CHAPTER 1

1.1

Define the following terms: data, database, DBMS, database system, database catalog, program-data independence, user view, DBA, end user, canned transaction, deductive database system, persistent object, meta-data, and transaction-processing application.

SOLUTION

Data –
Collection of known facts that can be recorded and have an implicit meaning

Database –
A collection of related data

DBMS –
It is a software which facilitates the creation and maintenance of a database.

Database System -
The DBMS software together with the data itself, sometimes the applications are also included.

Database Catalog –
Metadata such as definitions of database objects such as base tables, views and other function are stored.

Program Data Independence –
- Type of data transparency.
- Independence of changes between user applications and definition and organization of data

User View –
A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Each user may see a different view of the database, which describes only the data of interest to that user

DBA –
Database Administrator; A person who is responsible for maintenance of a database environment by directing or performing all related activities to keep the data secure.

End User –
A person who ultimately uses a product.

Canned Transaction -
Standard types of queries and updates which are frequently used by Naïve end users to constantly querying and updating database.

Deductive database system –
Database system that can make deductions based on rules and facts stored in the database

Persistent Object –
An object that has been assigned a storage location in a federated database

Meta Data –
A set of data that describes about other data

Transaction Processing Application –
A set of information which processes the data transaction in database system that monitors transaction program

1.3

Discuss the main characteristics of the database approach and how it differs from traditional file systems.

SOLUTION

CHARACTERISTIC	DATABASE APPROACH	FILE SYSTEM APPROACH
Self-describing nature of a database system	A DBMS catalog stores the description of a particular database (e.g. data structures, types, and constraints) The description is called meta-data this allows the DBMS software to work with different database application.	In file system approach, each user defines and implements the needed files for a specific application to run. File system doesn't contain any metadata
Program data independence	DBMS access programs do not require such changes in most cases. the structure of data files is stored in the DBMS catalog separately from the access programs	The structures of data files is embedded in the access programs, so many change to the structure of a file may require changing all programs that access this file.
Data abstraction	A data model is a type of data abstraction that is used to hide storage details and present the users with a conceptual view of the database.	No such data model is present in file system
Support multiple views of the data	A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.	If any changes are made to the structure of the files it will affect all the programs, so changes to the structure of a file may require changing of all programs that access the file
Sharing of data and multi user transaction processing	Allowing a set of concurrent users to retrieve from and to update the database. concurrency control within the DBMS guarantees that each transaction is correctly executed or aborted. Recovery subsystem ensures each completed transaction has its effect permanently recorded in the database	Cresting files over a long period leads to a possibility of information getting duplicated, this redundancy is storing same data multiple times leads to higher costs and wastage of space. This may result in data inconsistency in the application, this is because update is done to some of the files only and not all the files.

.

1.6

Discuss the capabilities that should be provided by a DBMS.

SOLUTION

- Backup and recovery
- Enforce Integrity constraints
- Restricting access
- Control redundancy [Normalization]
- Persistent storage for program objects
- Storage structures for efficient query processing
- Represent complex relationships among data
- Multiple user interfaces [query language, programming language interfaces]

1.7

Discuss the differences between database systems and information retrieval systems.

SOLUTION

	DATABASE SYSTEMS	INFORMATION RETRIEVAL SYSTEMS
INFORMATION	Given database represents ground truth of facts and uses this database to generate information	Uses a network/web to gather information
APPLICATION	Applies to structured and formatted data that arises in routine applications.	Concerned with searching for material based on these keywords and with many problems dealing with document processing and free form text processing
STORAGE	Data is structured and the definition of structure is stored in catalog	Data is indexed, catalogued, and annotated using key words.

1.8

Identify some informal queries and update operations that you would expect to apply to the database shown in Figure 1.2.

SOLUTION

QUERIES

- What are the prerequisites of the Database course?
- Fetch the names of all students majoring in Mathematics.
- Fetch the transcript of the student named Brown.

UPDATES

- Insert a new student in the database whose Name=Rohith, Student_number=007, Class=6360, and Major=CS.
- Change the grade that Smith received in Discrete Mathematics to A.

