  - Promotes database integrity
  - Data independence
  - Conformance to standards

- Disadvantages
  - System complexity
  - Lack of structural independence

Relational Data Model

- **Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.**
- **Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).**
- **Several free open source implementations, e.g. MySQL, PostgreSQL**
- **Currently most dominant for developing database applications.**
- **SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, …**

- **Considered ingenious but impractical in 1970**
- **Conceptually simple**
- **Computers lacked power to implement the relational model**
- **Today, microcomputers can run sophisticated relational database software**
- **Relational Database Management System (RDBMS)**
- **Performs same basic functions provided by hierarchical and network DBMS systems, plus other functions**
- **Most important advantage of the RDBMS is its ability to let the user/designer operate in a human logical environment**

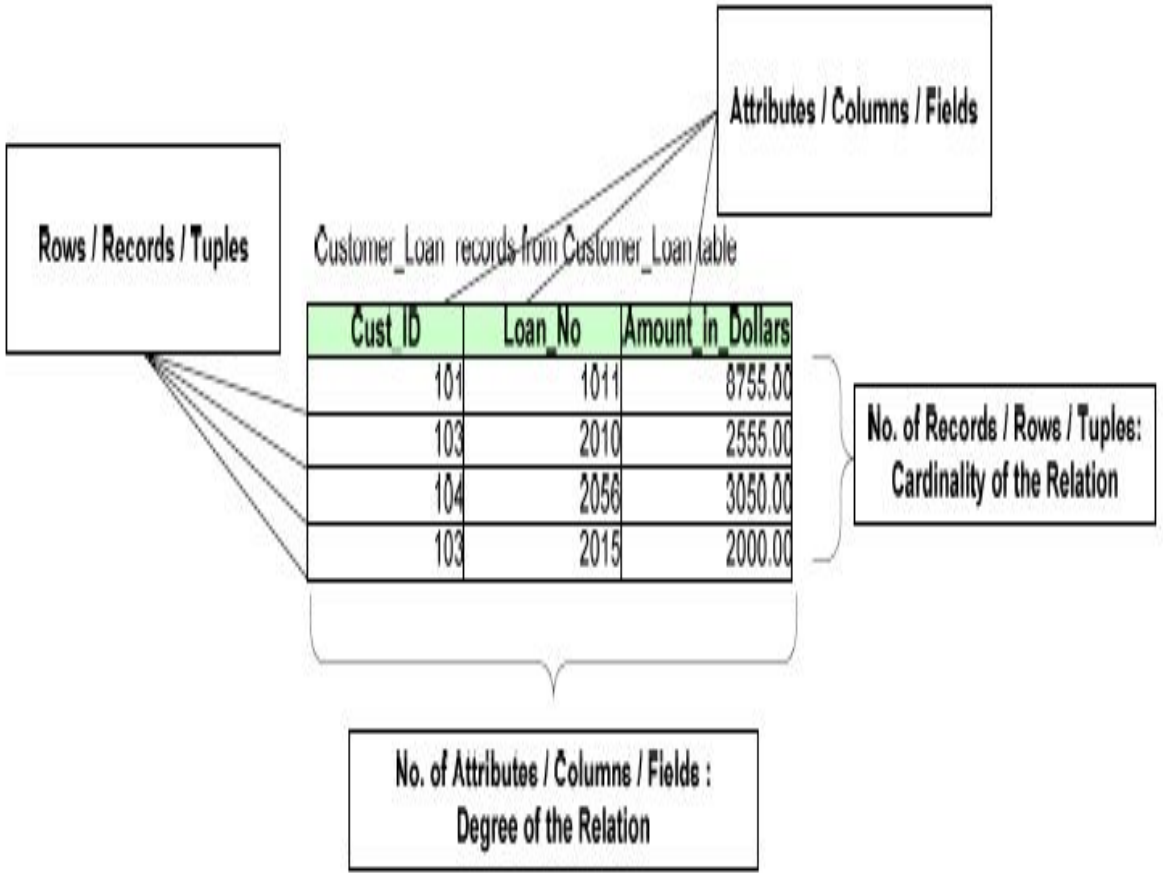
Relational Data Model

- Table (relations)
- Matrix consisting of a series of row/column intersections
- Related to each other by sharing a common entity characteristic
- Relational schema
- Visual representation of relational database's entities, attributes within those entities, and relationships between those entities
- Stores a collection of related entities
- Resembles a file
- Relational table is purely logical structure
- How data are physically stored in the database is of no concern to the user or the designer
- This property became the source of a real database revolution

Relational Data Model

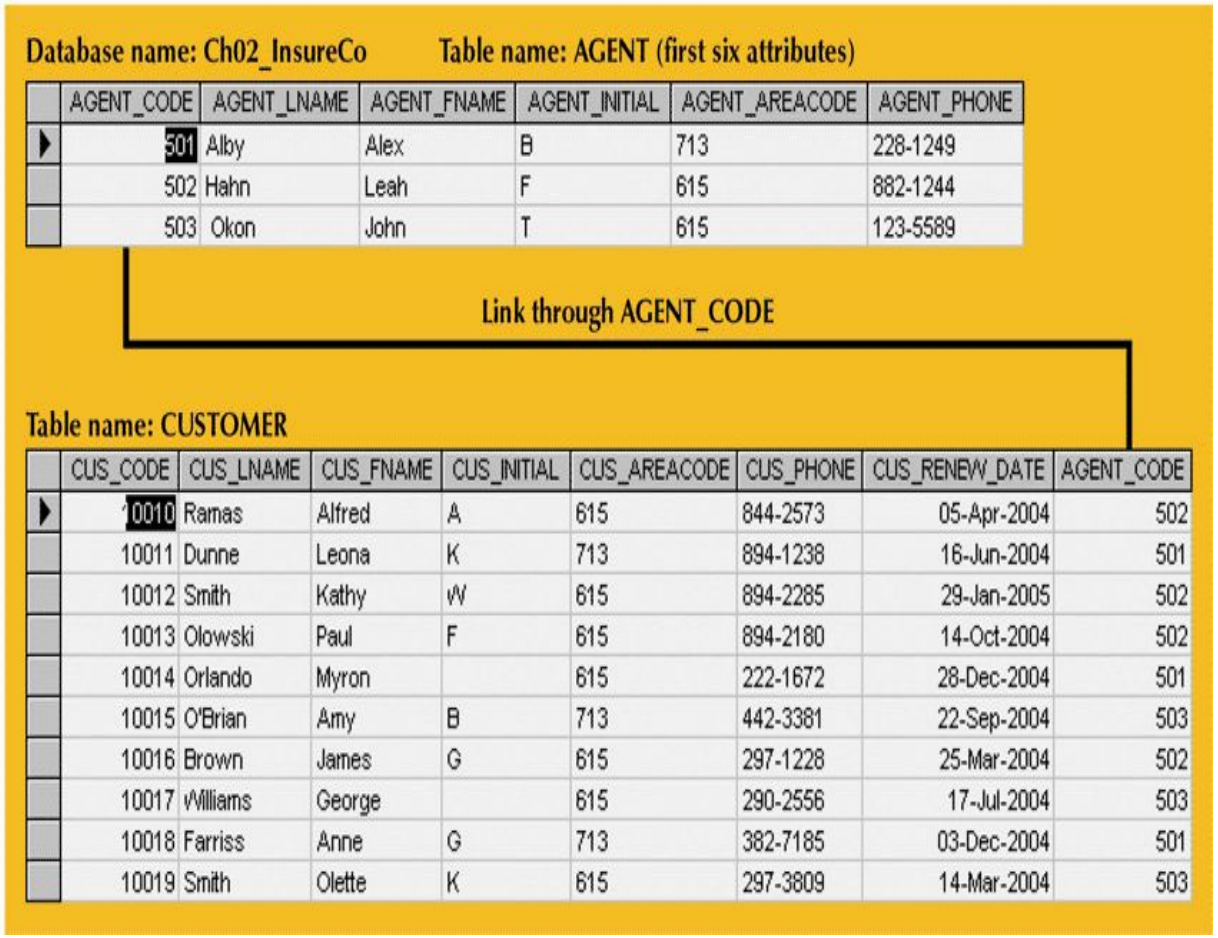
- The relational model uses a collection of tables (relations), each of which is assigned a unique name, to represent both data and the relationships among those data.
- A table has a specified number of columns but can have any number of rows.
- Rows stored in a table resemble records from flat files.
- A row in a table represents a relationship among a set of values.
- Refer to Figure 1-22, a row in the Customer_Loan table gives the details of a loan taken by a customer.
- Example: Customer (Cust _ID:101) has taken a loan (Loan_No: 1011) of amount (Amount_in_Dollars: 8755.00)
- Since a table is a collection of such relationships, there is a close correspondence between the concept of table and the mathematical concept of relation, from which the relational data model takes its name.

Relational Data Model



Linking Relational Tables

FIGURE 2.4 LINKING RELATIONAL TABLES



Database name: Ch02_InsureCo Table name: AGENT (first six attributes)

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

Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2004	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2004	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2005	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2004	502
10014	Orlando	Myron		615	222-1672	28-Dec-2004	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2004	503
10016	Brown	James	G	615	297-1228	25-Mar-2004	502
10017	Williams	George		615	290-2556	17-Jul-2004	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2004	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2004	503

Relational Data Model

- Advantages
- Structural independence
- Improved conceptual simplicity
- Easier database design, implementation, management, and use
- Adhoc query capability
- Powerful database management system
- Disadvantages
- Substantial hardware and system software overhead
- Can facilitate poor design and implementation

Formal Relational Term	Informal Equivalence
Relation	Table
Tuple	Row or Record
Cardinality of a Relation	Number of rows
Attribute	Column or Field
Degree of a Relation	Number of Columns
Primary Key	Unique Identifier
Domain	A pool of values from which the values of specific attributes of specific relations are taken

Entity-Relationship Model

- "A picture is worth a thousand words"
- Generally the business scenarios are complex in nature.
- A software application designer who is not an expert in a particular business domain may fail to capture the exact business requirement to build a software application.
- This is one of the prominent causes of software project failure.

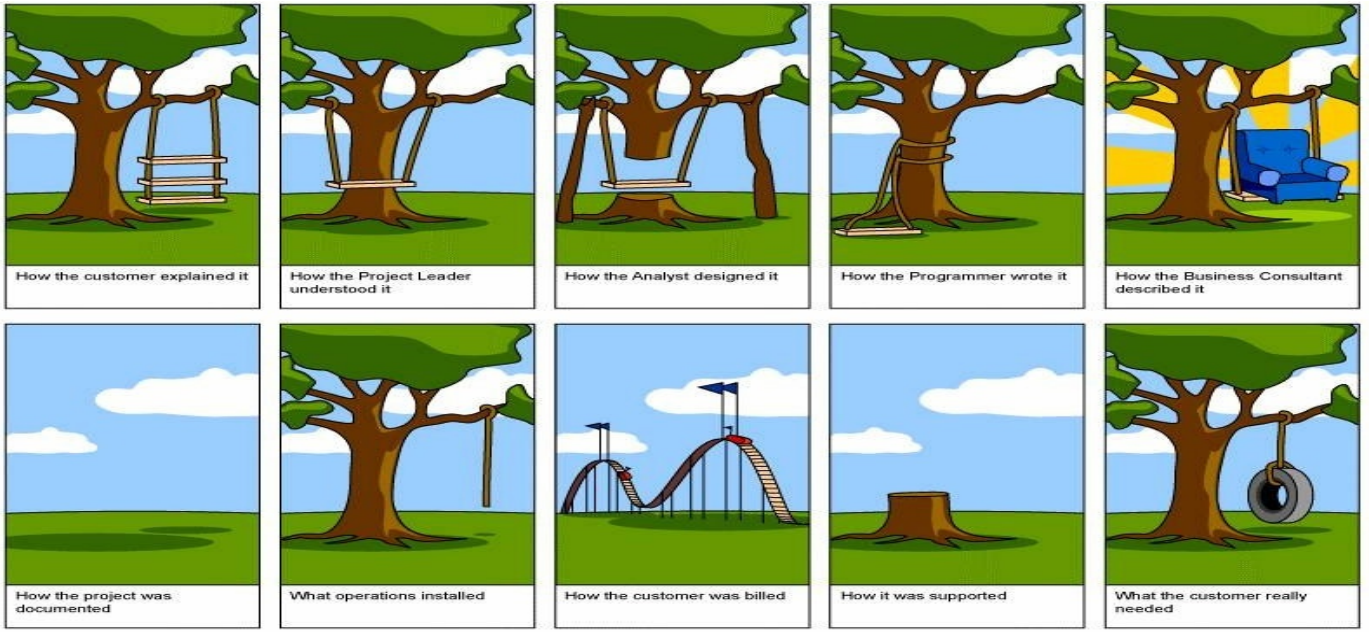


Figure 2-1 explains how miscommunications between different users could create chaos(confusion) in software development projects.

- It is always better to represent all the b*usiness rules in pictorial format so that the* business users can understand and review the business rules easily and correctly.
- One such technique, which is commonly used for designing of the *databases, is* ***Entity*-Relationship Modeling(E-R Modeling).**

- **The diagram used in this technique is called Entity Relationship Diagram (ERD).**

- In Infosys, **60% to 70% of projects use this technique to capture the requirement** specification for the database application design and development .

Entity and Relationship

- **Entity**

  Entity is a common word anything real or *abstract , about which we* want to store data.

- Entity types fall into five categories: roles, events, locations, *things or concepts.*
- Some examples of entities are employee, hockey match, campus, book and department .
- Department is an entity, and Education & Research, HR, Finance, etc., are *instances of the department entity.*

- **Attribute**
- An attribute is a characteristic property of an entity.
- An entity could have multiple attributes.
- ***Example: For an entity car, the attributes would be the***
- *color, model number, number of doors, right or left hand drive etc.*

- **Relationship**
- Relationship is a natural association that exists between one or more entities.
- ***Example: Employee borrows books from the library.***

Cardinality of a relationship

- Cardinality of relationship defines the type of relationship between two participating entities.

- ***Example: One employee can take many books from library.***
- ***One book can be taken*** *by only* ***one employee.***
- ***Cardinality of relationship between employee and book is " one to many" .***
- ***One person can sit on only one chair at any point of time.***
- ***One chair can*** *accommodate only one person in a given point of time.*
- *This relationship has* ***"one to one" cardinality.***
- There are **four types of cardinality relationship.**

- **One to One Relationship**
- **One to Many Relationship**
- **Many to One Relationship**
- **Many to Many Relationship**

One to One Relationship

- In this relationship, one instance of entity is related to another instance of the entity.
- Both participating entities have a one to one relationship.
- **Example1: One person (P1,P2,P3,P4) can sit on only one chair at any** point of time.
- And also one chair (C1,C2,C3,C4) can accommodate a maximum of one person at any given time.
- In this relationship both the participating entities have one-to-one relationship.

Example2: One country can have only one citizen as its president and one citizen can become president of **only one country.**

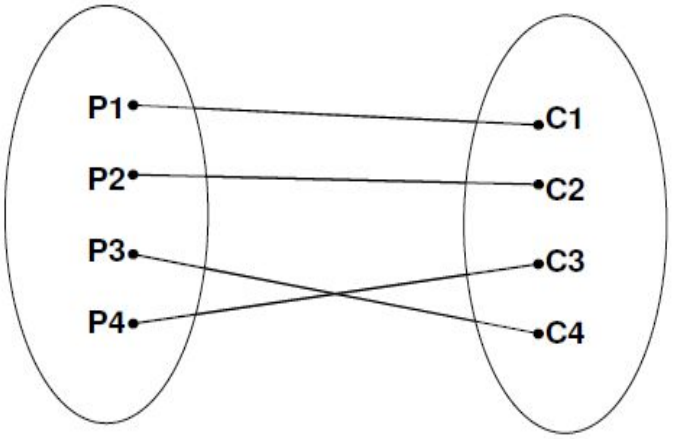


Figure 2-2: One to One Relationship

One to Many Relationship

- One instance of entity is related to multiple instance of another entity.
- ***Example1: One organization(O1,O2,O3) can have many employees but one employee*** *(E1,E2,E3,E4,E5) can work* ***only for one organization.***

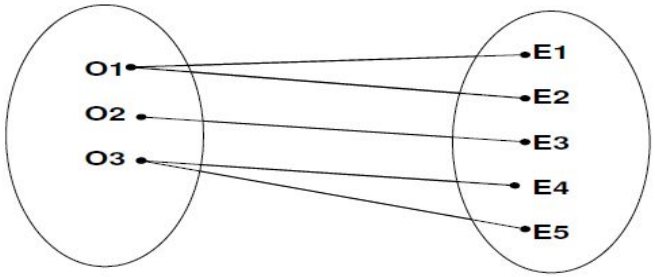


Figure 2-3: One to Many Relationship

Example2: One warehouse (W1,W2,W3) can be used to store many parts but one *part (P1,P2,P3…) can be stored only in one warehouse.*

In this example ***one instance*** *of warehouse accommodates* ***many parts. Hence the relationship between warehouse*** *and part is*

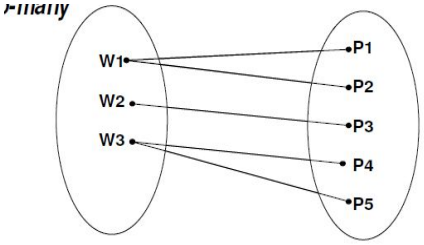


Figure 2-4: Another example of One to Many Relationship

Many to One Relationship

- This is the reverse of the **one to many relationship.**
- ***Example: Many employees (E1,E2,E3…) can work for only one department but one*** *department (D1,D2,D3) can have many employees.*
- *The relationship between employee and departmentis* ***many to one.***

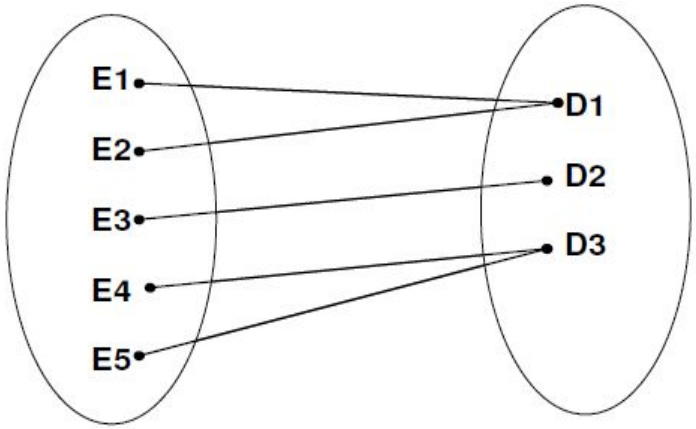


Figure 2-5: Many to One Relationship

Many to Many Relationship

- In this **many to many relationship multiple instances of one Entity are related to** multiple instances of another Entity.
- ***Example1: One student (S1,S2,S3,S4) is enrolled for many courses (C1,C2,C3,C4) and*** *one course is enrolled by* ***many students.***

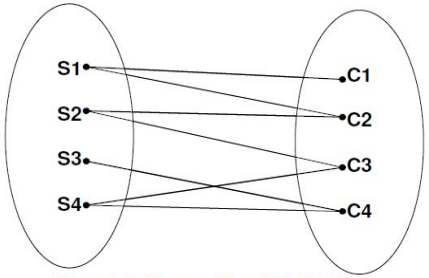


Figure 2-6: Many to Many Relationship

Example2: One student(S1,S2,S3,S4) trained by many instructors (I1,I2,I3,I4) and one *instructor trains* ***many students.***

Many to many relationship is superset of all the above mentioned relationships.

All other relationships are special case of **many to many relationships.**

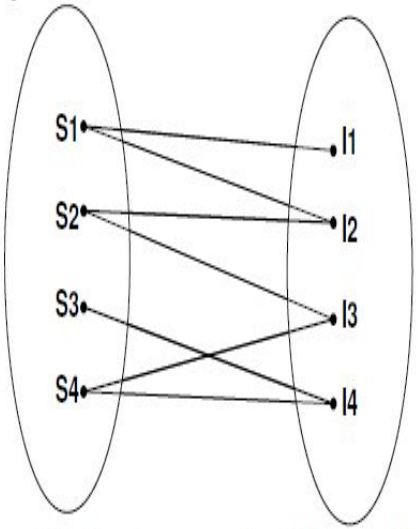
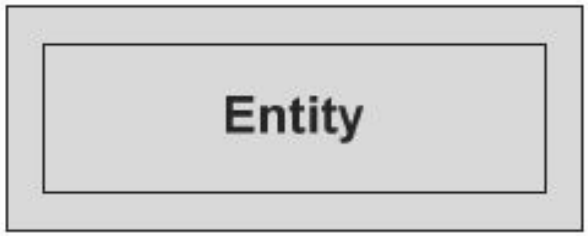


Figure 2-7: Another example of Many to Many Relationship

E-R Diagram Notations

Entity

An entity is an object or concept about which business user wants to store information.



Weak entity

A weak entity is dependent on another entity to exist .

Example bank branch depends upon bank name for its existence.

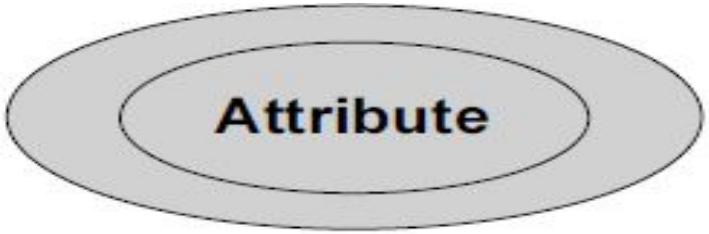
Without bank name it is impossible to identify bank branch uniquely.

Attributes

Attributes are the properties or characteristics of an entity.

Key attribute

A key attribute is the unique, distinguishing characteristic of the entity.
For example, an employee's employee number might be the employee's key
at tribute.



Multivalued attribute

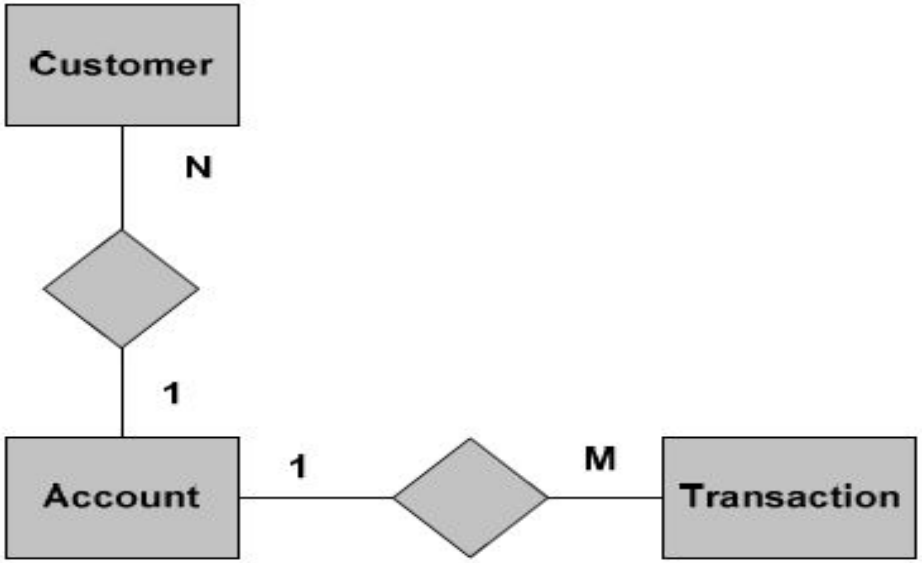
A multivalued attribute can have more than one value. For example, an employee entity can have multiple skill values.

Derived attribute

A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's basic salary and house rent allowance.

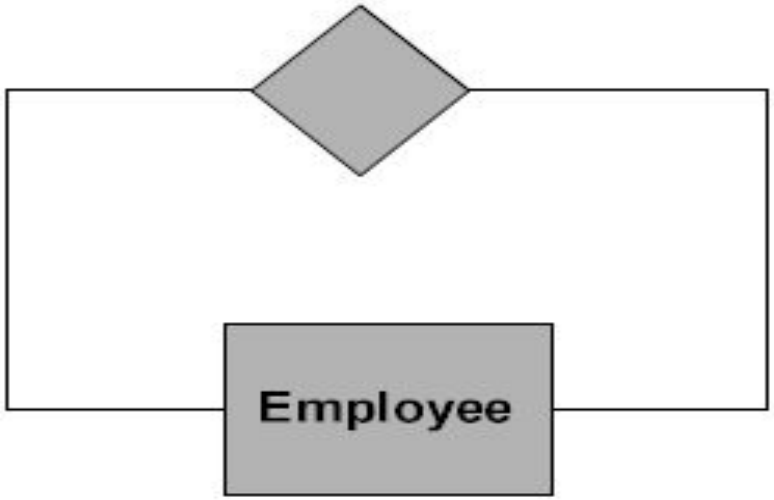
Relationships

Relationships illustrate how two entities share information in the database structure.