1.14

Consider Figure 1.2.
a. If the name of the 'CS' (Computer Science) Department changes to 'CSSE' (Computer Science and Software Engineering) Department and the corresponding prefix for the course number also changes, identify the columns in the database that would need to be updated.
b. Can you restructure the columns in the COURSE, SECTION, and PREREQUISITE tables so that only one column will need to be updated?

SOLUTION

a.

TABLE	COLUMN
STUDENT	Major
COURSE	Course_number & Department
SECTION	Course_number
PREREQUISITE	Course_number & Prerequisite_number

b.

TABLE	COLUMN	SPLIT COLUMNS
COURSE	Course_number	Course_Dept and Course_num
SECTION	Course_number	Course_Dept and Course_num
PREREQUISITE	Course_number	Course_Dept and Course_num
PREREQUISITE	Prerequisite_number	Prerequisite_Dept and Prerequisite_number

CHAPTER 2

2.1 Define the following terms: data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML, SDL, VDL, query language, host language, data sublanguage, database utility, catalog, client/server architecture, three-tier architecture, and n-tier architecture.

SOLUTION [From Text book]

Data model
A collection of concepts that can be used to describe the structure of a database-provides the necessary means to achieve this abstraction.

Database schema
The description of a database is called the database schema, which is specified during database design and is not expected to change frequently.

Database state
- The data in the database at a particular moment in time is called a database state or snapshot.
- It's also called the current set of occurrences or instances in the database.

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

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

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

Data independence
It can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

DDL
It is used by the DBA and by database designer to define both schema.

DML
The DBMS provides a set of operations or a language called the data manipulation language for typical manipulations include retrieval, insertion, deletion, and modification of the data.

SDL
It is used to specify the internal schema.

VDL
It is used to specify user views and their mappings to the conceptual schema.

Query language
A high-level DML used in a standalone interactive manner is called query language.

Host language
Whenever DML language, whether high level or low level, are embedded in a general-purpose programming language, the language is called the host language.

Data sublanguage

- Data sublanguage is short for a relationally complete data manipulation sublanguage (DML) that expresses retrievals and updates, the latter correctly understood as set-theoretic relation transformations.
- Portion of the language required to manage and modify a relational DBMS.
- It is a proper subset of a language.

Database utility
It help the DBA manage the database system.

Catalog
To share information about schemas and constraints, the data dictionary stores other information, such as design decisions, usage standards, application program description, and user info.

Client/server architecture
It divides the application system into the front-end client and the back-end server.

Three-tier architecture
It adds an intermediate layer between the client and the database server. The main usage is to improve the database security.

N-tier architecture
It is customary to divide the layers between the user and the stored data further into finer components, thereby giving rise to n-tier architectures where n may be four or five

2.3
What is the difference between a database schema and a database state?

SOLUTION

DATABASE SCHEMA
- A database schema is the skeletal structure that represent the logical view of all the entire datasets defining how the data are organized and how the relations among them are associated.

DATABASE STATE
- A database state provides the present state of the database and its data.
- It can be considered as an extension of the database schema.
- When a database is newly defined, the corresponding database state is empty.
- Factors that affect a database state are entering, deleting or modifying information in the database.

2.4
Describe the three-schema architecture. Why do we need mappings between schema levels? How do different schema definition languages support this architecture?

SOLUTION

Three level architectures consists of :
External schema, Logical schema, Internal schema

EXTERNAL SCHEMA
- It is the schema available for users.
- There is one for each class of users, it shows only the data and the relationships between data which are of interest for that class of users

LOGICAL SCHEMA
- It is the integrated view of all the external schemas
- It allows to represent all the data and the relationships without redundancies

INTERNAL SCHEMA
- It is the distribution of the data into the physical devices.
- It is strongly related to performances and safety

This three level architecture allows the independency of one layer w.r.t. changes in another layer, and this is made possible by changing the mappings between the levels.