- **Cardinality**
- Cardinality Specifies how many instances of an entity relate to one instance of another entity.

- **M,N both represent 'MANY' and 1 represents 'ONE' cardinality**



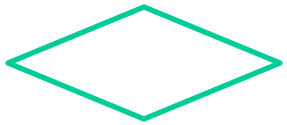
- **Recursive relationship**
- In some cases, entities can be self-linked.

- For example, Employees can supervise other employees

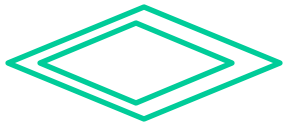
E_R Notations:

Entity

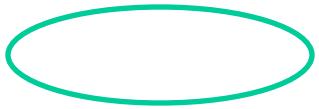
Weak Entity



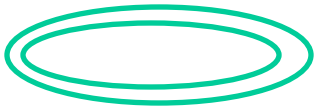
Relationship



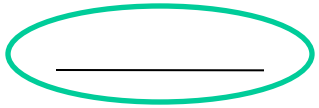
Identifying Relationship



Attribute

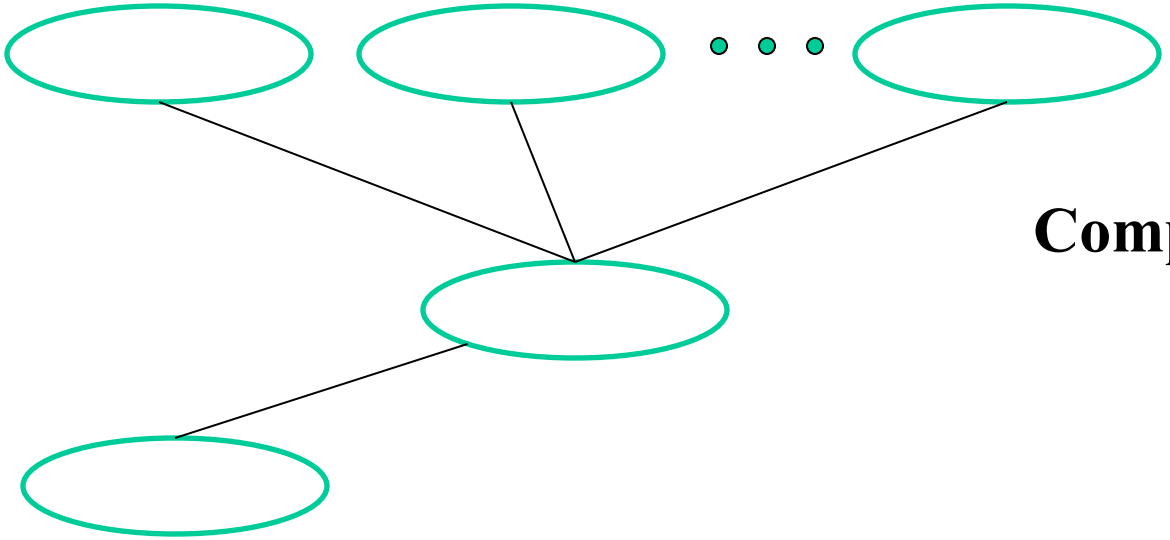


Multivalued Attribute



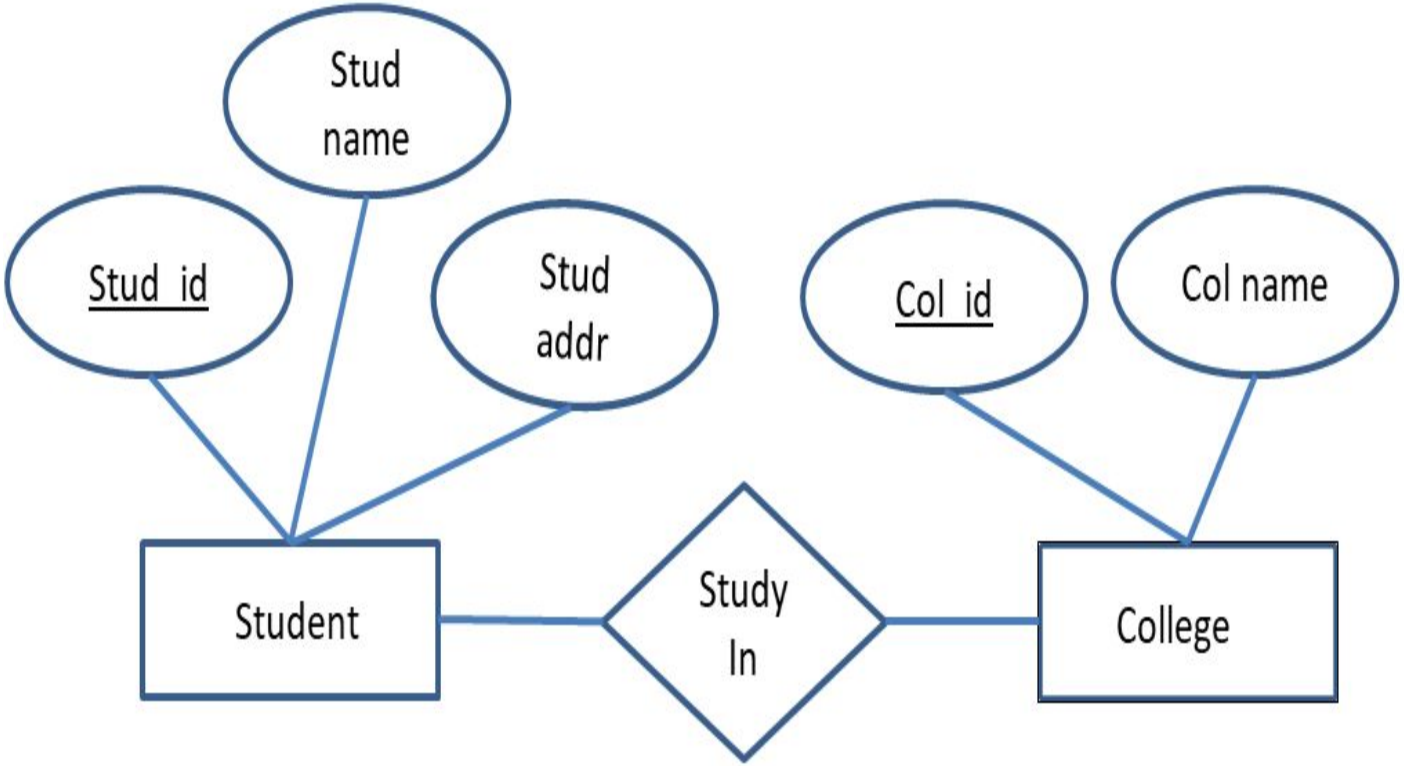
Key Attribute

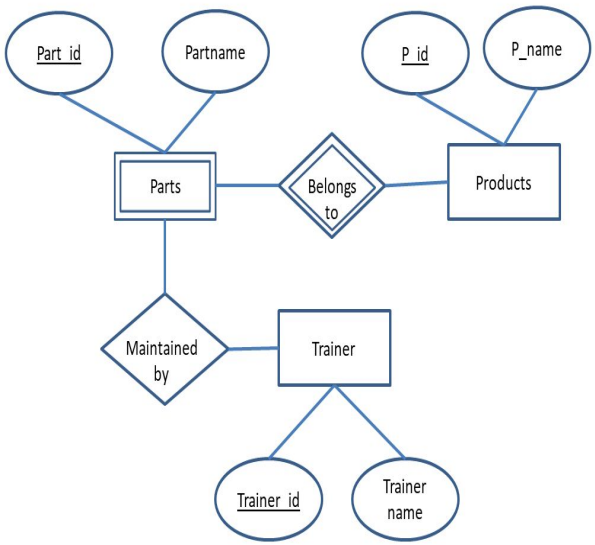
Derived Attribute

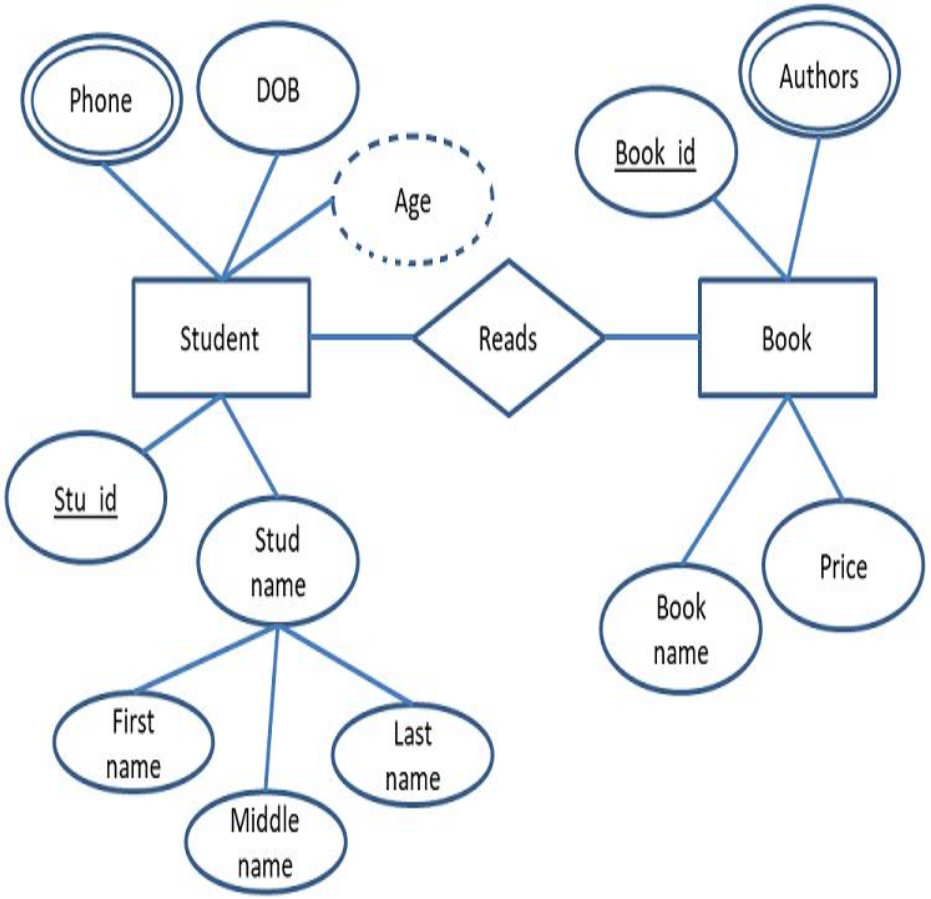


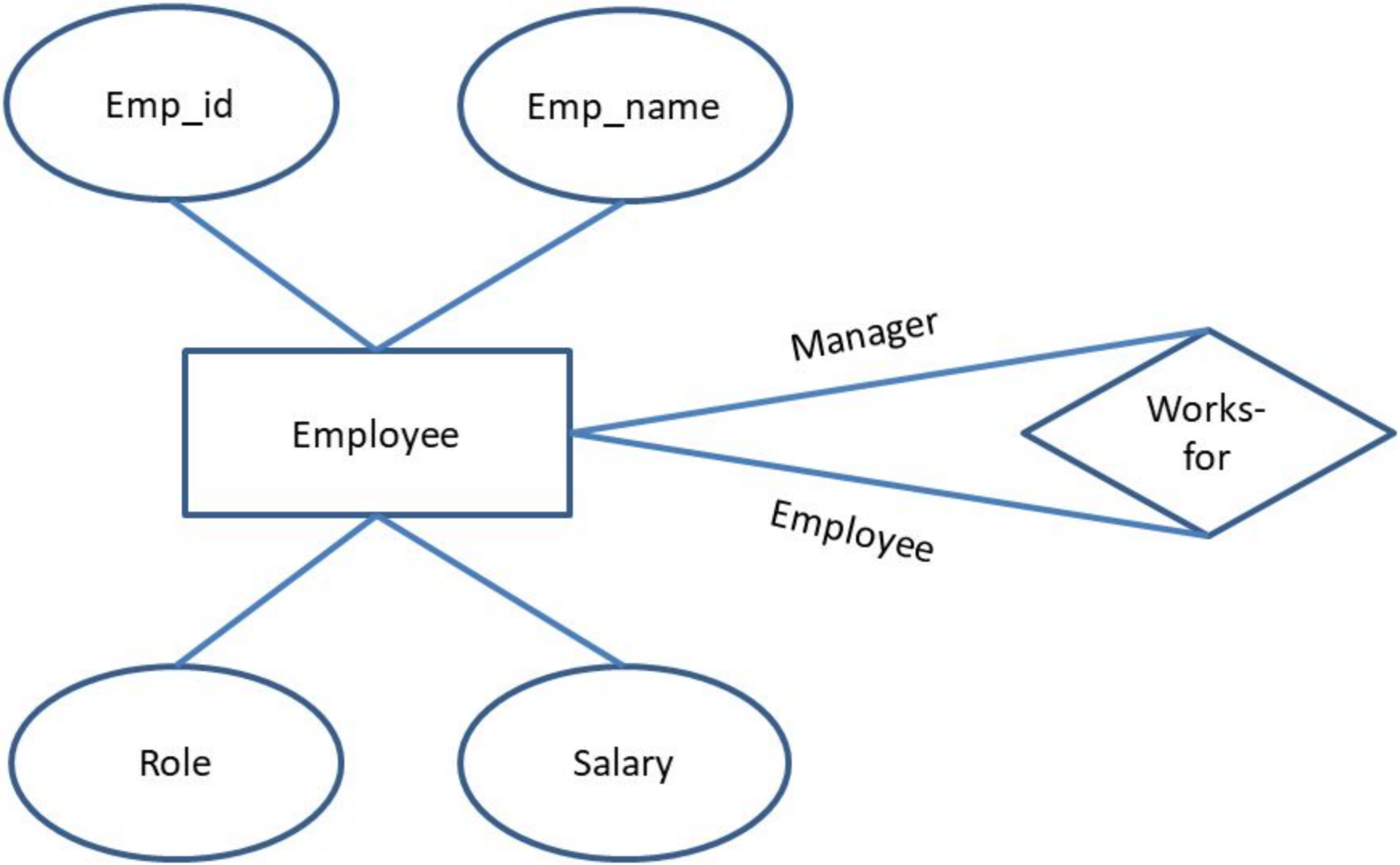
Composite Attribute

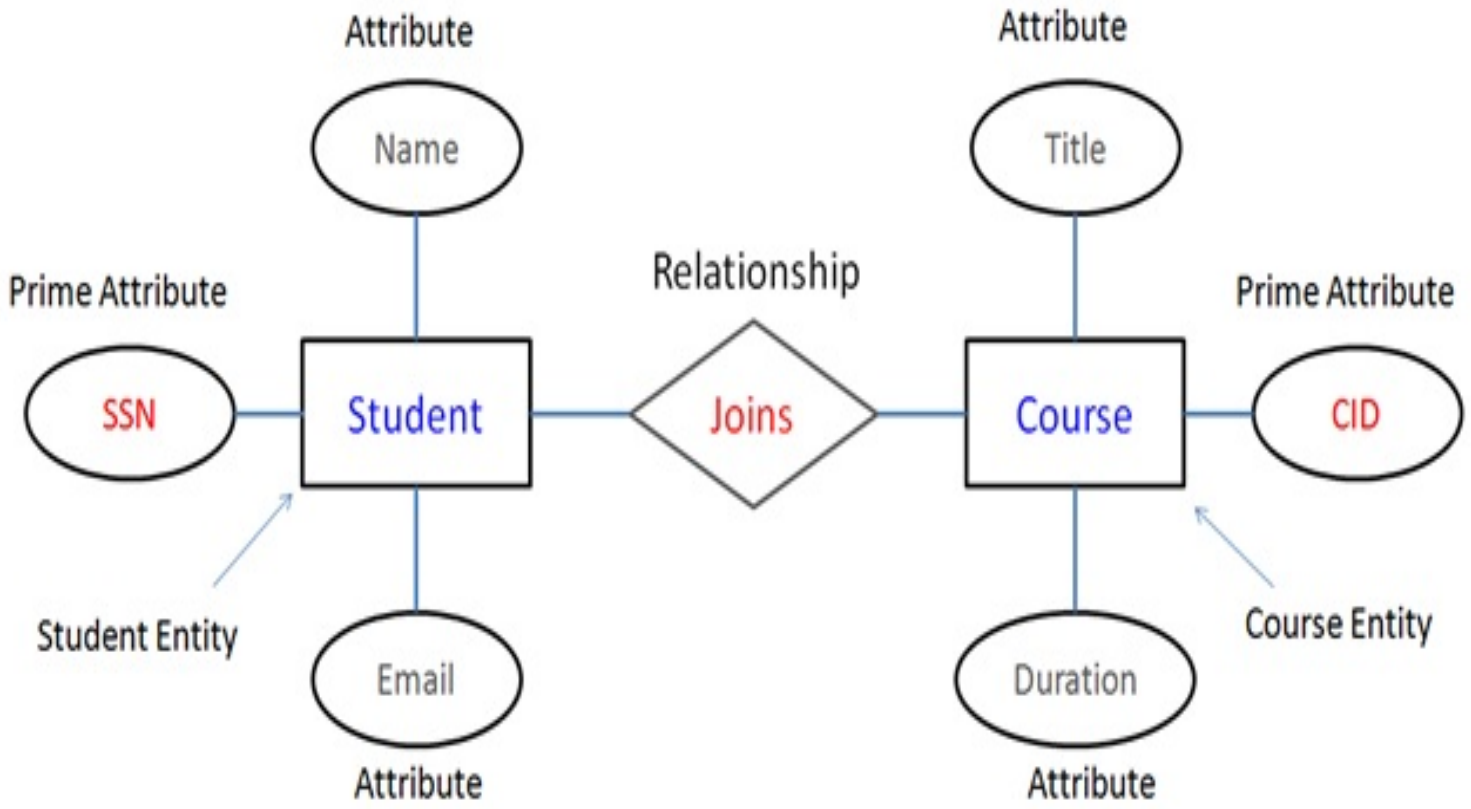
- **Rectangle:** entity sets
- **Ellipsis (oval):** attributes
- **Diamonds:** relationship sets
- **Lines:** they link attributes to entity sets and entities set to relationship sets
- **Double ellipsis:** multivalued attributes
- **Dashed ellipsis:** derived attributes
- **Double rectangle:** weak entity sets
- **Double lines:** total participation of an entity in a relationship sets




Part_id
Partname
P_id
P_name
Parts
Belongs to
Products
Maintained by
Trainer
Trainer_id
Trainer name


Phone
DOB
Age
Authors
Book_id
Student
Reads
Book
Stu_id
Stud name
Price
Book name
First name
Last name
Middle name


Emp_id
Emp_name
Employee
Manager
Works-
for
Employee
Role
Salary


Attribute
Name
Attribute
Title
Relationship
Prime Attribute
SSN
Student
Joins
Course
CID
Prime Attribute
Student Entity
Email
Duration
Course Entity
Attribute
Attribute

Few Terminologies

- **Schema** – the logical structure of the database (e.g., set of customers and accounts and the relationship between them).
- **Instance** – the actual content of the database at a particular point in time.
- **Degree** – the number of columns associated with a relation or table.
- **Cardinality** – the number of rows in a table.
- **Domain** – the range of possible values for an attribute.
  e.g. – the domain for the attribute gender may be {'M', 'F'}
  the domain for the attribute title may be { 'Mr.', "Ms.', 'Mrs.', 'Dr.'}

Types of Atrributes

- **Simple attribute** – attribute that consist of a single atomic value.
  e.g. – Roll, Marks, Salary, etc.
- **Composite Attribute** – attributes which can be decomposed further
  e.g. – Name (First Name, Middle Name, Last Name)
  Address (House No., Street, City, State)
- **Single-valued atribute** – attribute that hold a single value
  e.g. – City, EmpId, Salary, etc.
- **Multi-valued attribute** – attribute that hold multiple values
  e.g. – Phone no., EmailID
- **Derived attribute** – an attribute that can be calculated or derived from another attribute
  e.g. – Age (derived from DOB)
- **Null attribute** – an attribute whose value is missing for a record.
  e.g. - if a record does not contain an assignment for the Price attribute

Steps in E-R Modeling

a) **Identify the entities**: Look for general nouns in requirement specification document which are of business interest to business users.

b) **Find relationships:** Identify the natural relationship and their cardinalities between the entities.

c) **Identify the key attributes for every entity:** Identify the attribute or set of attributes which can identify instance of entity uniquely.

d) **Identify other relevant attributes:** Identify other attributes which are interest to business users and want to store the information in database

e) **Complete E-R diagram:** Draw complete E-R diagram with all attributes including primary key

f) **Review your results with your business users:** Look at the list of attributes associated with each entity to see if anything has been omit ted.

Case Study : Problem Statement

- **University database application.**
- An university has many departments
- Each department has multiple instructors;
- one among them is the head of the department
- An instructor belongs to only one department
- Each department offers multiple courses, each of which is taught by a single instructor
- A student may enrol for many courses offered by different departments

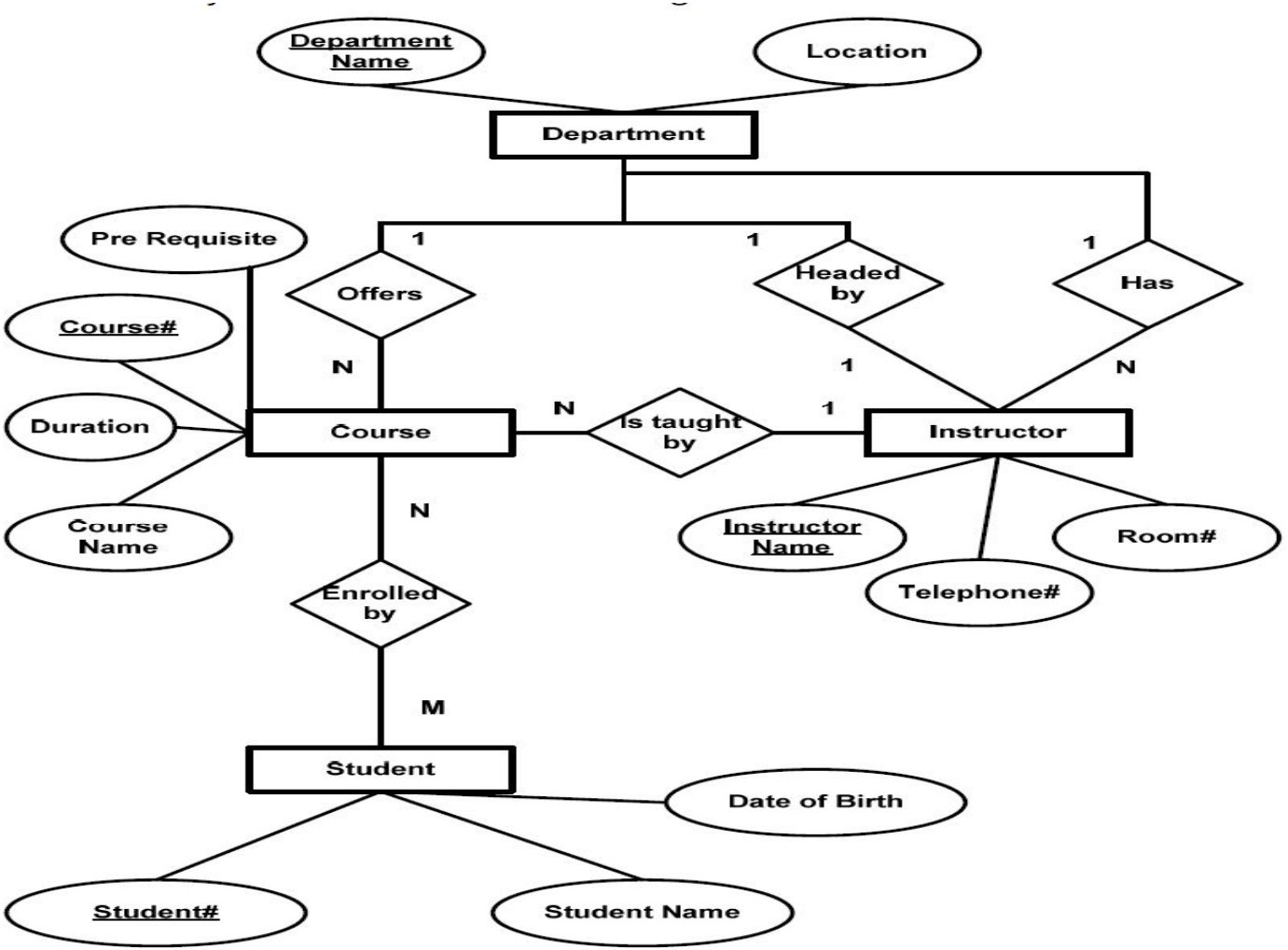


Figure 2-9: E-R Diagram for University

Exercise 2.5 Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $2500/day).