Examples:
1. Due to an increase of the database size we need to change the architecture of physical devices, and therefore change the internal schema, we can simply change the mapping between the internal and the logical level, and nothing will change at the logical and external level
2. Due to a new class of users, we need to create a new external schema, and possibly extend the logical schema, then we will only introduce a new mapping between the logical and the external level, and perhaps extend the mapping between the logical and physical level, but nothing will change for the other external schemas

Stability of the external schemas is crucial, because usually the external schema is the basis for application programs, web applications, API's, and a change on the external schema implies a lot of maintenance on such applications.

2.5
What is the difference between logical data independence and physical data independence? Which one is harder to achieve? Why?

SOLUTION

PHYSICAL DATA INDEPENDENCE:

- Separates conceptual levels from the internal/physical levels.
- ***It is easy to achieve physical data independence***.
- With this type of independence, user is able to change the physical storage structures or the devices which have an effect on the conceptual schema.
- Any change at the physical level, does not require to change at the application level.
- It is concerned with the internal schema.

E.g.:

1. Modifying the file organization technique in the Database
2. Switching to different data structures.
3. Changing the access method.
4. Modifying indexes.
5. To change the compression techniques or hashing algorithms.
6. To change the Location of Database from say C drive to D Drive

LOGICAL DATE INDEPENDENCE

- Change the conceptual scheme without changing the following things :
    - External views
    - External API or programs
- It is concerned with the conceptual schema.

E.g.:

1. To Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs.
2. Merging two records into one
3. To break an existing record i.e. to divide the record into two or more records

2.6
What is the difference between procedural and nonprocedural DMLs?

SOLUTION

NON PROCEDURAL DML
- It is also known as high level Data Manipulation language.
- It is used to specify complex database operations.
- We can enter these high level DML statements from a display monitor with the help of Database Management Systems or these statements can also be entered through a terminal. We can also embed these high level DML statements in a programming language.

PROCEDURAL DML
- It is also known as low level DML.
- It is used to get data or objects from the database.
- It processes each operation separately. That's why it has to use programming language constructs to get a record or to process each record from a set of records. Because of this property low level DML is also called set at a time or set oriented DMLs.

Low level and high level DMLs are considered as part of the query language because both languages may be used interactively. Normally casual database (end) users use a nonprocedural language.

2.14
If you were designing a Web-based system to make airline reservations and sell airline tickets, which DBMS architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

SOLUTION

CHOICE
Three-tier architecture.

REASON
1. Client is actually a GUI that the user interacts.
2. The data about the airline reservations is held in a database that the web server that the GUI lives on interacts with. The web interface calls upon functions from the web server, but the application logic and the GUI do not have to live on the same machine.
3. When both the logic and GUI have to live on the same machine, there is a heavy load put on the server which is why the basic client/server architecture and the two-tier client/server architecture aren't the best choices.
4. The centralized DBMS architecture would not work at all because users would not be able to be on the same machine as the database at an industrial scale.

CHAPTER 7

7.7
What is a participation role? When is it necessary to use role names in the description of relationship types?

SOLUTION

PARTICIPATION ROLE

- Part of relationship in which each entity participates in a relationship.
- It is necessary to use role name in the description of relationship type when the same entity type participates more than once in a relationship type in different roles.
- Each entity type that participates in a relationship type plays a particular role in the relationship.

Participation Role or Role name signifies role that a participating entity from the entity type plays in each relationship instance, and helps to explain what relationship means.

E.g.:
In WORKS_FOR relationship type,
EMPLOYE plays the role of worker and
DEPARTMENT plays role of department or employer.

7.11
What is meant by a recursive relationship type? Give some examples of recursive relationship types.

SOLUTION

RECURSIVE RELATIONSHIP TYPE

- A relationship between two entities of similar entity type is called a recursive relationship.
- This means that the relationship is between different instances of the same entity type.

E.g.:
An employee can supervise multiple employees.

7.13
Can an identifying relationship of a weak entity type be of a degree greater than two? Give examples to illustrate your answer.

SOLUTION

An identifying relationship of a weak entity type **can be of a degree greater than two**.

For a ternary identifying relationship type, the weak entity type has several owner entity types.

E.g.:

1. An example of a ternary relationship such as Supply must be represented as a weak entity type with no partial key and with three identifying relationships.
2. These participating entity types Supplier, Part, and Project are together the owner entity types.