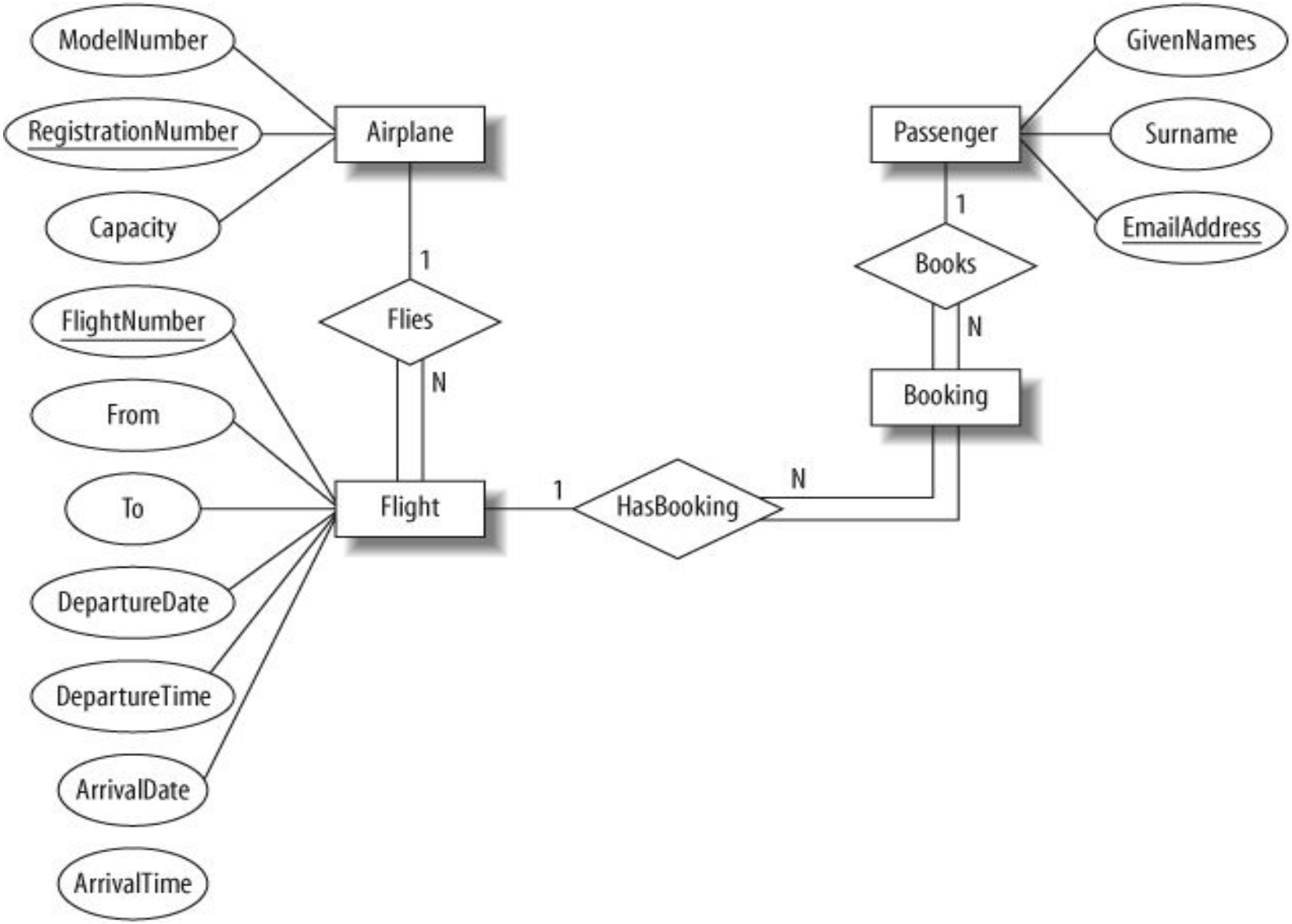
- Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
- Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
- Each song recorded at Notown has a title and an author.

- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Design a conceptual schema for Notown and draw an ER diagram for your schema. The preceding information describes the situation that the Notown database must model. Be sure to indicate all key and cardinality constraints and any assumptions you make. Identify any constraints you are unable to capture in the ER diagram and briefly explain why you could not express them.

- The Flight Database

- The airline has one or more airplanes.
- An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- Each flight is carried out by a single airplane.
- A passenger has given names, a surname, and a unique email address.
- A passenger can book a seat on a flight.


ModelNumber
RegistrationNumber
Capacity
Airplane
1
Flies
N
FlightNumber
From
To
DepartureDate
DepartureTime
ArrivalDate
ArrivalTime
Flight
1
HasBooking
N
Booking
N
Books
1
Passenger
GivenNames
Surname
EmailAddress

- An Airplane is uniquely identified by its RegistrationNumber, so we use this as the primary key.
- A Flight is uniquely identified by its FlightNumber, so we use the flight number as the primary key. The departure and destination airports are captured in the From and To attributes, and we have separate attributes for the departure and arrival date and time.
- Because no two passengers will share an email address, we can use the EmailAddress as the primary key for the Passenger entity.
- An airplane can be involved in any number of flights, while each flight uses exactly one airplane, so the Flies relationship between the Airplane and Flight relationships has cardinality 1:N; because a flight cannot exist without an airplane, the Flight entity participates totally in this relationship.
- A passenger can book any number of flights, while a flight can be booked by any number of passengers. As discussed earlier in Intermediate Entities,"
- We could specify an M:N Books relationship between the Passenger and Flight relationship, but considering the issue more carefully shows that there is a hidden entity here: the booking itself.
- We capture this by creating the intermediate entity Booking and 1:N relationships between it and the Passenger and Flight entities. Identifying such entities allows us to get a better picture of the requirements.

Case Study : Problem Statement

- Let us consider a university library scenario for developing the E-R model.

Assume in a university

- There are multiple libraries and each library has multiple student members
- Students can become members to multiple libraries by paying appropriate membership fee
- Each library has its own set of books. Within the library these books are identified by a unique number
- Students can borrow multiple books from subscribed library
- Students can order books using inter-library loan.
- This can be useful if a student wishes to borrow books from a library where s/ he is not a member.
- The student orders the books through a library where s/ he is a member.

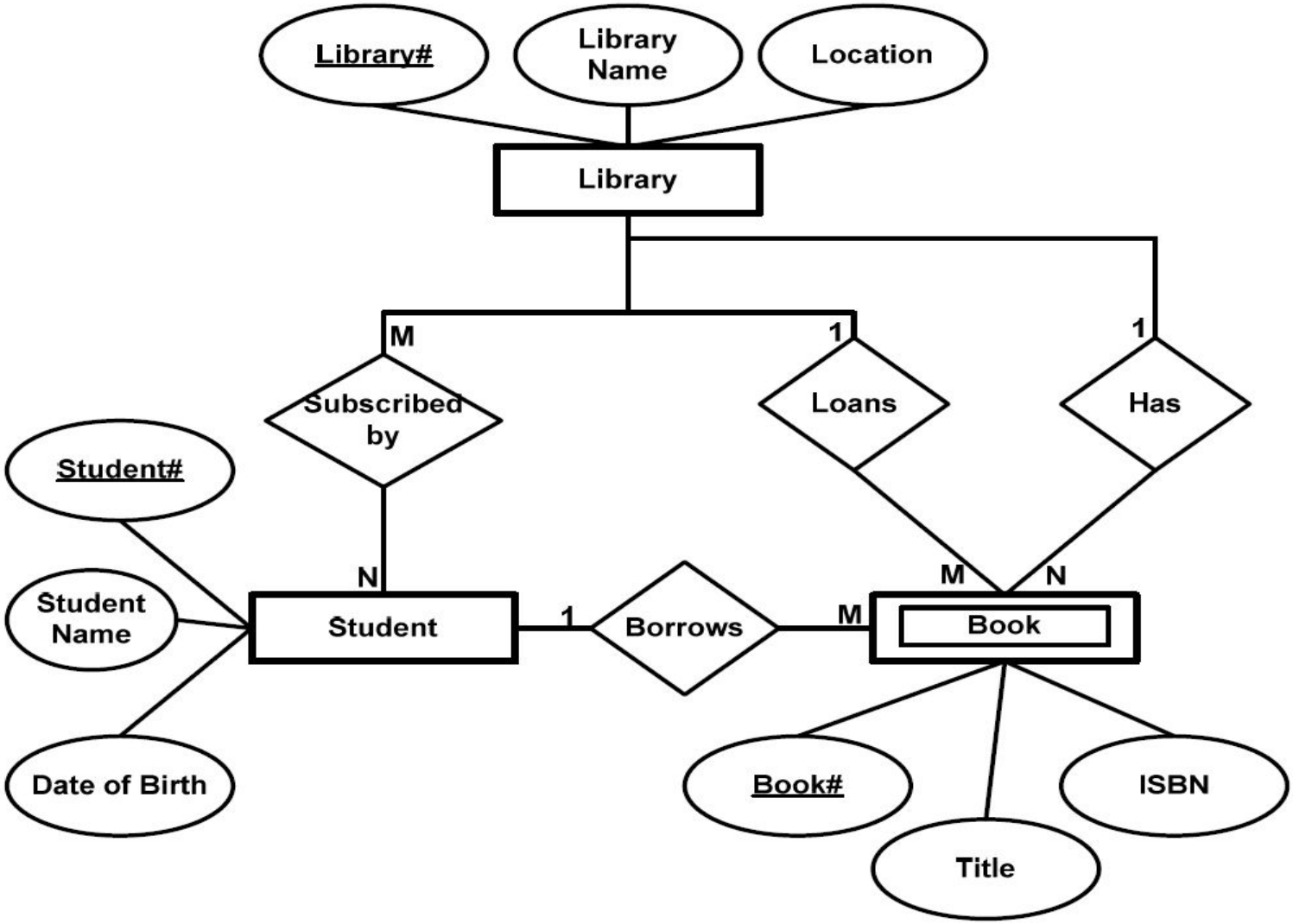


Figure 2-10: E-R Diagram for University Library

Case Study : Problem Statement

- Let us consider a banking business scenario for developing the ER model.
- **Assume in a city**
- There are multiple banks and each bank has many branches. Each branch has multiple customers
- Customers have various types of accounts
- Some customers also had taken different types of loans from these bank Branches
- One customer can have multiple accounts and loans

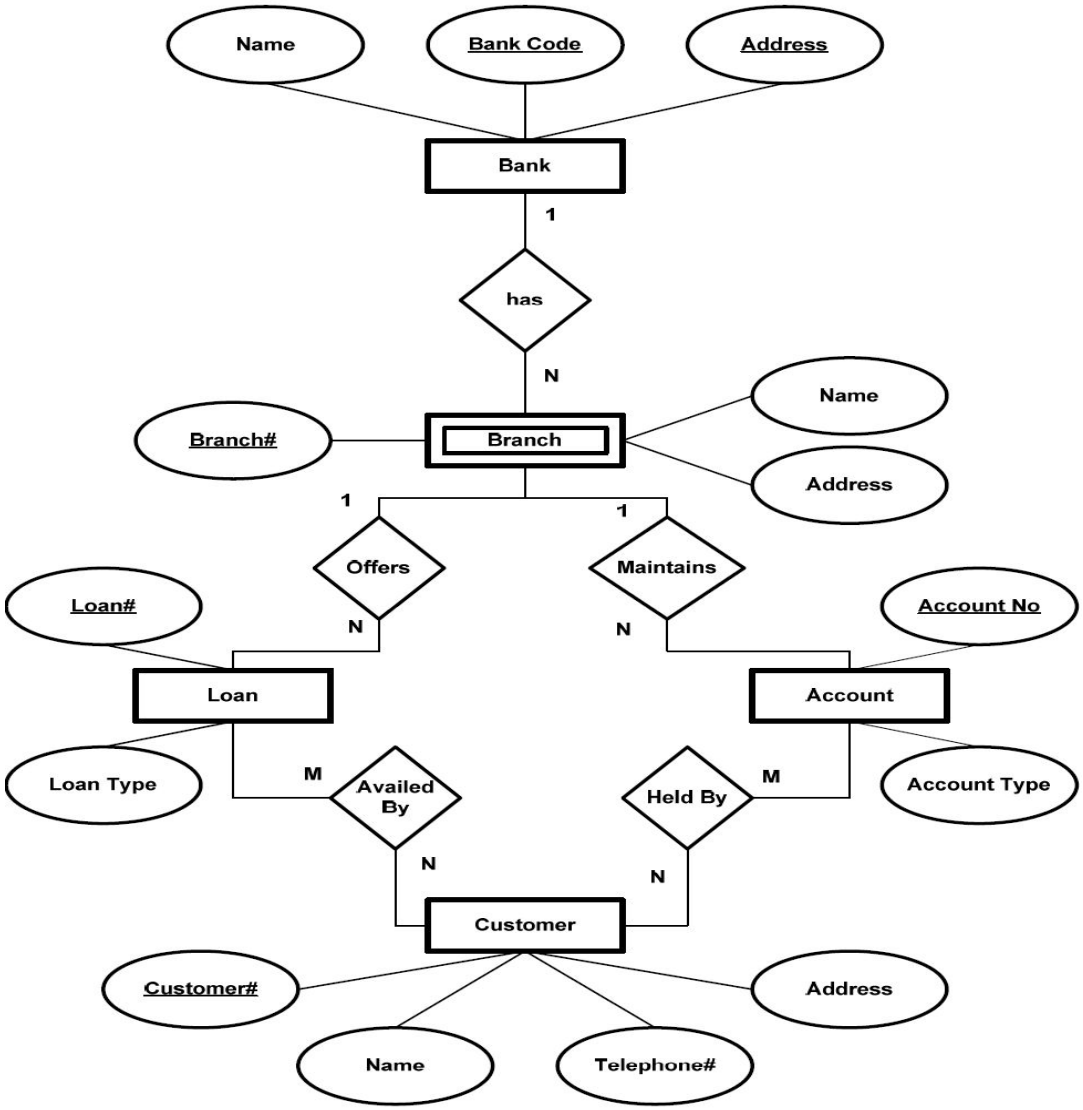


Figure 2-11: E-R Diagram for Bank

Merits and Demerits of E-R Modeling

Merits of E-R Modeling

1. Easy to understand. Represented in business users language. Can be understood by non-technical specialist .
2. *Intuitive 32and helps in physical database creation.*
3. Can be generalized and specialized based on needs.
4. Can help in database design.
5. Gives a higher level abstract ion of the system.

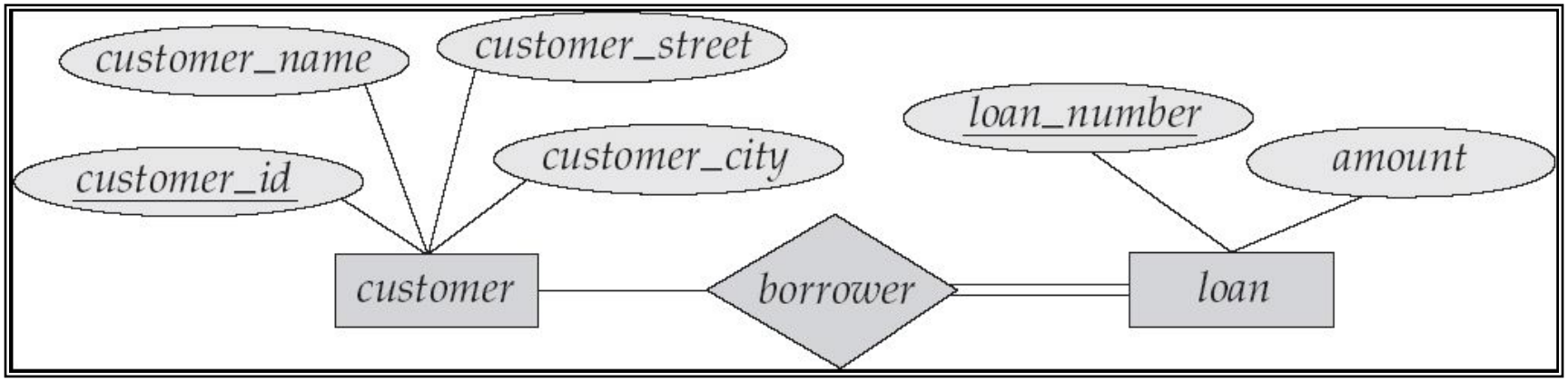
Demerits of E-R Modeling

1. Physical design derived from E-R Model may have some amount of ambiguities or inconsistency.
2. Sometime diagrams may lead to misinterpretations.

Example:

Participation of an Entity Set in a Relationship Set

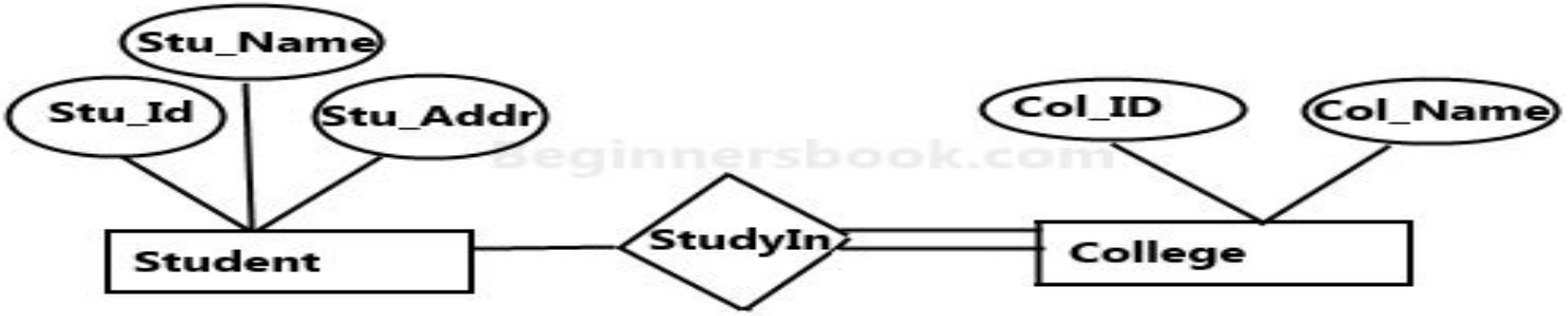
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g. participation of loan in borrower is total
    - every loan must have a customer associated to it via borrower
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of customer in borrower is partial



Total Participation of an Entity set

A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set.

For example: In the below diagram each college must have at-least one associated Student.



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

Strong and Weak Entity Set

- **Strong entity set** – An entity set that has a fixed entity set.

- **Weak entity set** – An entity set that does not have sufficient attributes to form a primary key.

  e.g. –

  loan = (loan_id, loan_name, borrower, amount)

  payment = (payment_id, payment_date, amount)

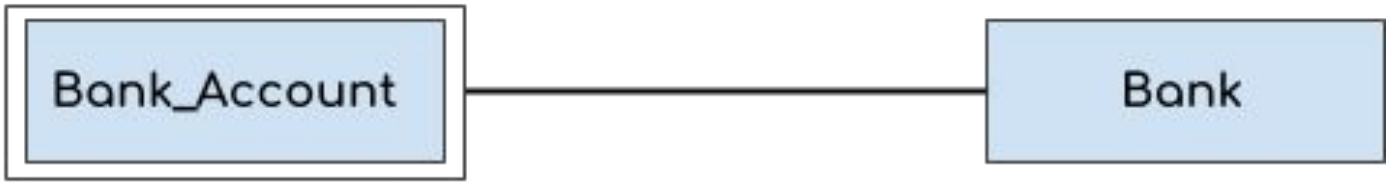
  The payment relation does not have a primary key (payment_id cannot uniquely identify tuples) and hence is a weak entity set.

- **Discriminator/Partial Key** – An attribute of a weak entity set which normally does not show uniqueness, but shows uniqueness when combined with a strong Entity set.

  e.g. – payment_id when combined with loan_id, can uniquely identify tuples.

Weak Entity:

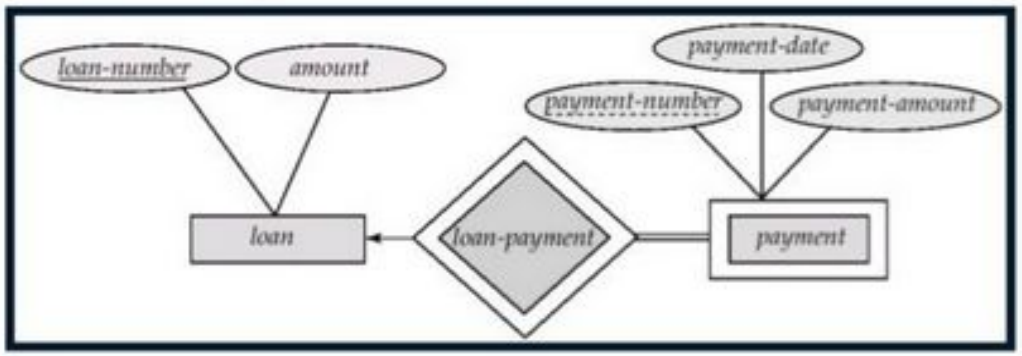
- An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity.
- The weak entity is represented by a double rectangle.
- For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



Weak Entity Set

- Some entity sets in real world naturally depend on some other entity set.
- They can be uniquely identified only if combined with another entity set

  - **We depict a weak entity set by double rectangles.**
  - **We underline the discriminator of a weak entity set with a dashed line.**
  - ***payment-number* – discriminator of the *payment* entity set**
  - **Primary key for *payment* – (*loan-number, payment-number*)**



❖ Double rectangles for weak entity set

❖ Double diamond for weak entity relationship

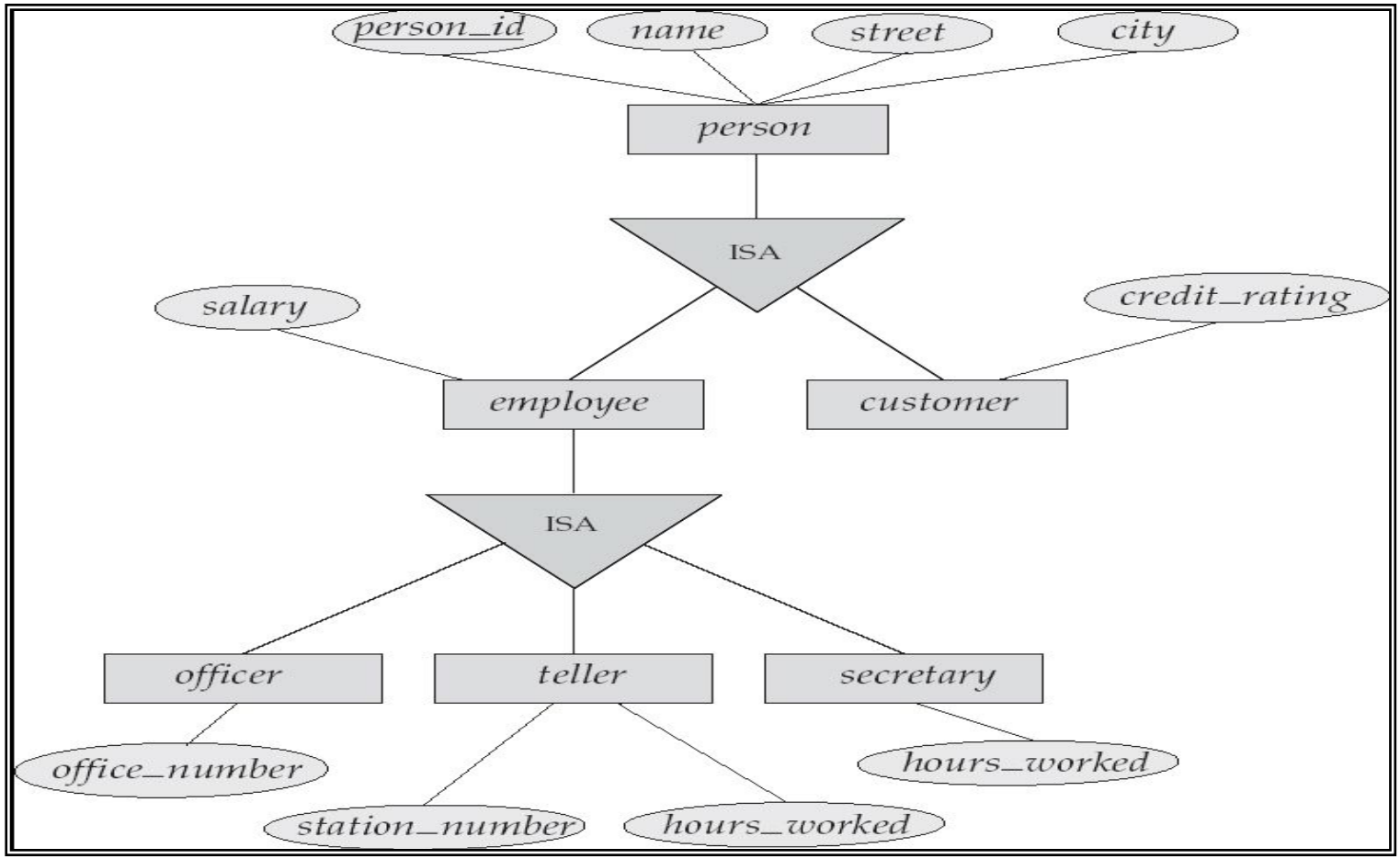
❖ Dashed underscore for discriminator

Specialization:

- An entity set may include subgroups of entities within it which are distinct in some way from other entities.
- The process of designating sub groupings within an entity set is called Specialization.
- **Top-down design process;** we designate sub groupings within an entity set that are distinctive from other entities in the set.
- These sub groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g. *customer* "is a" *person*).

- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

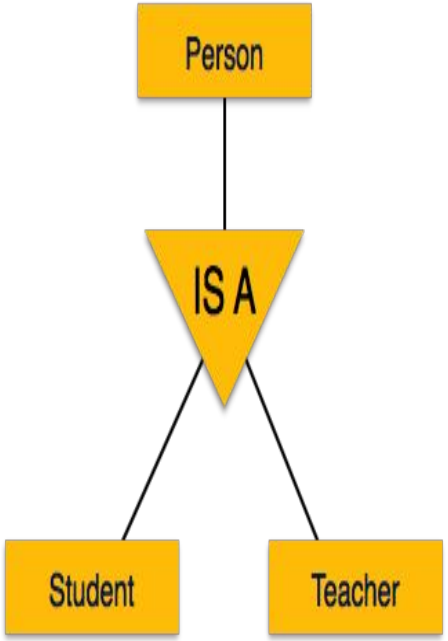
Specialization Example



Specialization

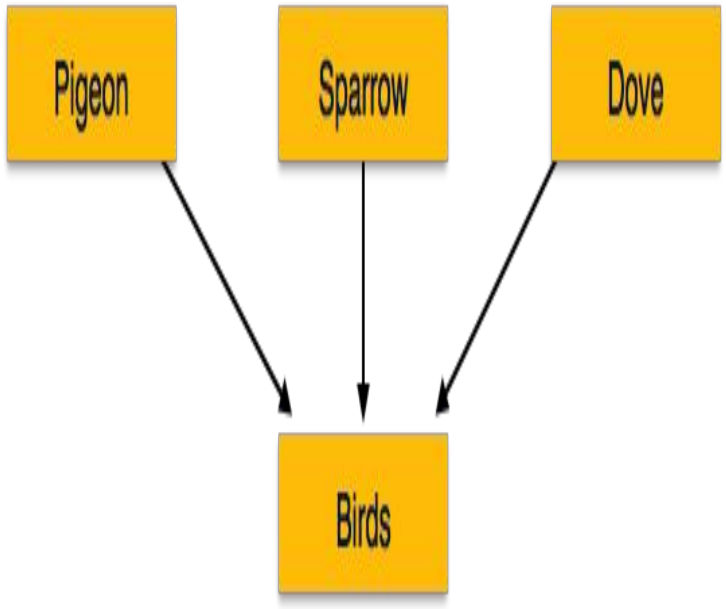
Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.



- **Generalization**
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.
- **Generalization and Specialization**
- Can have multiple specializations of an entity set based on different features.
- E.g. *permanent_employee* vs. *temporary_employee*, in addition to *officer* vs. *secretary* vs. *teller*
- Each particular employee would be
  - a member of one of *permanent_employee* or *temporary_employee*,
  - and also a member of one of *officer*, *secretary*, or *teller*
- The ISA relationship also referred to as **superclass - subclass** relationship

- **Generalization**
- As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



What is Generalisation

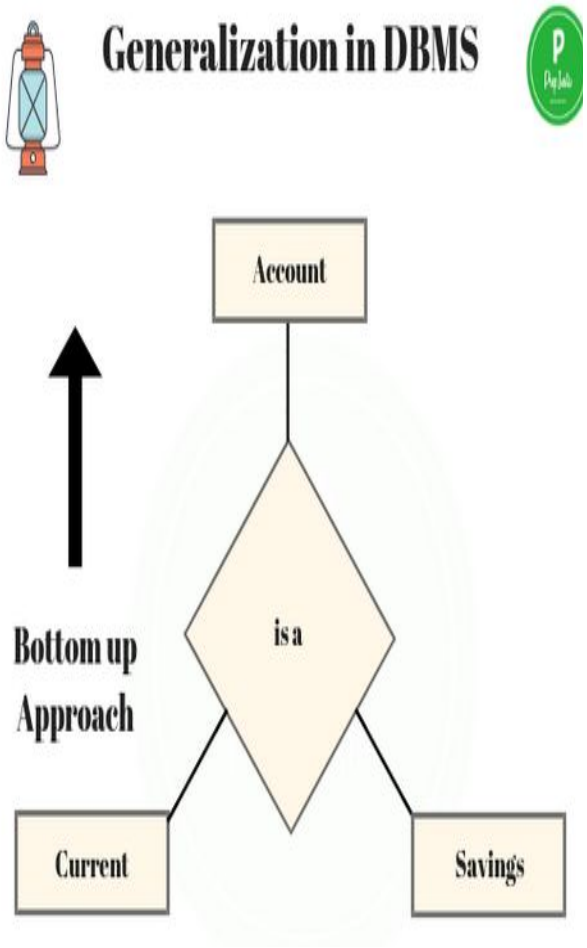
In generalisation we combine lower level entities to form a higher level entity. Thus its clear that it follows a bottom up approach.

Example –

In a bank there are two different types of accounts – Current and Savings, combine to form a super entity Account.

It thus follows system like classes, like superclasses and subclasses right?

It may also be possible that the higher level entity may also combine with further entity to form a one more higher level entity.

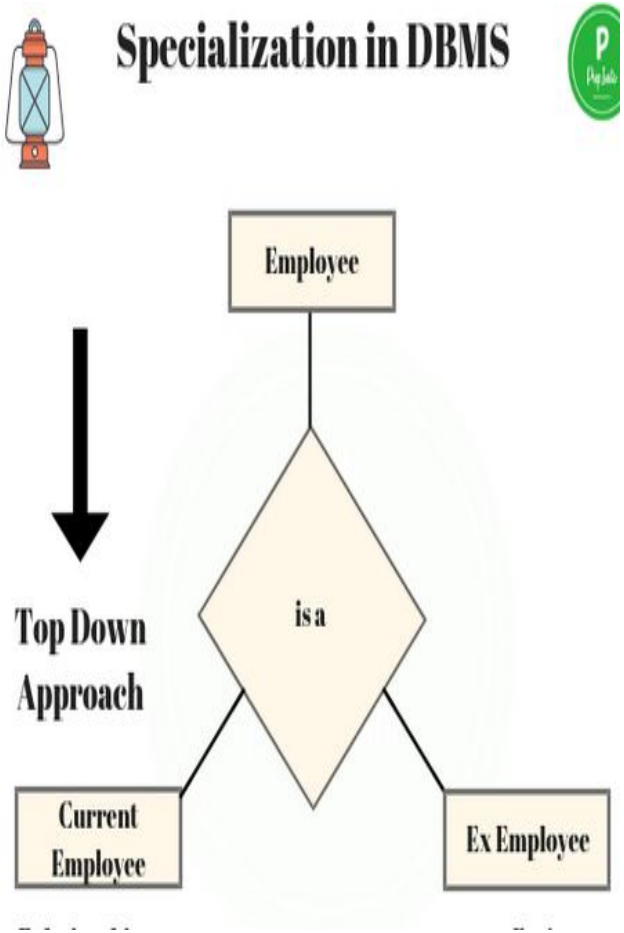


What is Specialisation

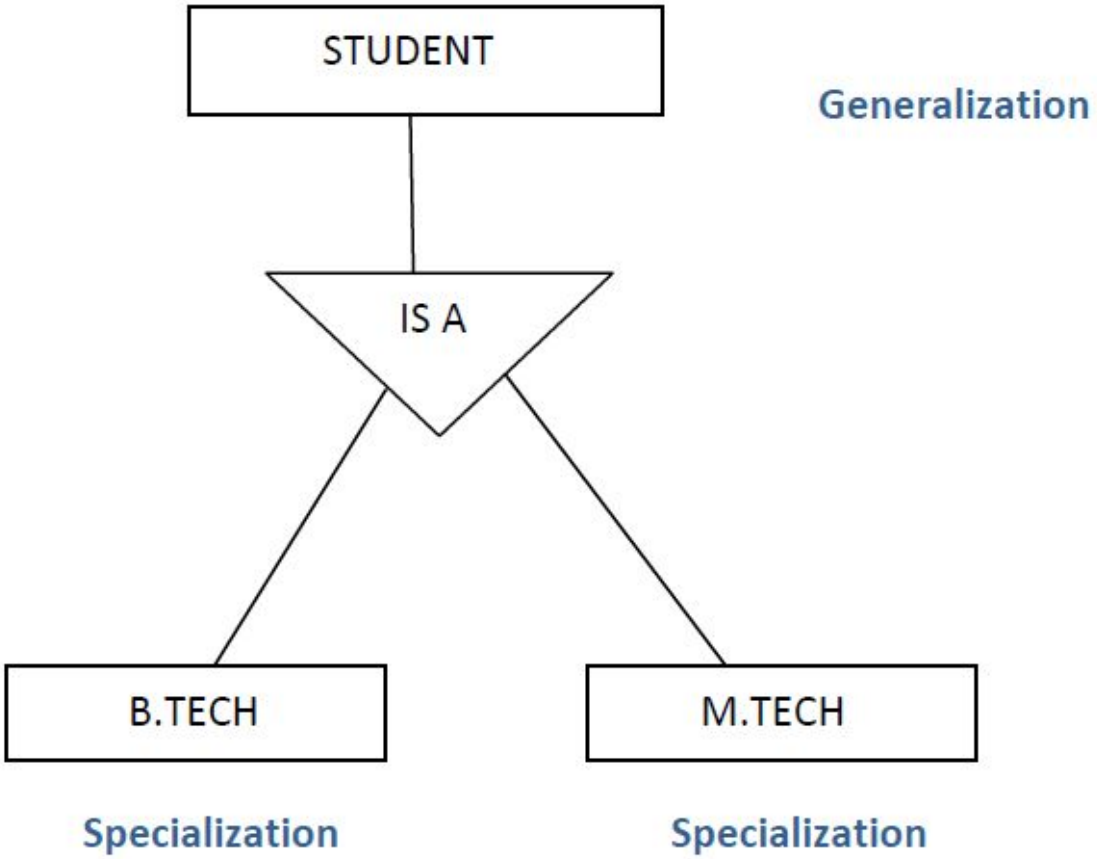
While generalisation may follow a bottom up approach. Specialisation is opposite to that, it follows a top down approach rather.

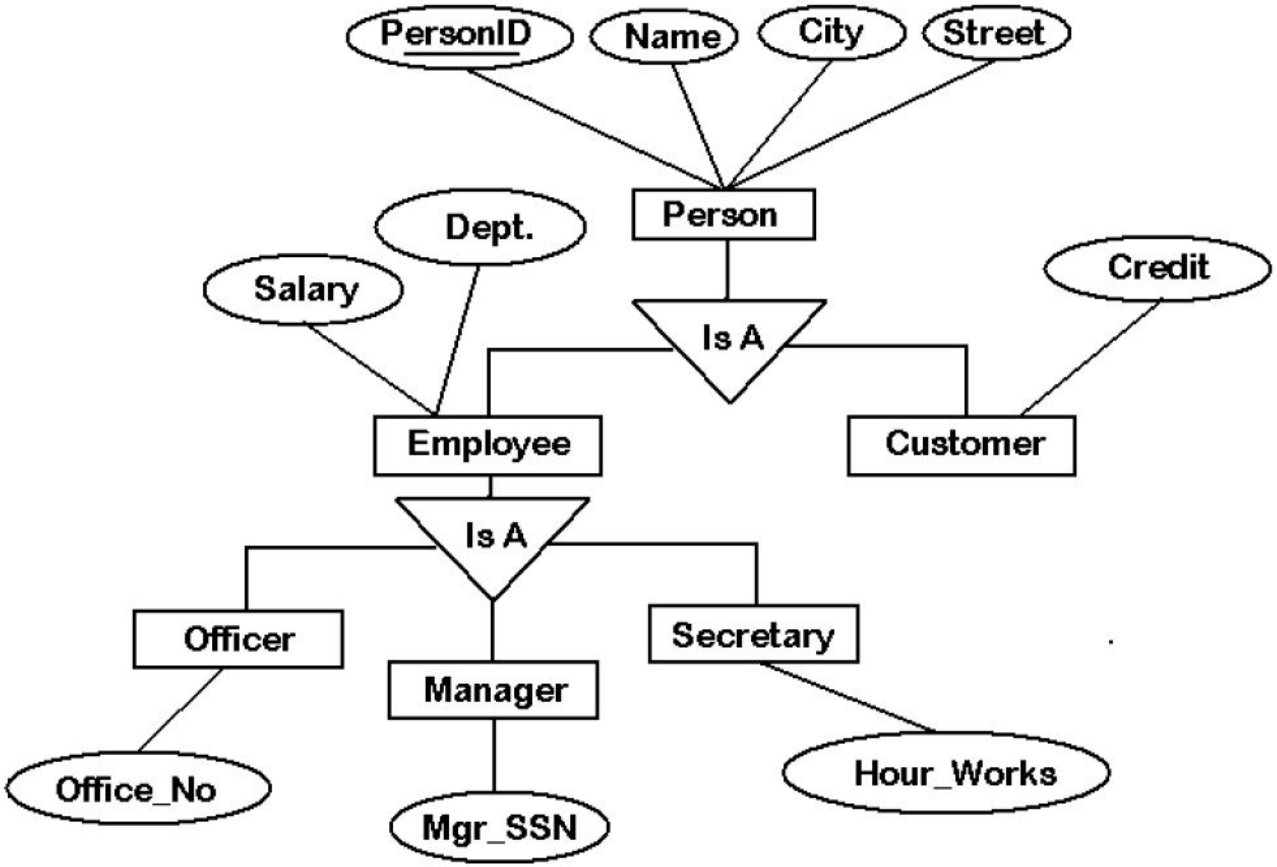
For example –

Employee may be decomposed to further as current employee entity and ex employee entity.



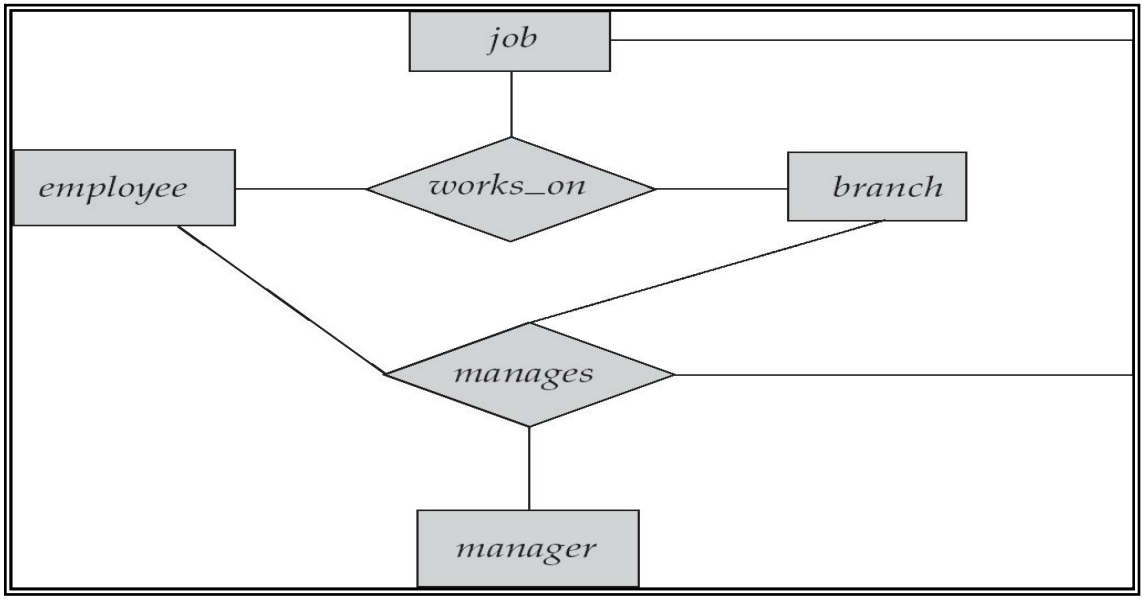
Ex:


STUDENT
Generalization
IS A
B.TECH
M.TECH
Specialization
Specialization


PersonID
Name
City
Street
Person
Dept.
Salary
Credit
Is A
Employee
Customer
Is A
Officer
Manager
Secretary
Office_No
Mgr_SSN
Hour_Works

Aggregation

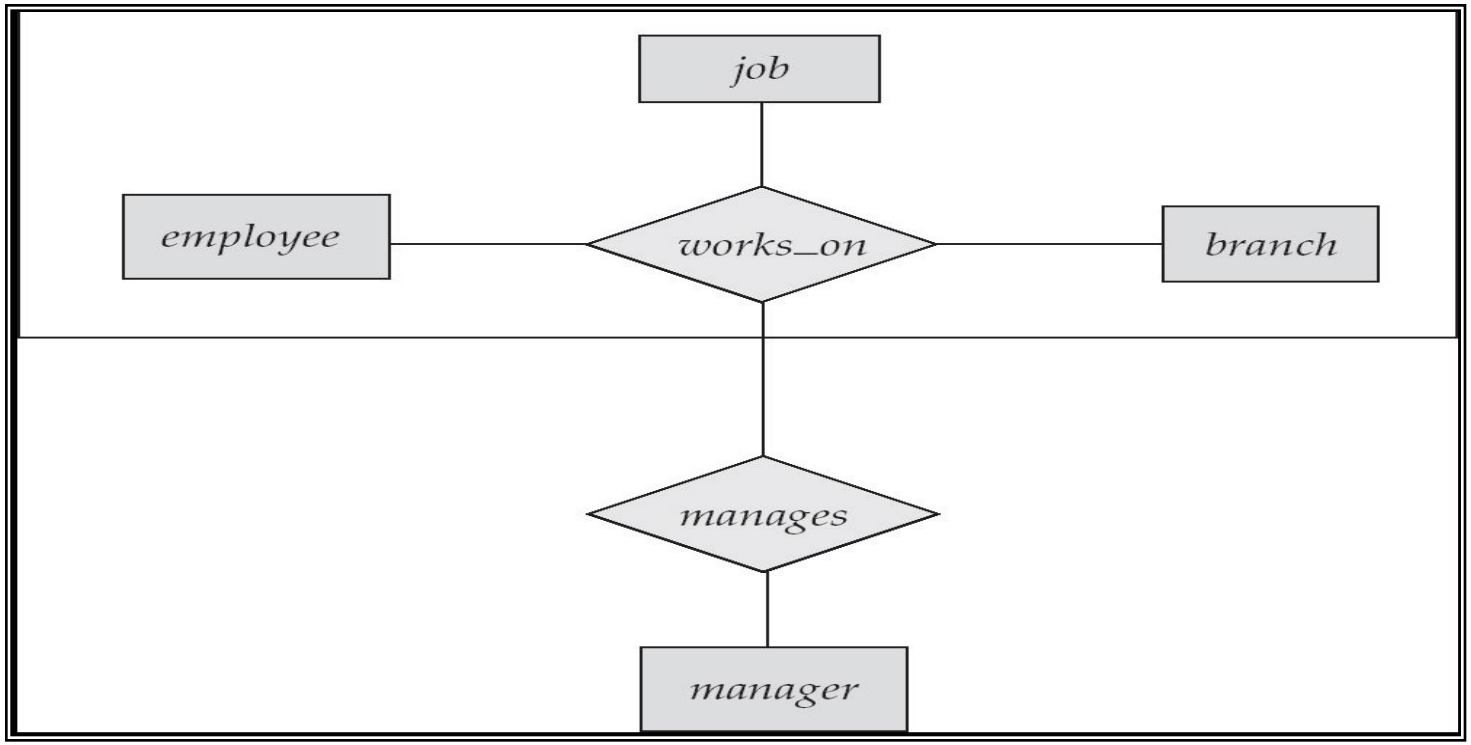
- Its an abstraction through which relationships are treated as higher level entities.
- Consider the ternary relationship *works_on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



Aggregation (Cont.)