Hence an entity in the weak entity type supply is identified by the combination of its three owner entities from Supplier, Part and Project.

7.17

Composite and multivalued attributes can be nested to any number of levels. Suppose we want to design an attribute for a STUDENT entity type to keep track of previous college education. Such an attribute will have one entry for each college previously attended, and each such entry will be composed of college name, start and end dates, degree entries (degrees awarded at that college, if any), and transcript entries (courses completed at that college, if any). Each degree entry contains the degree name and the month and year the degree was awarded, and each transcript entry contains a course name, semester, year, and grade. Design an attribute to hold this information. Use the conventions in Figure 7.5.

SOLUTION

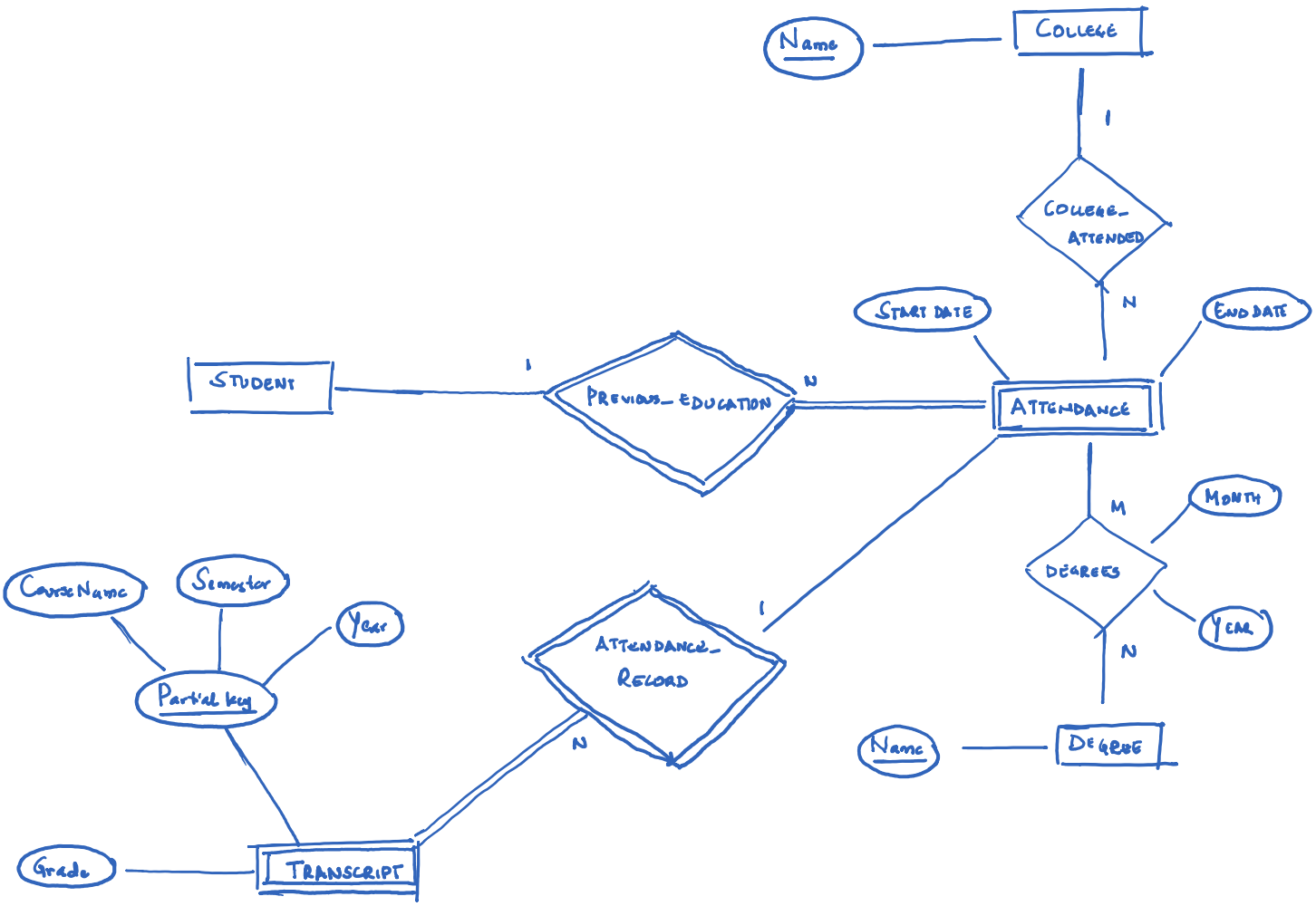
```
{
        PreviousEducation
                ( CollegeName, StartDate, EndDate,
                        { Degree
                                (DegreeName, Month, Year)
                        },
                        { Transcript
                                (CourseName, Semester, Year, Grade)
                        }
                )
}
```

7.18

Show an alternative design for the attribute described in Exercise 7.17 that uses only entity types (including weak entity types, if needed) and relationship types

SOLUTION

- A weak entity type ATTENDANCE; each (weak) entity in ATTENDANCE represents a period in which a STUDENT attended a particular COLLEGE, and is identified by the STUDENT and the StartDate of the period.
- Hence, the StartDate attribute is the partial key of ATTENDANCE.
- Each ATTENDANCE entity is related to one COLLEGE and zero or more DEGREEs (the degrees awarded during that attendance period).
- The TRANSCRIPT of the STUDENT during each attendance period is modelled as a weak entity type, which gives the records of the student during the attendance period.
- Each (weak) entity in TRANSCRIPT gives the record of the student in one course during the attendance period, as shown in the ER diagram below.



7.19

Consider the ER diagram in Figure 7.20, which shows a simplified schema for an airline reservations system. Extract from the ER diagram the requirements and constraints that produced this schema. Try to be as precise as possible in your requirements and constraints specification.

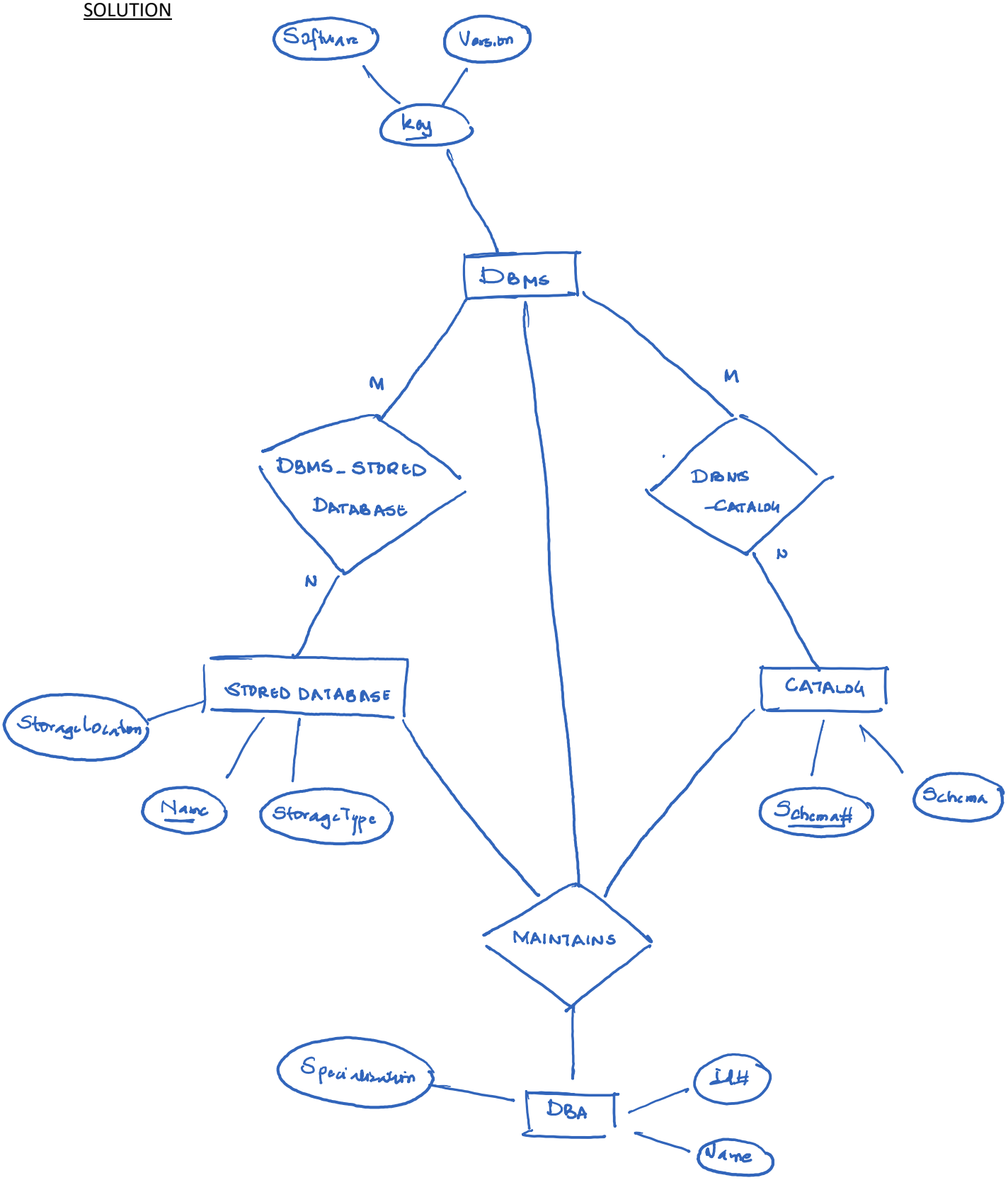
SOLUTION

1. AIRPORT: unique AirportCode, the AIRPORT Name, and the City and State in which the AIRPORT is located.
2. FLIGHT: has a unique number, the Airline for the FLIGHT, and the days on which the FLIGHT is scheduled
3. A FLIGHT is composed of one or more FLIGHT LEGs. Each FLIGHT LEG has a DEPARTURE AIRPORT and Scheduled Departure Time, and an ARRIVAL AIRPORT and Scheduled Arrival Time.
4. RESERVATIONs on each LEG INSTANCE include the Customer Name, Phone, and Seat Number(s) for each reservation.
5. Information on AIRPLANEs and AIRPLANE TYPEs are also kept. For each AIRPLANE TYPE (for example, DC-10), the TypeName, manufacturing Company, and Maximum Number of Seats are kept. The AIRPORTs in which planes of this type CAN LAND are kept in the database. For each AIRPLANE, the AirplaneId, Total number of seats, and TYPE are kept.