- Relationship sets *works_on* and *manages* represent overlapping information
  - Every *manages* relationship corresponds to a *works_on* relationship
  - However, some *works_on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works_on* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity
- Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch
  - An employee, branch, job combination may have an associated manager

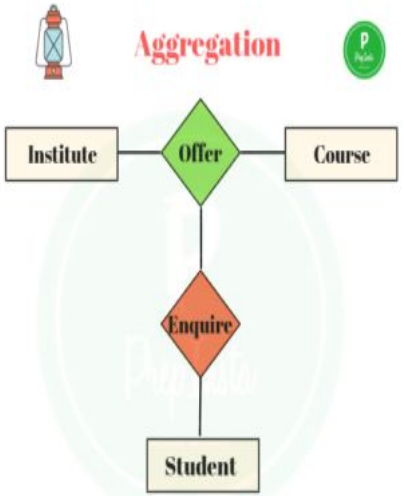
E-R Diagram With Aggregation



Aggregation in DBMS

- `Aggregation` is a design strategy in which the relationship is modeled between a collection of entities and another relationship
- Simply it is used when ***we need Express a relationship among other relationships***

Definition: Aggregation is a design process in which the relationship between two entities is treated as a single entity



Example:

In Real world situation for example if students visit a coaching institute then he shows interest not only to inquire about the course alone or not only just coaching center, he will definitely enquire the details about both the coaching institute and the details of the concerned course

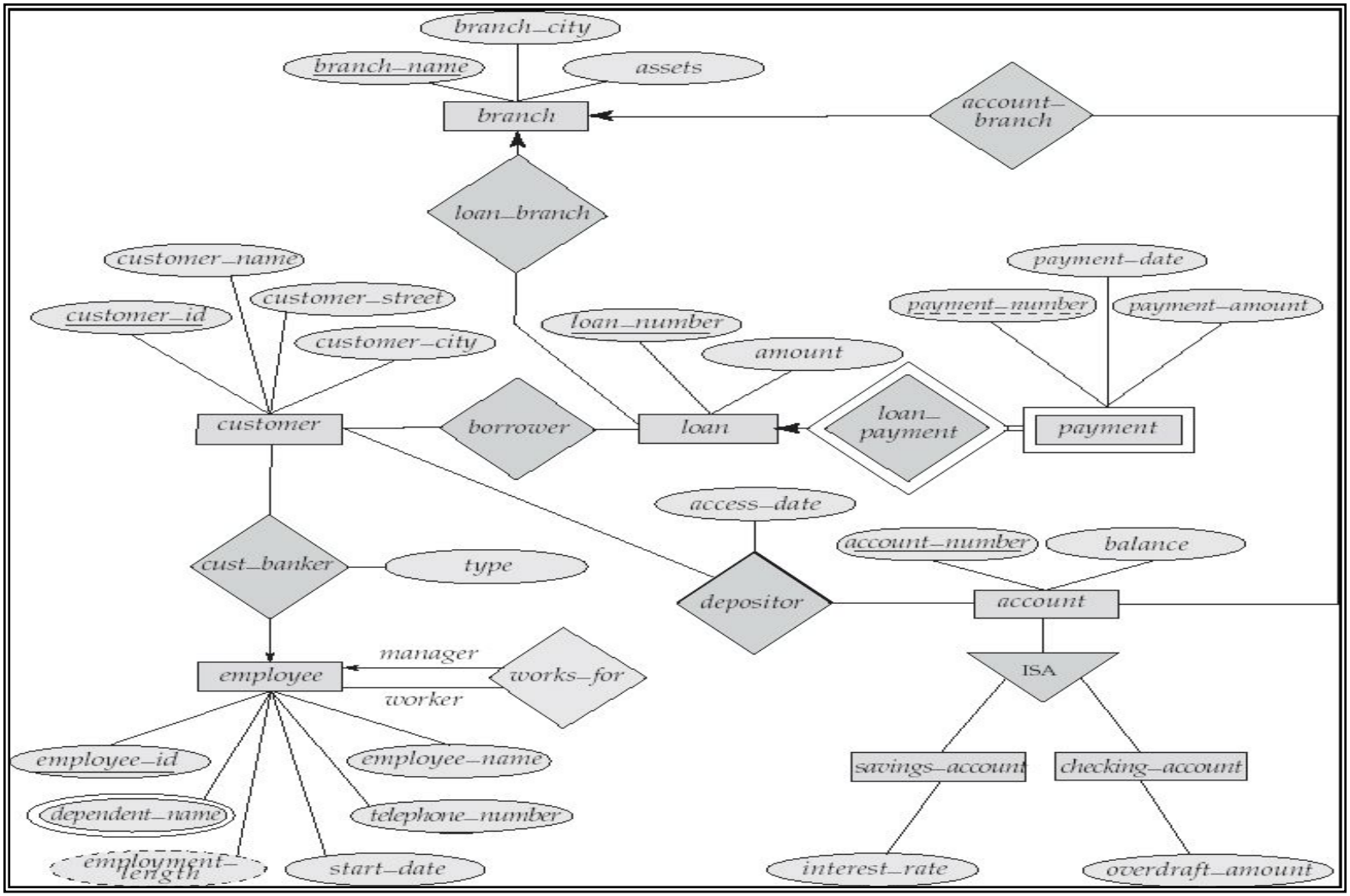
Design steps for aggregation

- Define entities and their attributes
- Add a relationship between these entities
- Define another entity so that the relationship can be established between the existing relationship and this entity

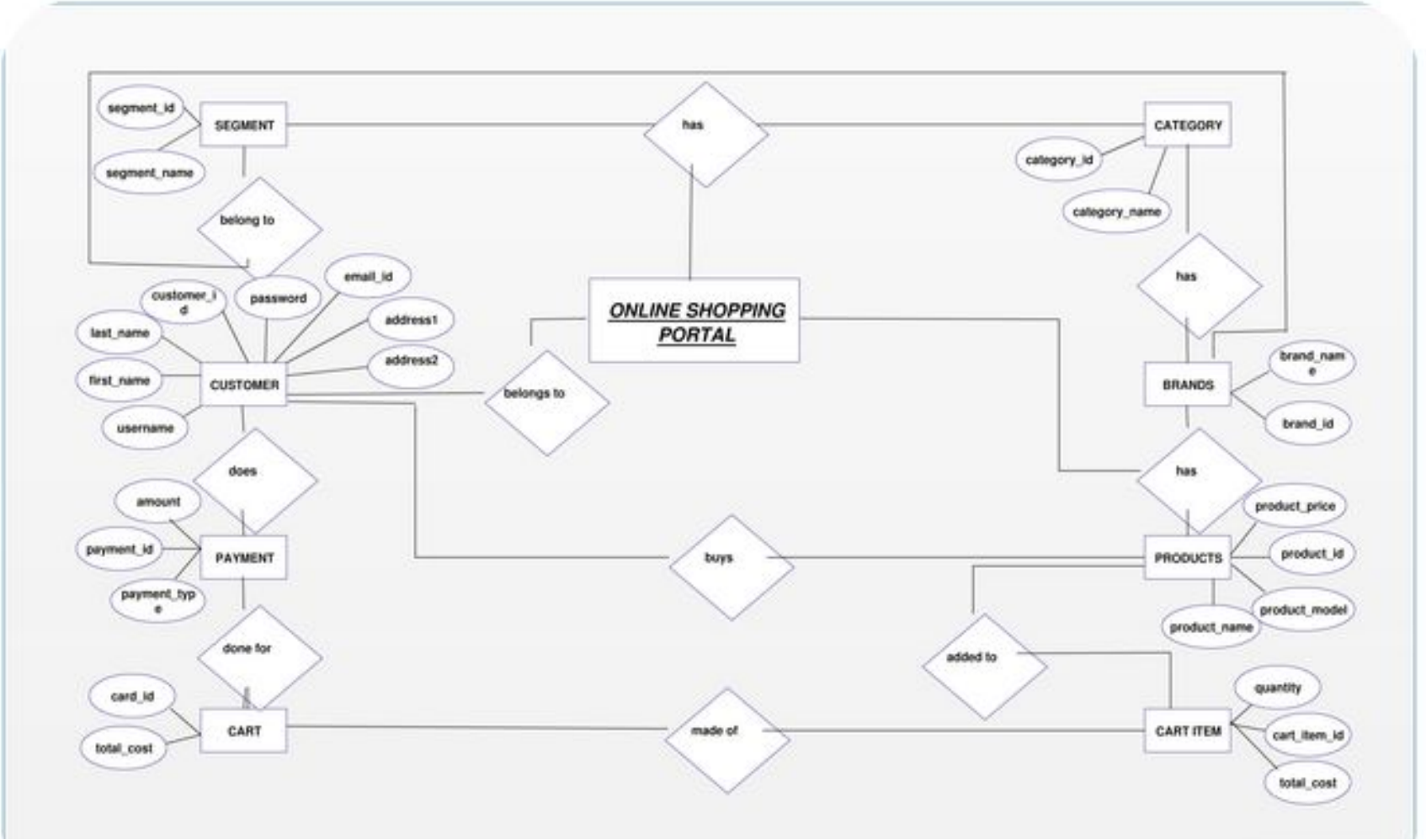
Why aggregation

- Aggregation is a process of compiling information on an object thereby abstracting higher-level object
- In SQL we need to find the sum of salaries of all the employees working in an organization or to find the highest paid employee from all branches of the organization etc

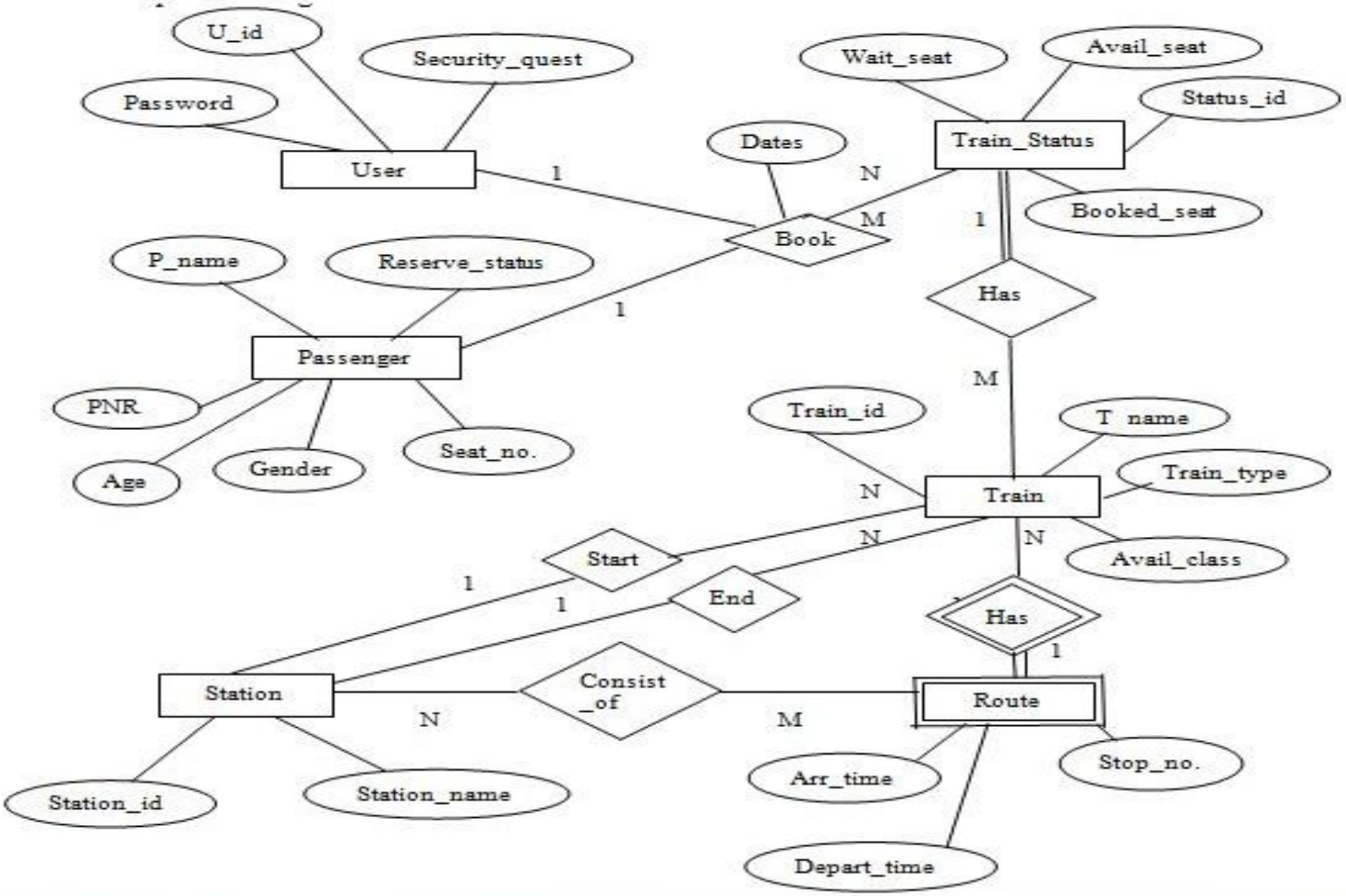
E-R Diagram for a Banking Enterprise



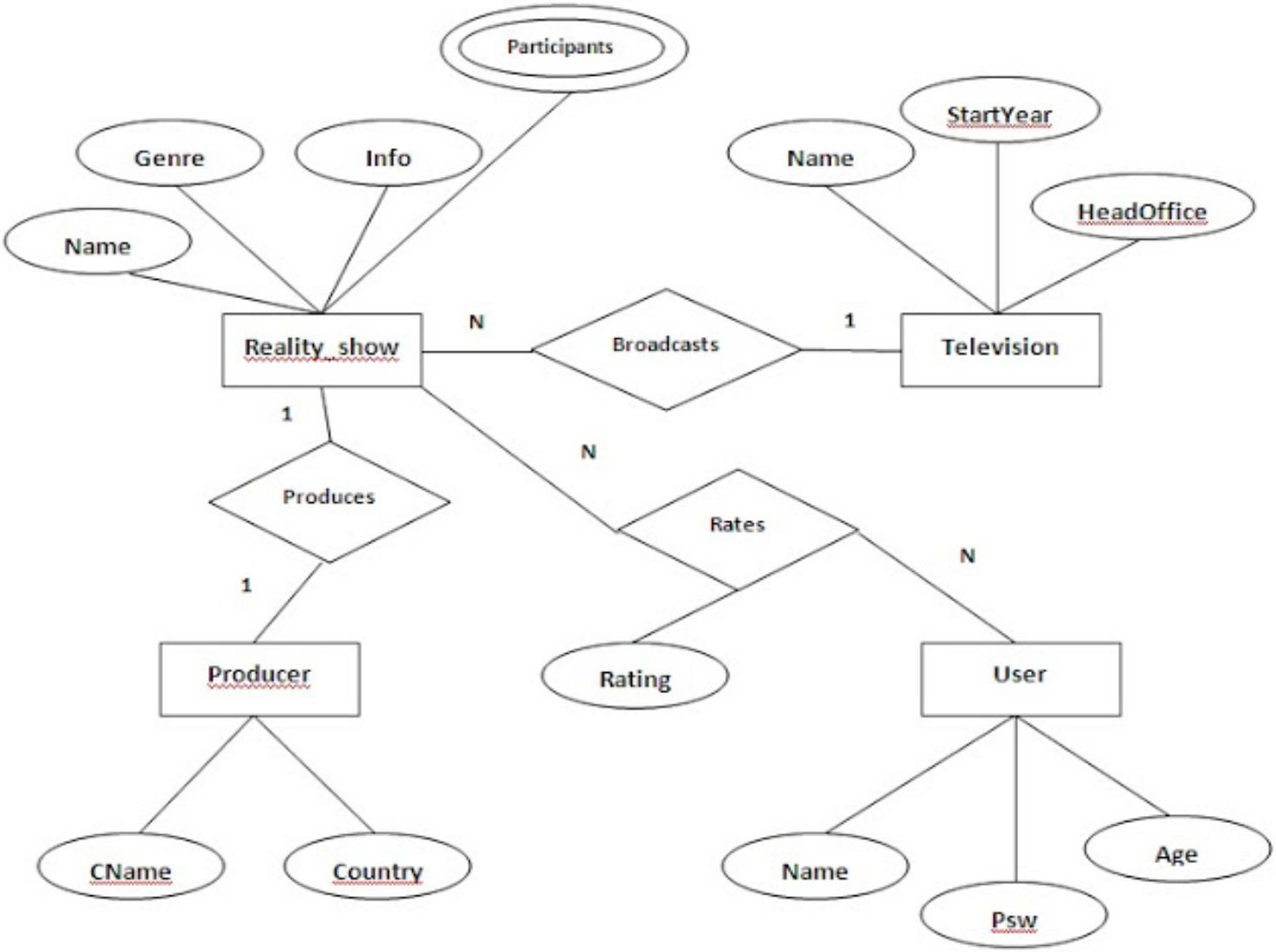
Online Shopping



Railway Management System



- **Draw an ER diagram for the given scenario;**
- Suppose that you are designing a schema to record information about reality shows on TV. Your database needs to record the following information:
- _ For each reality show, its name, genre, basic_info and participants name. Any reality show has at least two or more participants.
- _ For each producer, the company name, company country. A show is produced by exactly one producer. And one producer produces exactly one show.
- _ For each television, its name, start year, head office. A television may broadcasts multiple shows. Each show is broadcasted by exactly one television.
- _ For each user, his/her username, password, and age. A user may rate multiple shows, and a show may be rated by multiple users. Each rating has a score of 0 to 10.
- Draw an entity relationship diagram for this database.

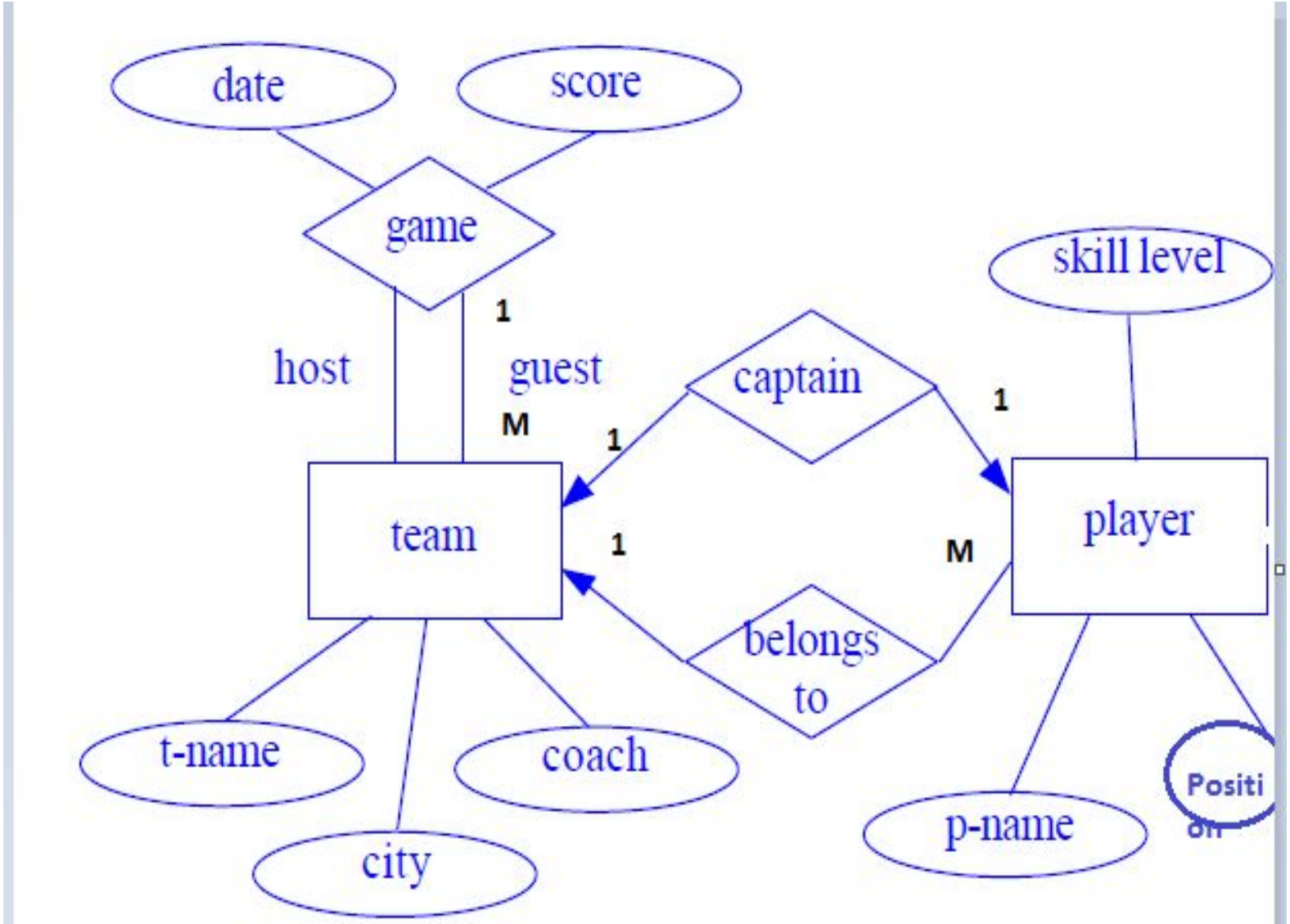

Participants
Genre
Info
Name
Reality_show
N
Broadcasts
1
Television
StartYear
Name
HeadOffice
1
Produces
1
Producer
CName
Country
N
Rates
Rating
N
User
Name
Psw
Age

Practice ER Diagram Question

Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as *left wing or goalie), a skill level, and a set* of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as *May 11th, 1999) and a score (such as 4 to 2).*
- Construct a clean and concise ER diagram for the NHL database using the Chen notation as in your textbook.
- List your assumptions and clearly indicate the cardinality mappings as well as any role indicators in your ER diagram.

Here is one sample solution.


date
score
game
1
host
guest
M
captain
1
skill level
1
team
1
M
player
belongs
to
t-name
coach
city
p-name
Positi
on

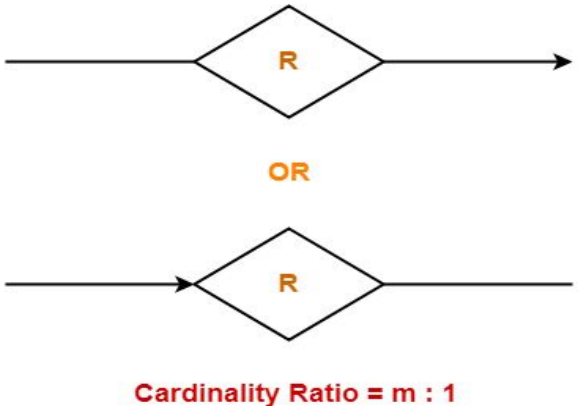
Reduction to relational schemas

- (1) The conversion of E-R diagrams into relational tables.
- (2) The data normalization technique.
- (3) The use of the data normalization technique to test the tables resulting from the E-R diagram conversions.

Converting E-R Diagrams into Relational Tables

- Each entity will convert to a table.
- Each many-to-many relationship or associative entity will convert to a table.
- During the conversion, certain rules must be followed to ensure that foreign keys appear in their proper places in the tables.

M:1



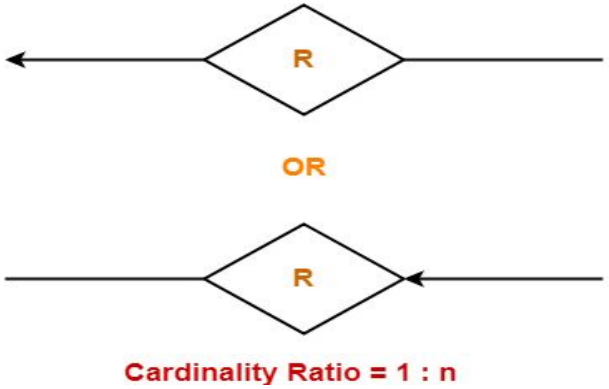
M:1(2 tables)
A (a1, a2, b1) B (b1, b2)

M:M

M:M (3 tables)
A(a1, a2) B(b1, b2) R(a1, b1)

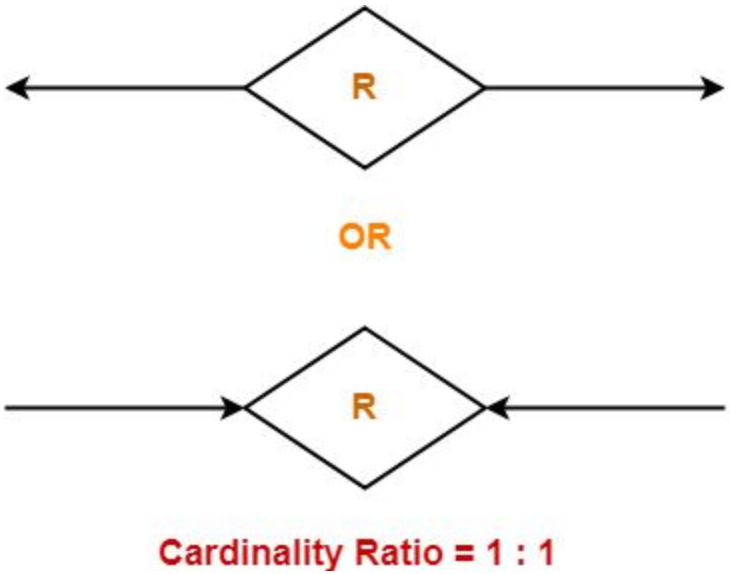
indicates primary key _
indicates foreign key_ _

1:M



1:M(2 tables)
A (a1, a2) B (b1, b2, a1)

1:1



1:1 (2 tables)
A (a1, a2, b1) B(b1, b2)
OR
A (a1, a2) B(b1, b2, a1)

Converting a Simple Entity

Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire

SALESPERSON

- The table simply contains the attributes that were specified in the entity box.
- Salesperson Number is underlined to indicate that it is the unique identifier of the entity and the primary key of the table.

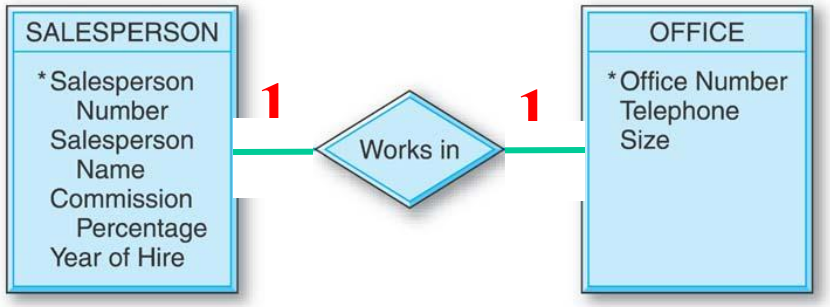
Transforming E-R Diagrams into Relations

Represent Relationships

– Binary or Unary 1:1

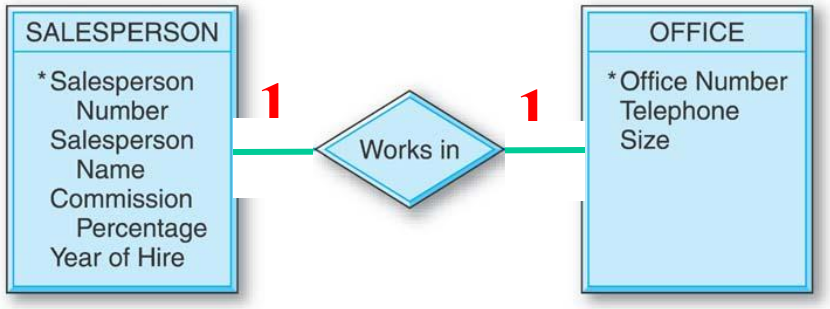
- Three possible options
  a. Both
  b. Add the primary key of A as a foreign key of B
  c. Add the primary key of B as a foreign key of A

Converting Entities in Binary Relationships: One-to-One



- There are three options for designing tables to represent this data.

One-to-One: Option #1

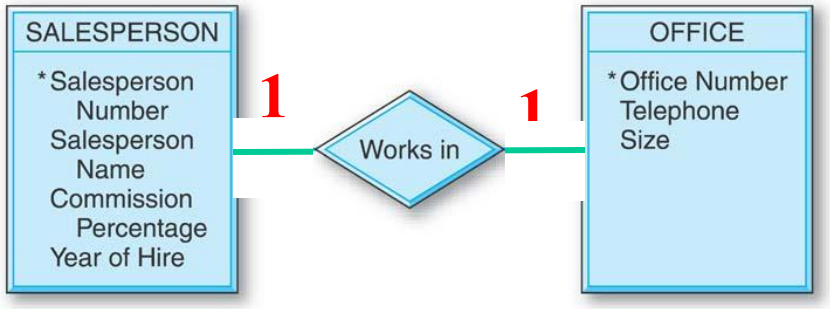


- The two entities are combined into one relational table.

SALESPERSON/OFFICE

Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire	Office Number	Telephone	Size

One-to-One: Option #2



- Separate tables for the SALESPERSON and OFFICE entities, with Salesperson Number as a foreign key in the OFFICE table.

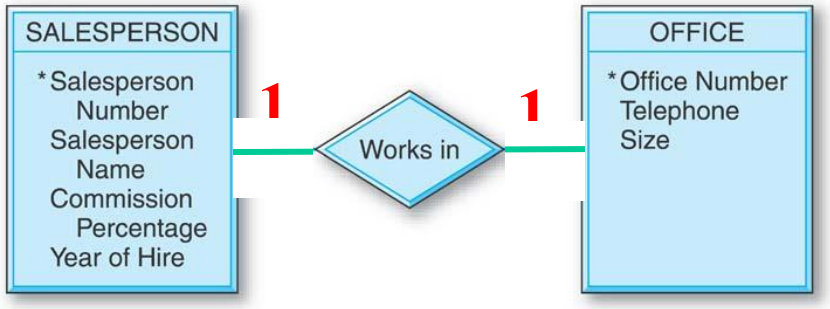
SALESPERSON

Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire

OFFICE

Office Number	Telephone	Size	Salesperson Number

One-to-One: Option #3



- Separate tables for the SALESPERSON and OFFICE entities, with Office Number as a foreign key in the SALESPERSON table.

SALESPERSON

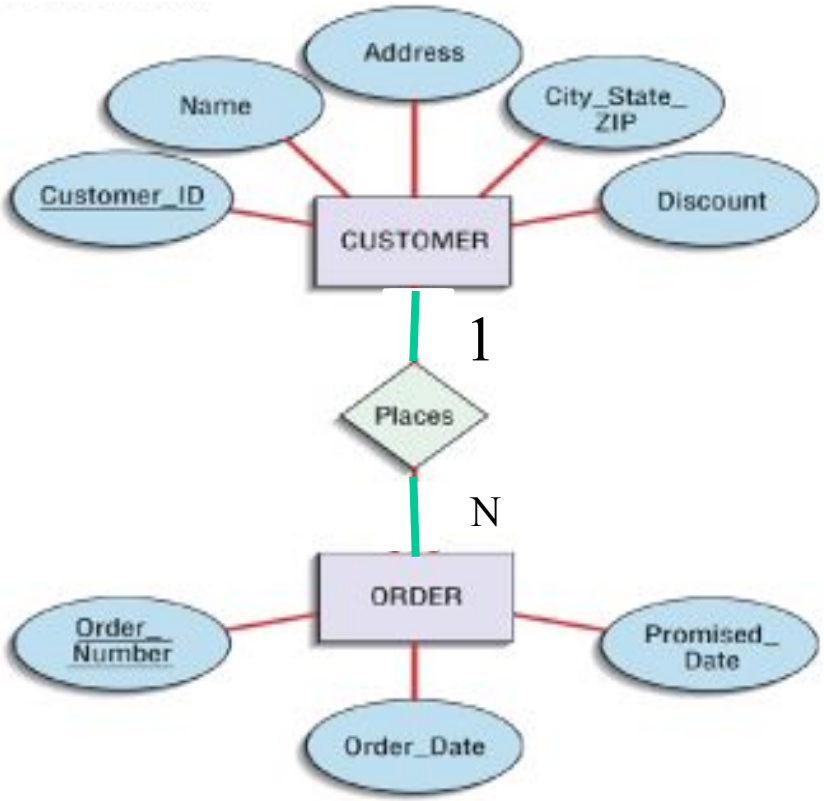
Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire	Office Number

OFFICE

Office Number	Telephone	Size

Transforming E-R Diagrams into Relations

- Binary 1:N Relationships
  - Add the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation on the right side
  - The one side *migrates* to the many side



CUSTOMER

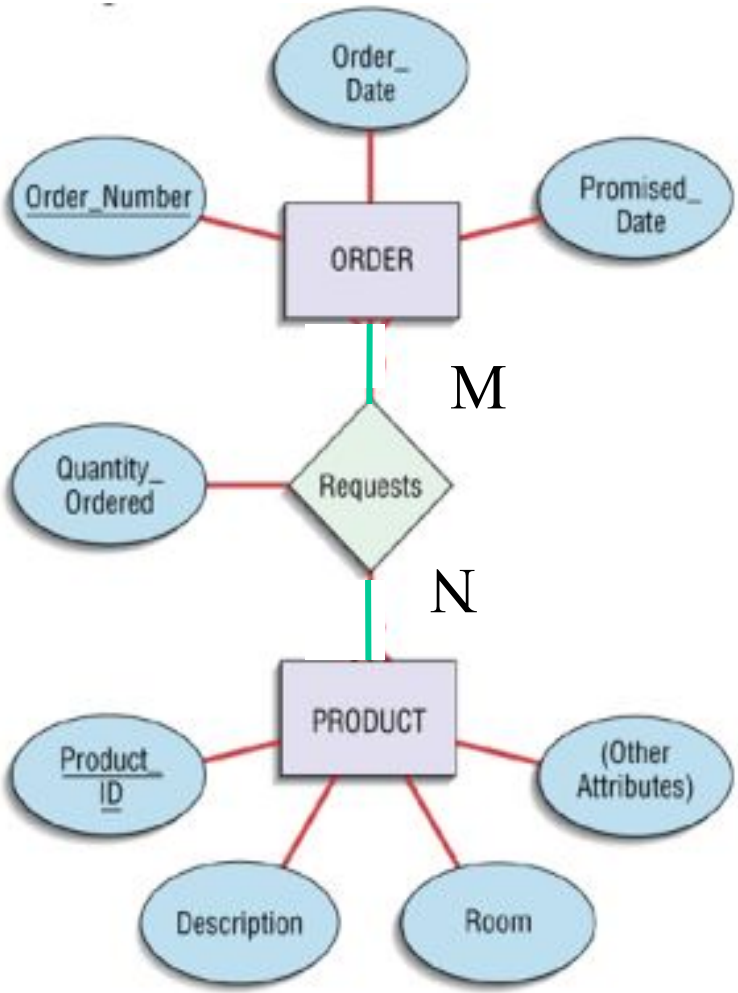
Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/0X	3/28/0X	6390
63725	3/17/0X	4/01/0X	1273
80149	3/14/0X	3/24/0X	6390

Transforming E-R Diagrams into Relations

2. Represent Relationships (continued)
   - Binary and higher M:N relationships
     - Create another relation and include primary keys of all relations as primary key of new relation



ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2002	3/01/2002
62009	2/13/2002	2/27/2002
62807	2/15/2002	3/01/2002

ORDER LINE

Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

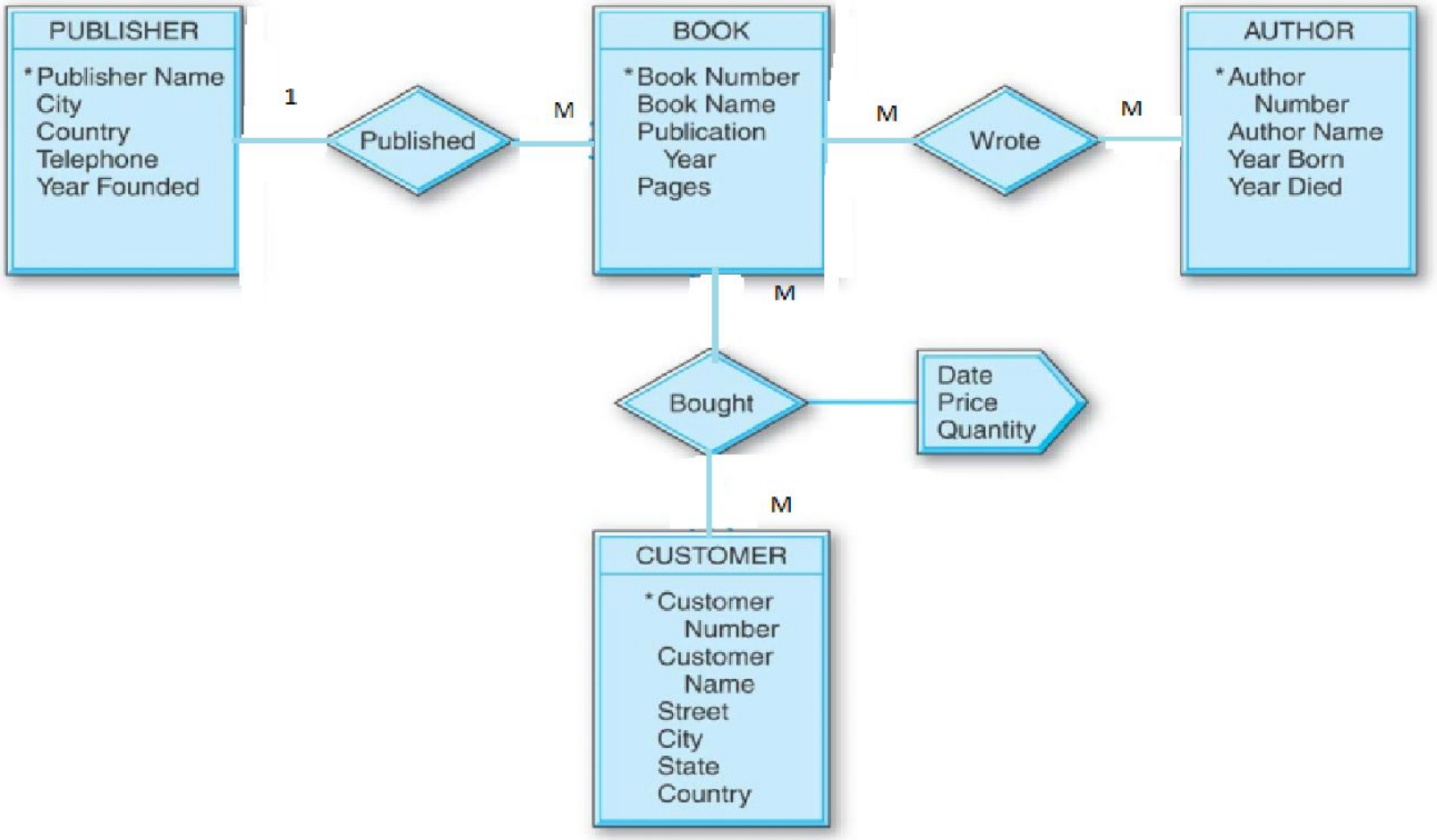
PRODUCT

Product_ID	Description	(Other Attributes)
M128	Bookcase	—
A261	Wall unit	—
R149	Cabinet	—

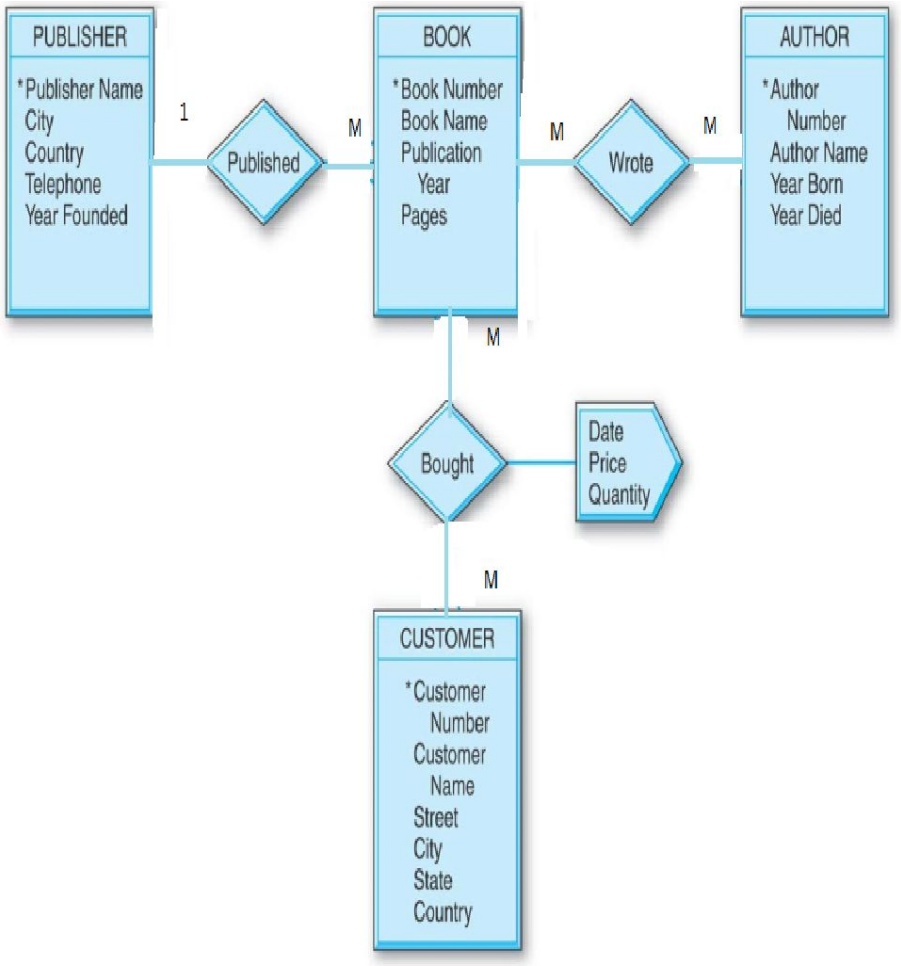
Transforming E-R Diagrams into Relations

- Unary 1:N Relationships
  - Relationship between instances of a single entity type
  - Utilize a recursive foreign key
    - A foreign key in a relation that references the primary key values of that same relation
- Unary M:N Relationships
  - Create a separate relation
  - Primary key of new relation is a composite of two attributes that both take their values from the same primary key

Convert the given E-R diagram into relational schema



Convert the given E-R diagram into relational schema



Publisher Name	City	Country	Telephone	Year Founded
PUBLISHER				

Author Number	Author Name	Year Born	Year Died
AUTHOR			

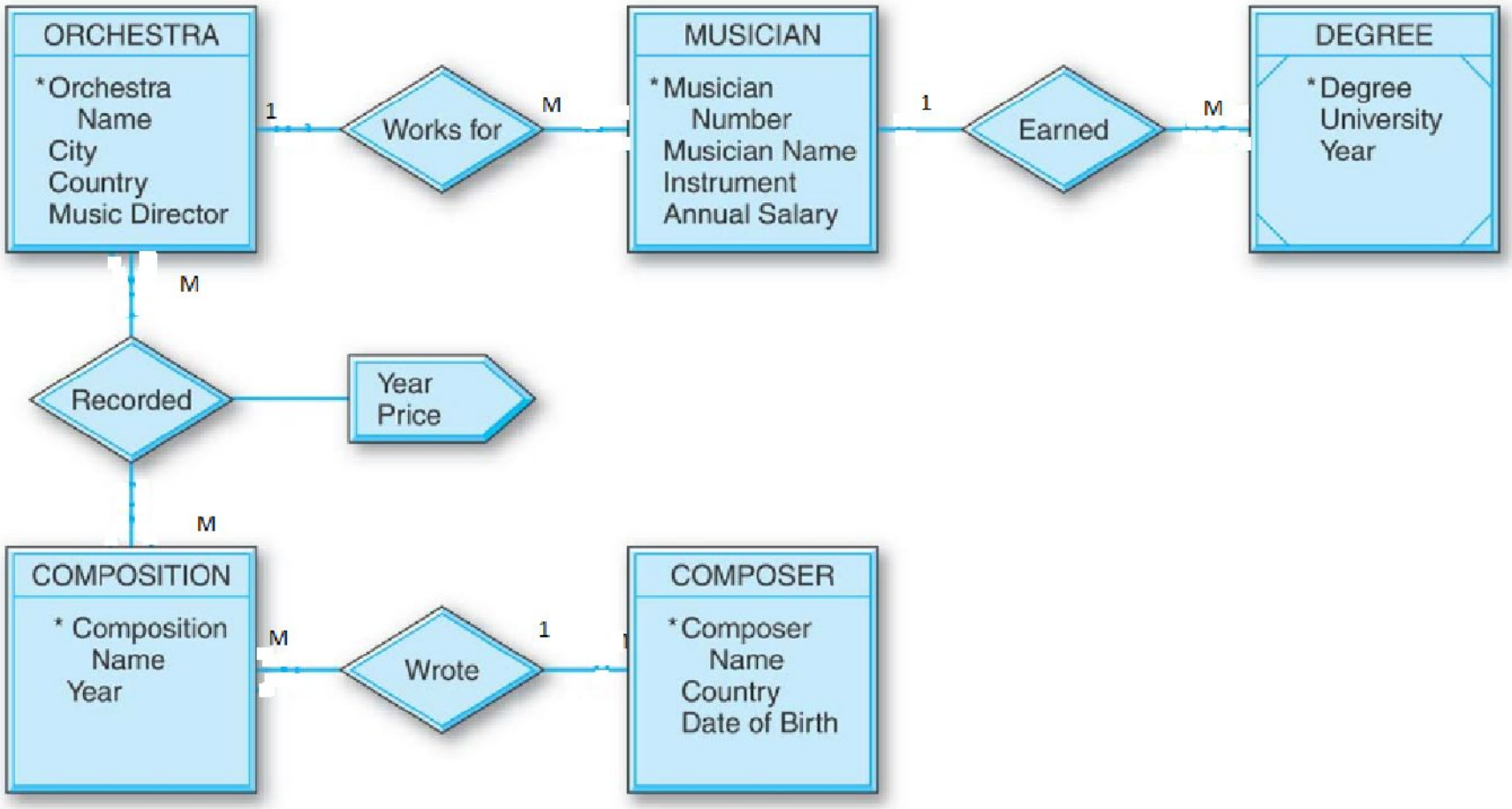
Book Number	Book Name	Publication Year	Pages	Publisher Name
BOOK				

Customer Number	Customer Name	Street	City	State	Country
CUSTOMER					

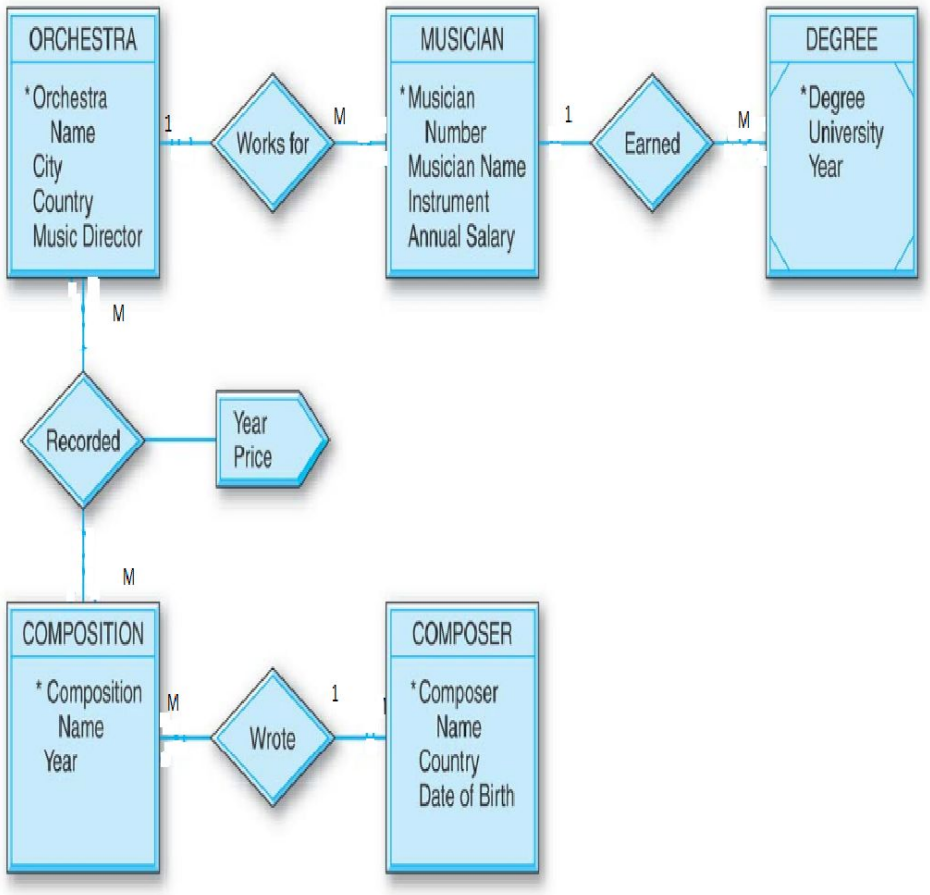
Book Number	Customer Number	Date	Price	Quantity
SALE				

Book Number	Author Number
WRITING	

Convert the given E-R diagram into relational schema



Convert the given E-R diagram into relational schema



Orchestra Name	City	Country	Music Director

ORCHESTRA

Musician Number	Degree	University	Year

DEGREE

Musician Number	Musician Name	Instrument	Annual Salary	Orchestra Name

MUSICIAN

Composition Name	Composer Name	Year

COMPOSITION

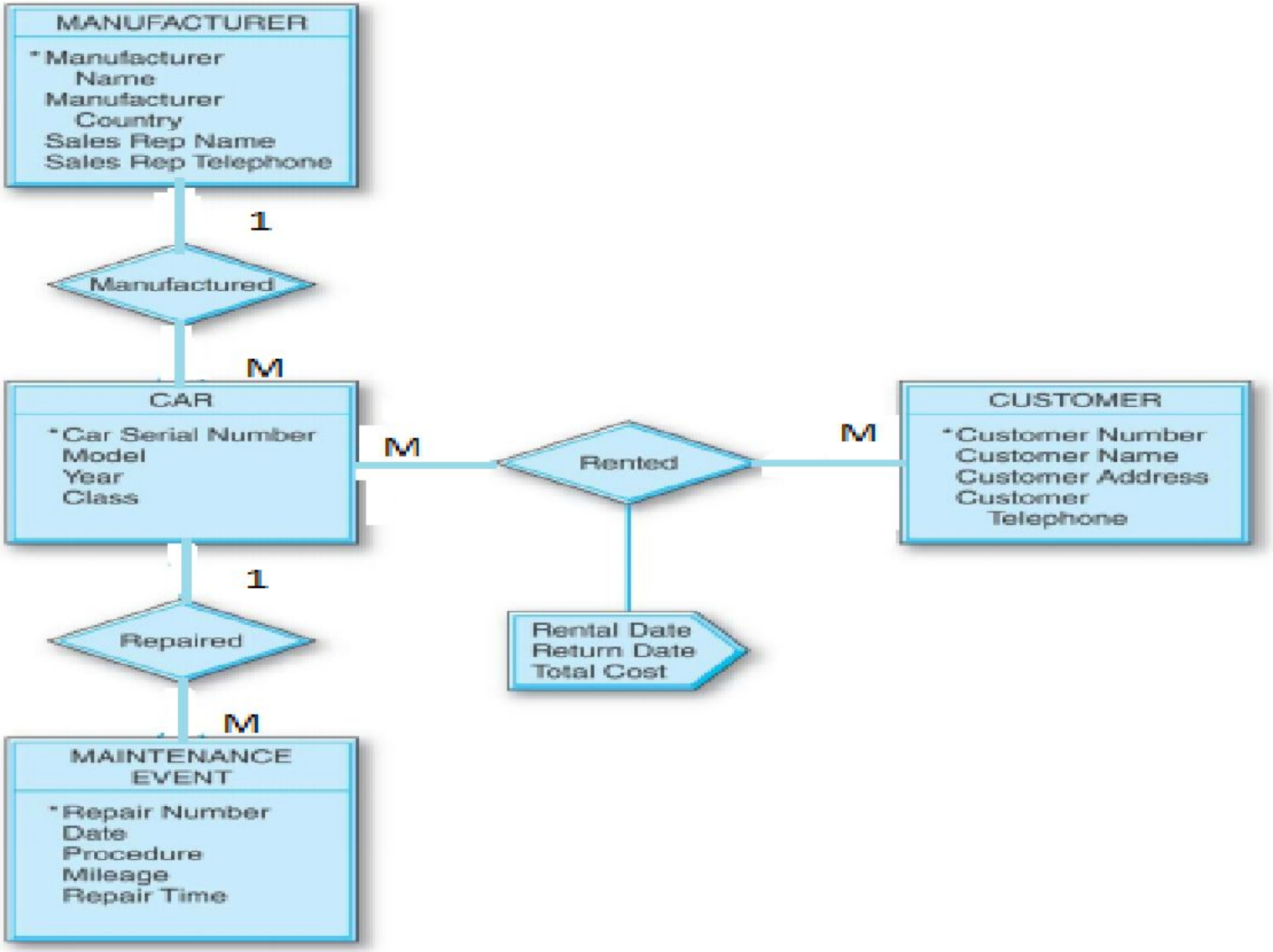
Composer Name	Country	Date Of Birth

COMPOSER

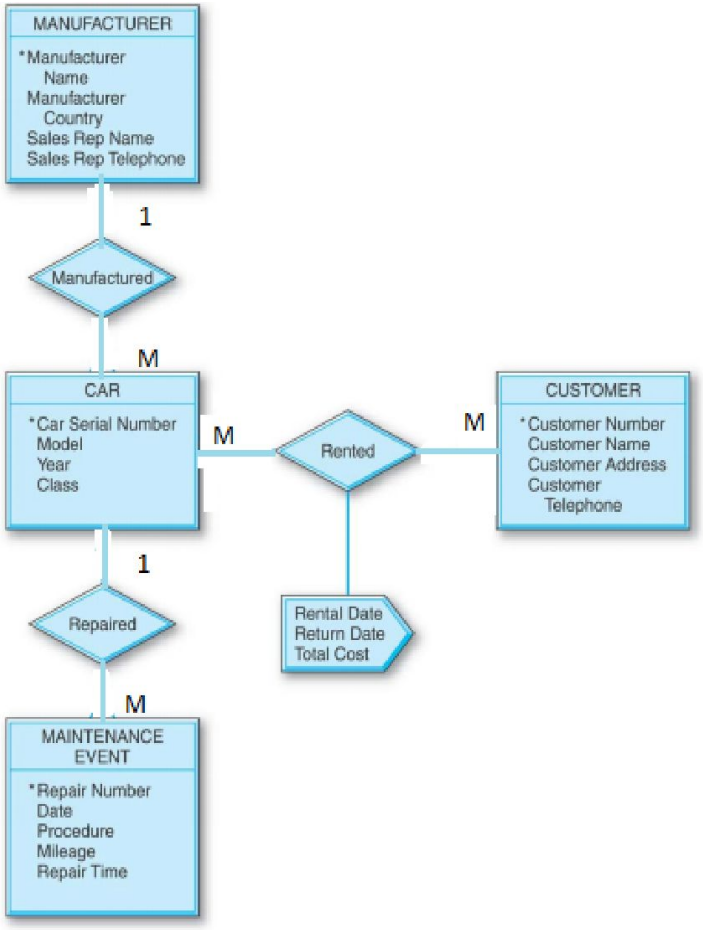
Orchestra Name	Composition Name	Composer Name	Year	Price

RECORDING

Convert the given E-R diagram into relational schema



Convert the given E-R diagram into relational schema



Manufacturer Name	Manufacturer Country	Sales Rep Name	Sales Rep Telephone

MANUFACTURER

Car Serial Number	Model	Year	Class	Manufacturer Name

CAR

Repair Number	Car Serial Number	Date	Procedure	Mileage	Repair Time

MAINTENANCE

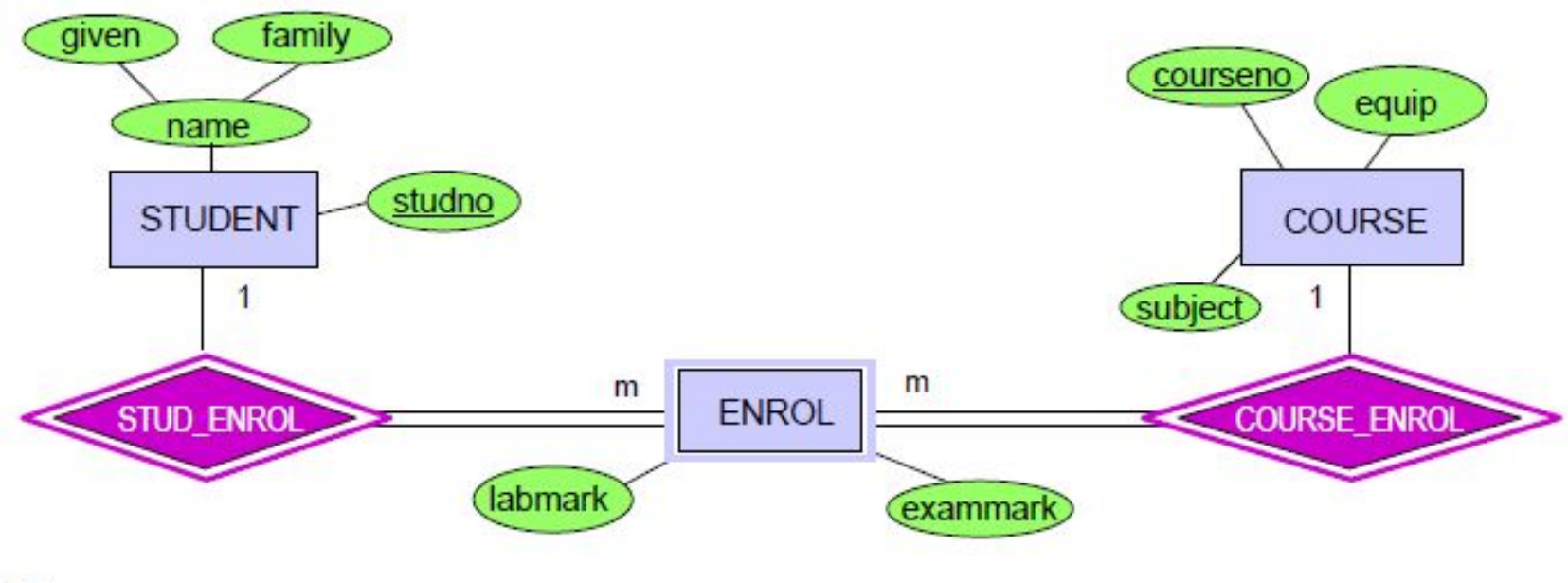
Customer Number	Customer Name	Customer Address	Customer Telephone

CUSTOMER

Car Serial Number	Customer Number	Rental Date	Return Date	Total Cost

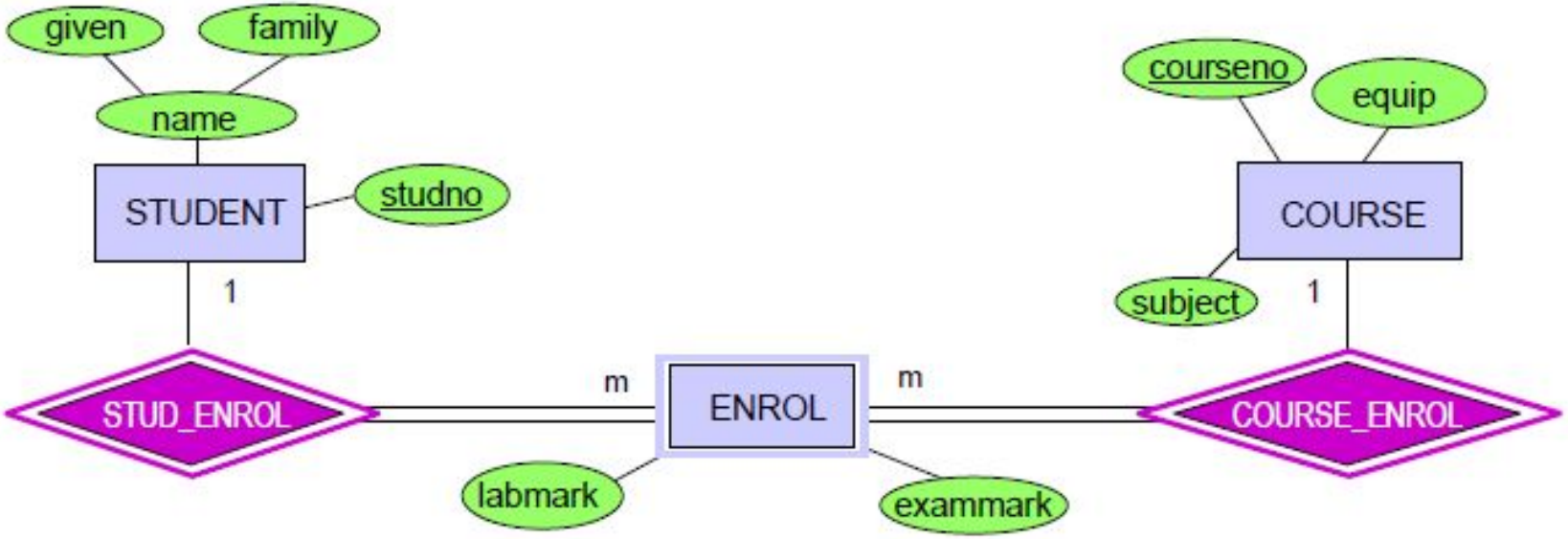
RENTAL

Convert the given E-R diagram into relational schema



Convert the given E-R diagram into relational schema

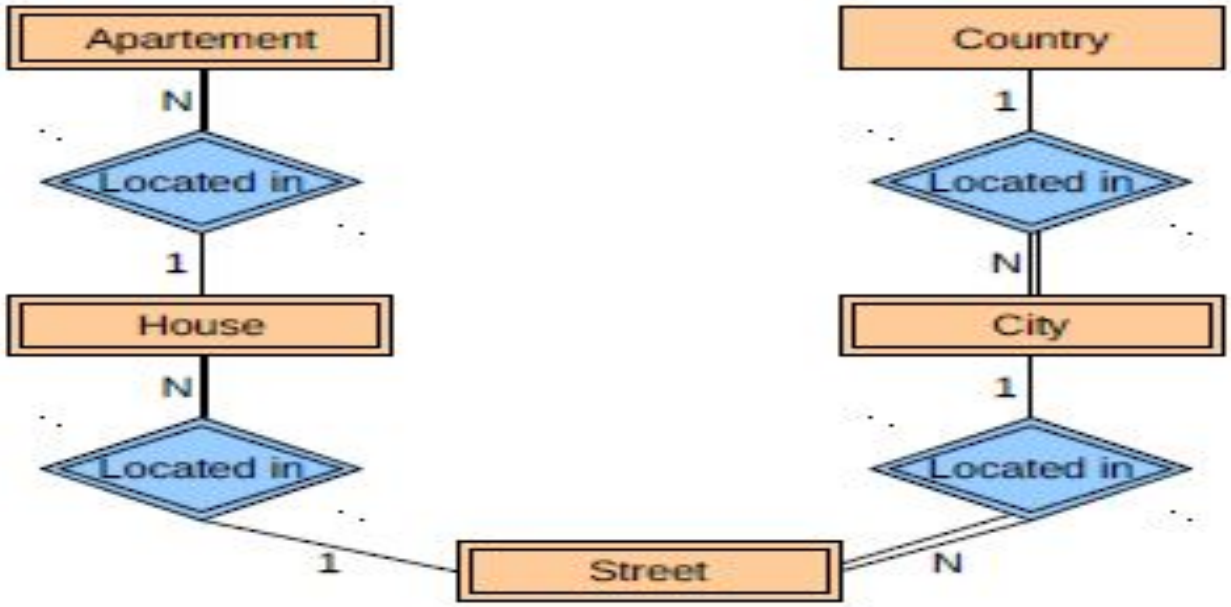
- COURSE(courseno, subject, equip)
- STUDENT(studno, givenname, familyname)



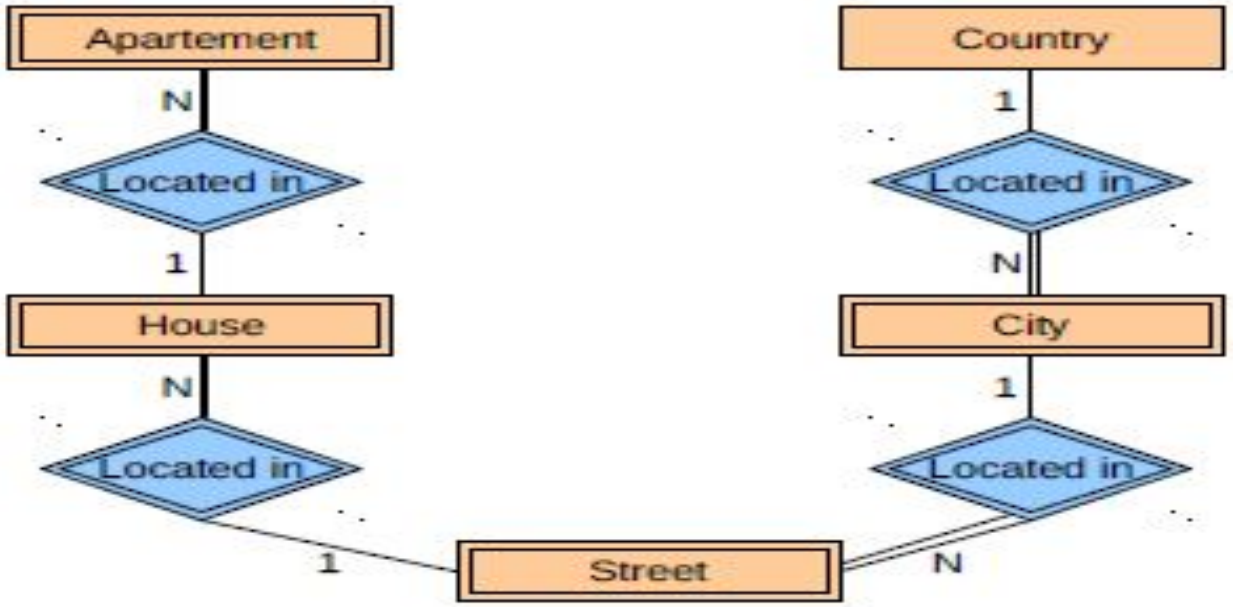
Then:

- ENROL(courseno, studno, labmark, exammark)

Convert the given E-R diagram into relational schema

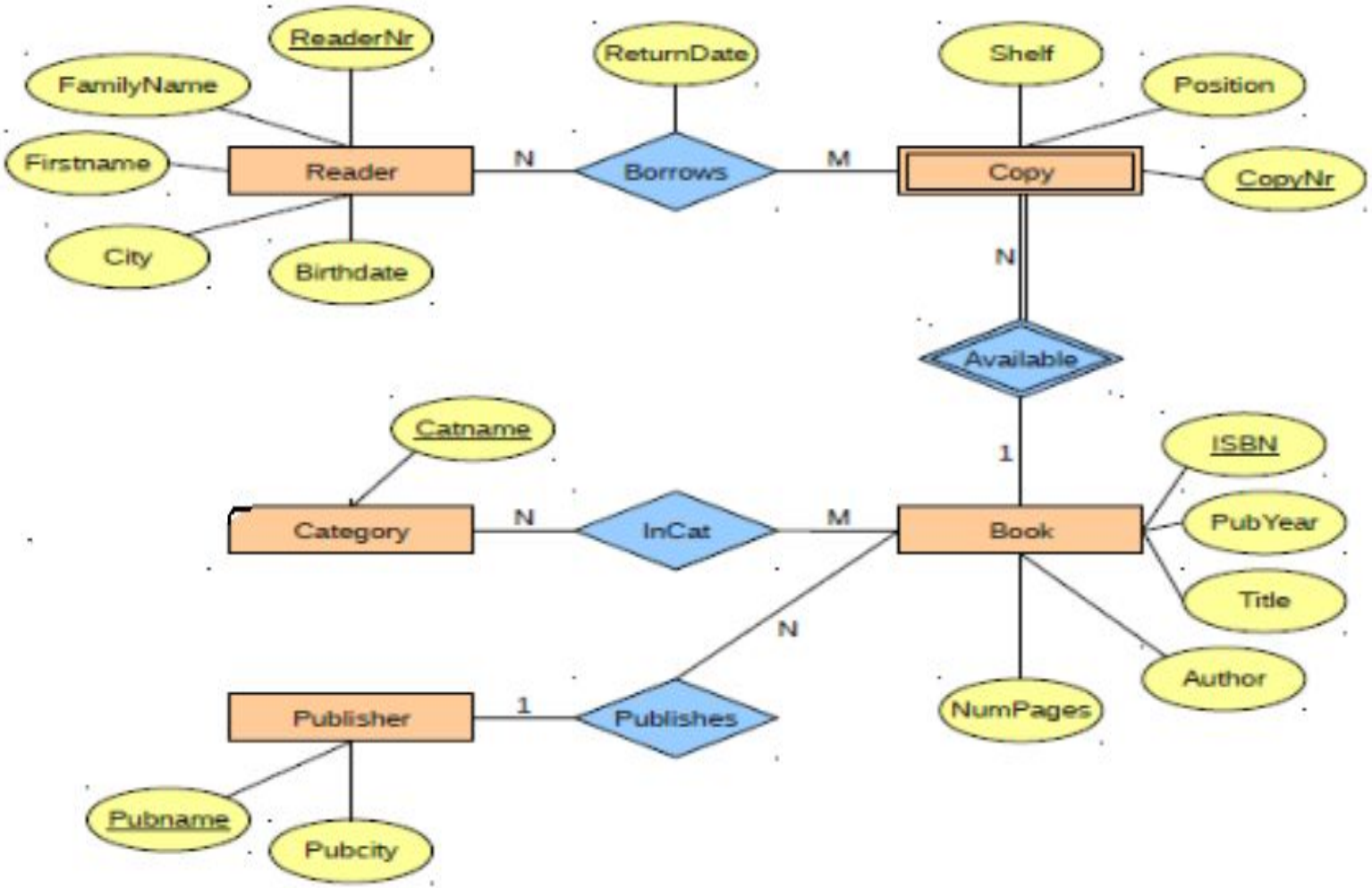


Convert the given E-R diagram into relational schema

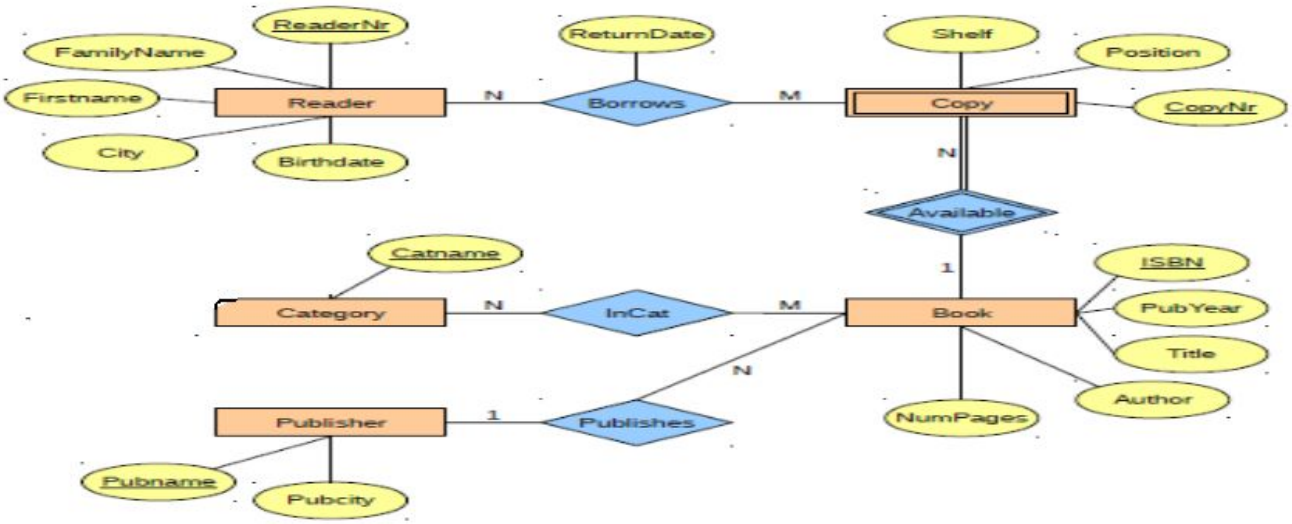


- **Country (<u>CoName</u>)**
- **City (<u>CoName</u>, <u>CiName</u>)**
- **Street (<u>CoName</u>, <u>CiName</u>, <u>SName</u>)**
- **House (<u>CoName</u>, <u>CiName</u>, <u>SName</u>, <u>HNumber</u>)**
- **Apartement (<u>CoName</u>, <u>CiName</u>, <u>SName</u>, <u>HNumber</u>, <u>ANumber</u>)**

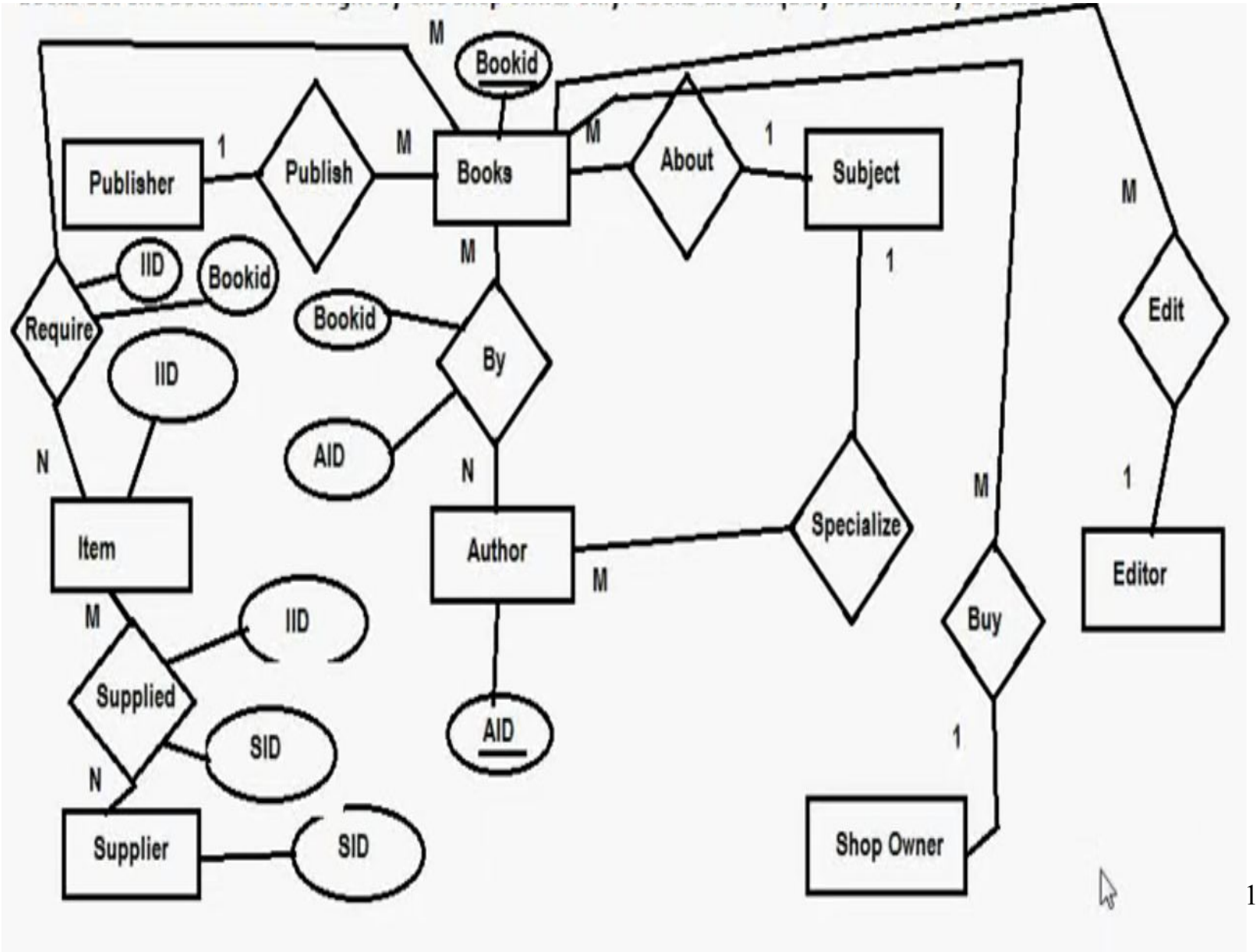
Convert the given E-R diagram into relational schema

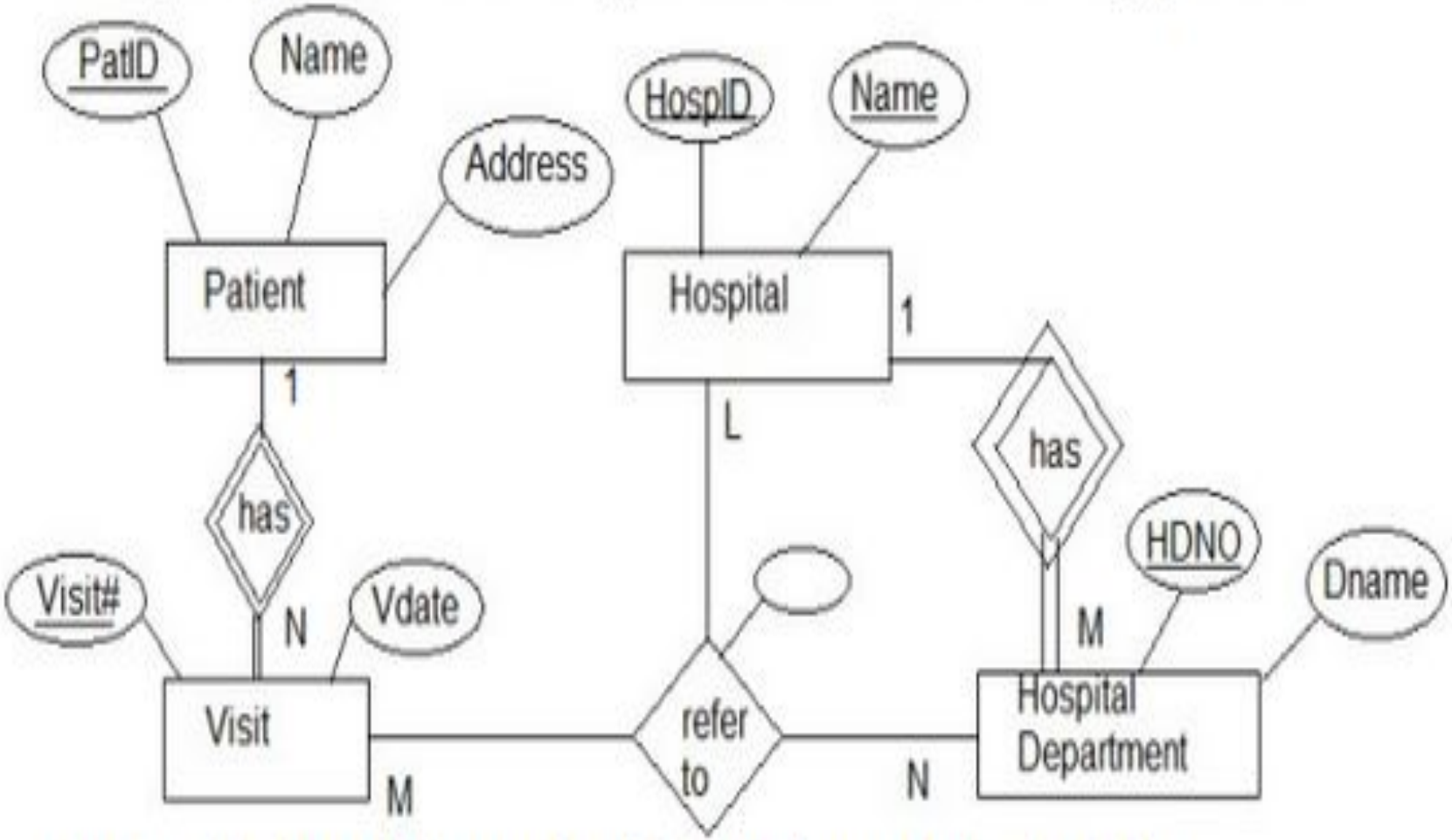


Convert the given E-R diagram into relational schema



- **Reader (<u>ReaderNr</u>, FamilyName, Firstname, City, Birthdate)**
- **Book (<u>ISBN</u>, Title, Author, NumPages, PubYear, Pubname)**
- **Copy (<u>ISBN</u>, <u>CopyNr</u>, Shelf, Position)**
- **Borrows (<u>ReaderNr</u>, <u>ISBN</u>, <u>CopyNr</u>, ReturnDate)**
- **Category (<u>Catname</u>)**
- **Publisher (<u>Pubname</u>, Pubcity)**
- **InCat (<u>ISBN</u>, <u>Catname</u>)**


M
Bookid
Publisher
1
Publish
M
Books
M
About
1
Subject
M
IID
Bookid
Require
M
1
Edit
Bookid
By
IID
AID
N
N
Item
Author
Specialize
M
1
M
Editor
Buy
IID
M
Supplied
AID
SID
1
N
Supplier
SID
Shop Owner


PatID
Name
Address
Patient
1
has
N
Visit#
Vdate
Visit
M
HospID
Name
Hospital
1
L
has
M
HDNO
Dname
refer
to
N
Hospital
Department

The Music Database

The music database stores details of a personal music library, and could be used to manage your MP3, CD, or vinyl collection.

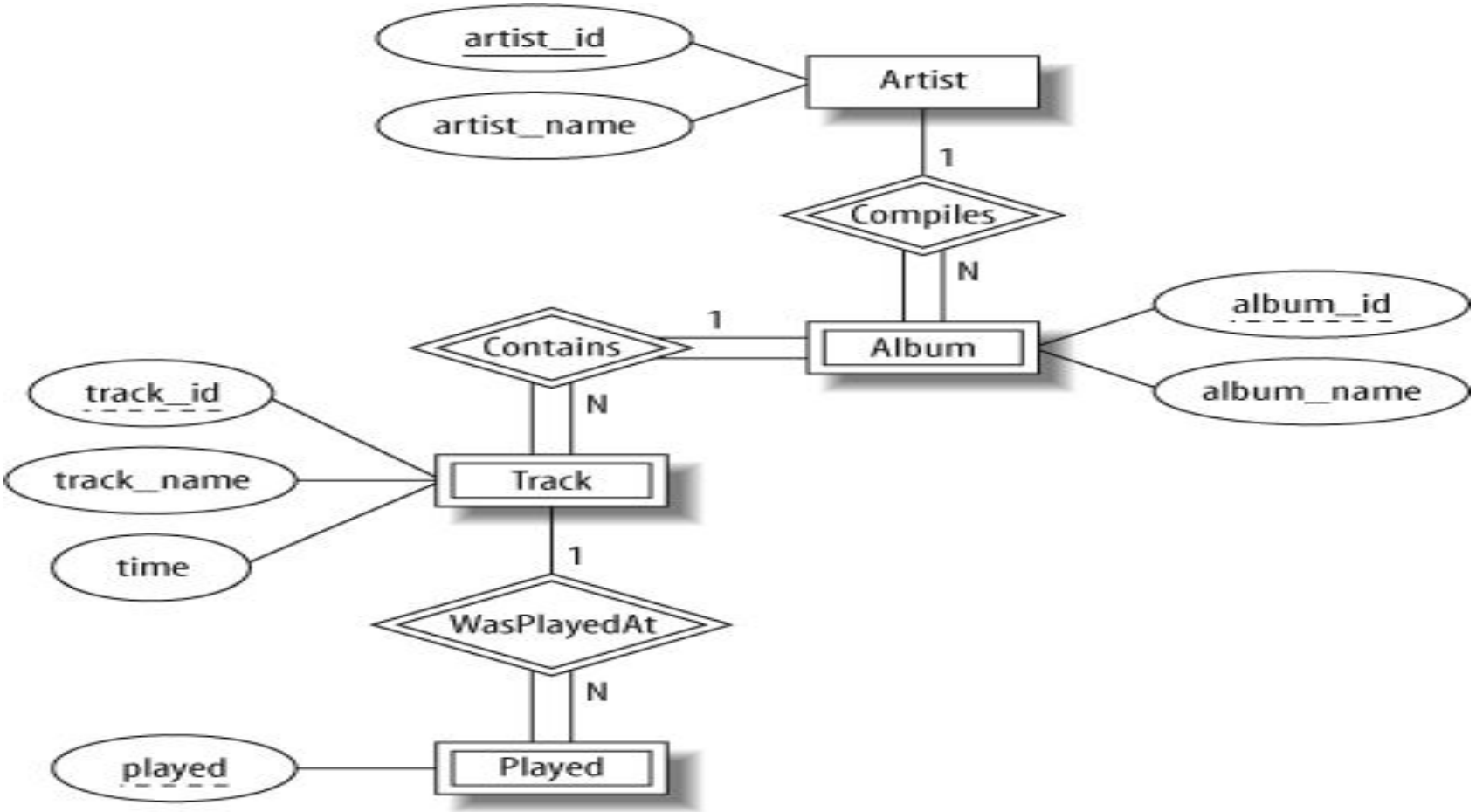
Because this database is for a personal collection, it's relatively simple and stores only the relationships between artists, albums, and tracks.

It ignores the requirements of many music genres, making it most useful for storing popular music and less useful for storing jazz or classical music.

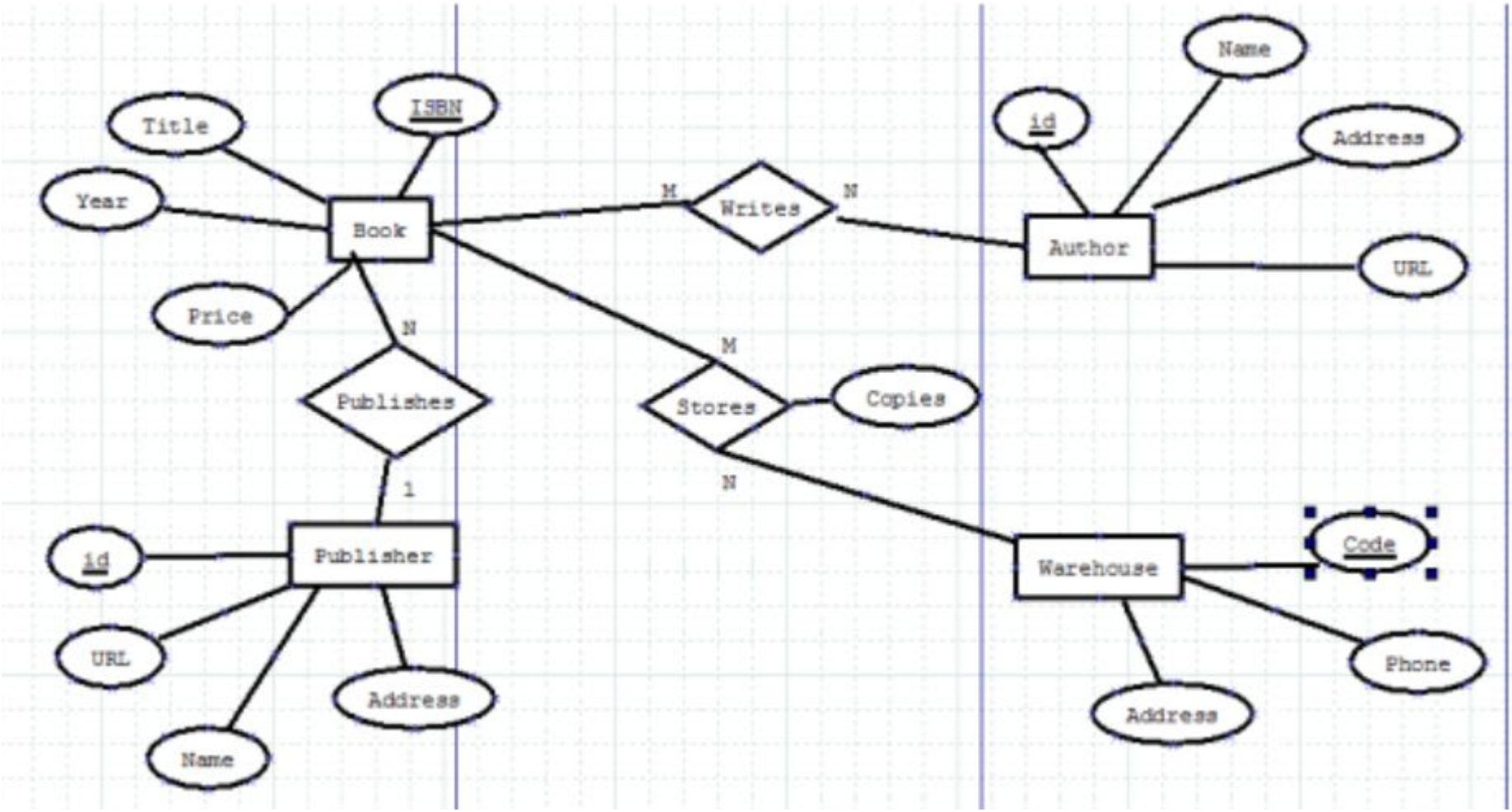
We first draw up a clear list of requirements for our database:

- The collection consists of albums.
- An album is made by exactly one artist.
- An artist makes one or more albums.
- An album contains one or more tracks
- Artists, albums, and tracks each have a name.
- Each track is on exactly one album.
- Each track has a time length, measured in seconds.

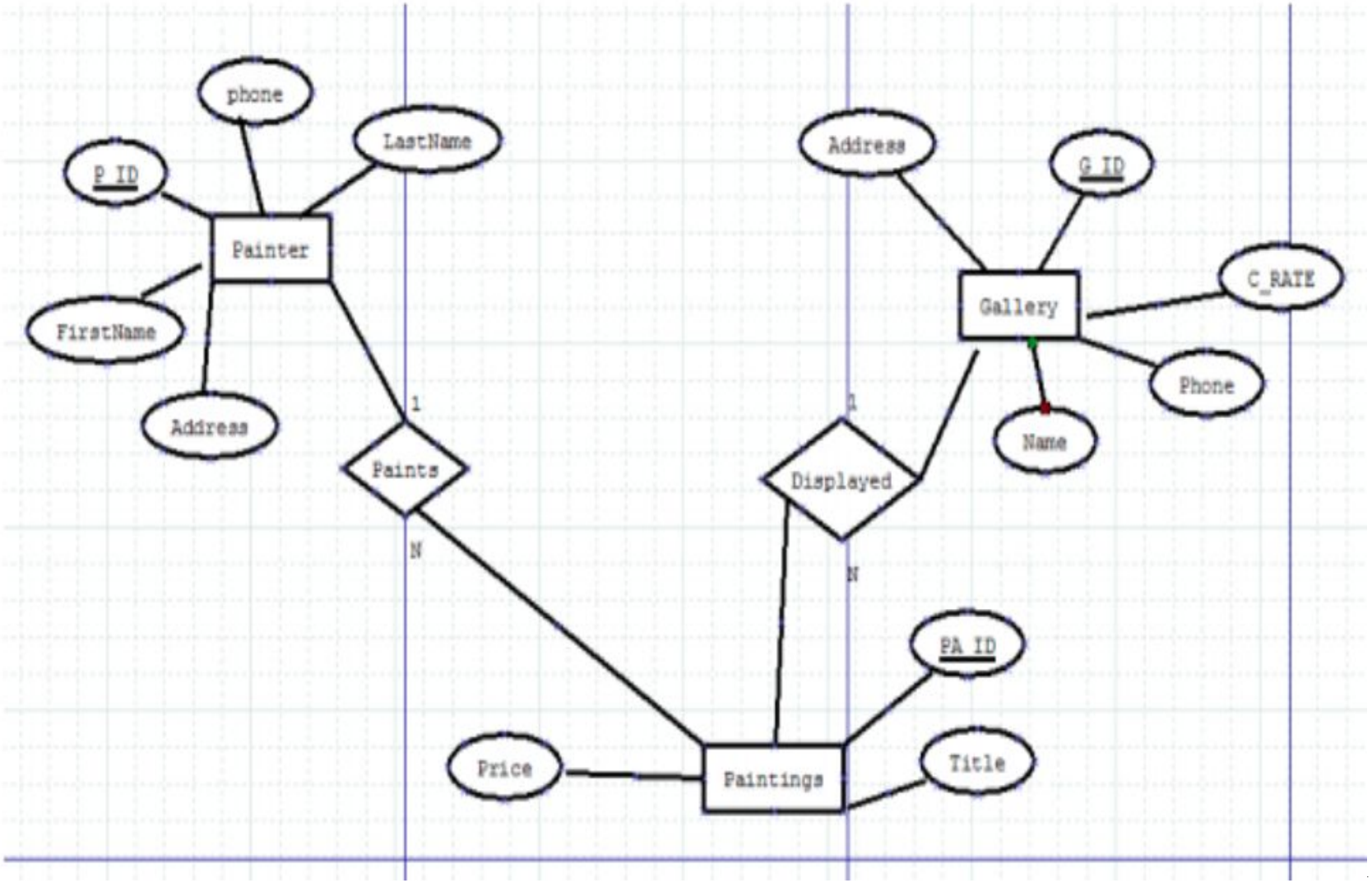
•When a track is played, the date and time the playback began (to the nearest second) should be recorded;
• this is used for reporting when a track was last played,
•as well as the number of times music by an artist, from an album, or a track has been played.
There's no requirement to capture composers, group members or sidemen, recording date or location,
the source media, or any other details of artists, albums, or tracks.
The ER diagram derived from our requirements.
You'll notice that it consists of only one-to-many relationships:
one artist can make many albums, one album can contain many tracks, and one track can be played many times.
Conversely, each play is associated with one track, a track is on one album, and an album is by one artist.
The attributes are straightforward: artists, albums, and tracks have names, as well as identifiers to uniquely identify each entity.
The track entity has a time attribute to store the duration, and the played entity has a timestamp to store when the track was played.


artist_id
artist_name
Artist
1
Compiles
N
album_id
album_name
Album
1
Contains
N
track_id
track_name
time
Track
1
WasPlayedAt
N
played
Played

- The database must store book, author, publisher and warehouse information.
- For every book you must capture the title, isbn, year and price information. The isbn value is unique for a book.
- For every author you must store an id, name, address and the URL of their homepage. Each author can write many books, and each book can have many authors, for example.
- For every publisher you must store an id, name, address, phone number and an URL of their website.
- Books are stored at several warehouses, each of which has a code, address and phone number.
- A book has only one publisher.
- The warehouse stocks many different books. A book may be stocked at multiple warehouses.
- The database records the number of copies of a book stocked at various warehouses.
- Design an ER diagram for such a bookstore. Your ER diagram must show entities, attributes and the relationships between entities. [Document any assumptions that you make]


Title
ISBN
Year
Book
Price
Publishes
Publisher
id
URL
Address
Name
M
Writes
N
Author
Name
id
Address
URL
N
M
Stores
Copies
N
Warehouse
Code
Phone
Address

- United Direct Artists (UDA) is an insurance broker that specialise in insuring paintings for galleries. You are required to design a database for this company.
- The database must store painters, paintings, and galleries information.
- Painters have a unique number, Name, and phone number
- Paintings have unique number, title and price
- Galleries have unique number, owner, phone number, commission rate and address
- A painting is painted by a particular artist, and that painting is exhibited in a particular gallery. A gallery can exhibit many paintings, but each painting can be exhibited in only one gallery. Similarly, a painting is painted by a single painter, but each painter can paint many paintings.


phone
LastName
P_ID
Painter
FirstName
Address
1
Paints
N
Address
G_ID
C_RATE
Gallery
Phone
Name
1
Displayed
N
PA_ID
Price
Paintings
Title

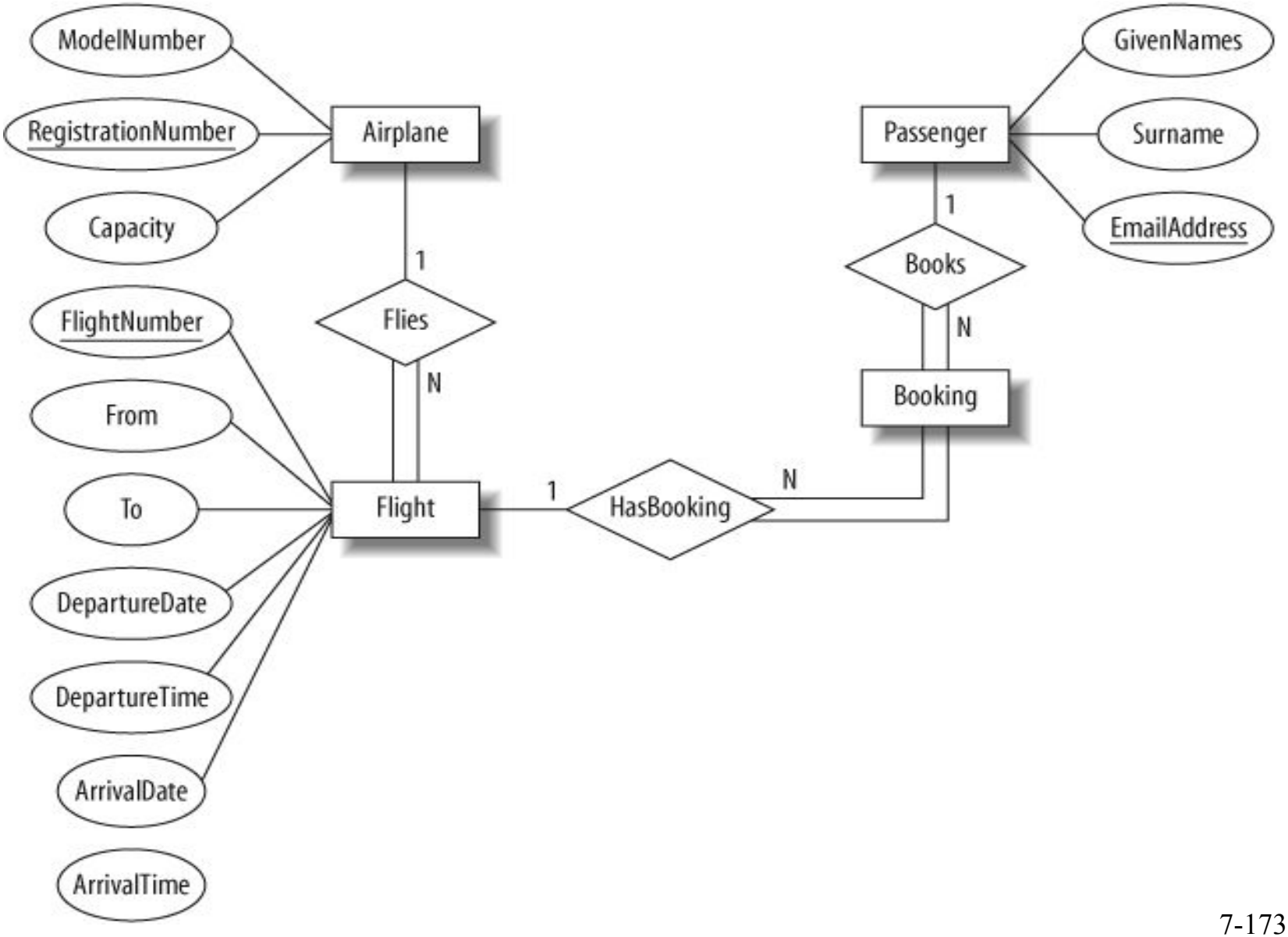
The Flight Database

The flight database stores details about an airline's fleet, flights, and seat bookings. Again, it's a hugely simplified version of what a real airline would use, but the principles are the same.

Consider the following requirements list:

- The airline has one or more airplanes.
- An airplane has a model number, a unique registration number, and the capacity to take one or more passengers.
- An airplane flight has a unique flight number, a departure airport, a destination airport, a departure date and time, and an arrival date and time.
- Each flight is carried out by a single airplane.
- A passenger has given names, a surname, and a unique email address.
- A passenger can book a seat on a flight.

The ER diagram derived from our requirements is shown in Figure 4-13:


ModelNumber
RegistrationNumber
Capacity
Airplane
1
Flies
N
FlightNumber
From
To
DepartureDate
DepartureTime
ArrivalDate
ArrivalTime
Flight
1
HasBooking
N
Booking
N
Books
1
Passenger
GivenNames
Surname
EmailAddress