7.20

In Chapters 1 and 2, we discussed the database environment and database users. We can consider many entity types to describe such an environment, such as DBMS, stored database, DBA, and catalog/data dictionary. Try to specify all the entity types that can fully describe a database system and its environment; then specify the relationship types among them, and draw an ER diagram to describe such a general database environment

SOLUTION



7.22

A database is being constructed to keep track of the teams and games of a sports league. A team has a number of players, not all of whom participate in each game. It is desired to keep track of the players participating in each game for each team, the positions they played in that game, and the result of
the game. Design an ER schema diagram for this application, stating any assumptions you make. Choose your favourite sport (e.g., soccer, baseball, football).

SOLUTION

The following design may be used for a baseball league.

- Each game in the schedule is identified by a unique Game#, and a game is also identified uniquely by the combination of Date, starting Time, and Field where it is played.
- The Performance attribute of PARTICIPATE is used to store information on the individual performance of each player in a game. This attribute can be designed to keep the information needed for statistics, and may be quite complex. One possible design for the Performance attribute may be the following

```
Performance( {
        Hitting(AtBat#, Inning#, HitType, Runs, RunsBattedIn, StolenBases)
    }, {
        Pitching(Inning#, Hits, Runs, EarnedRuns, StrikeOuts, Walks, Outs, Balks, WildPitches)
    }, {
        Defense(Inning#, {
                FieldingRecord(Position, PutOuts, Assists, Errors)
            }
        )
    } )
```

Performance is a composite attribute made up of three multivalued components:

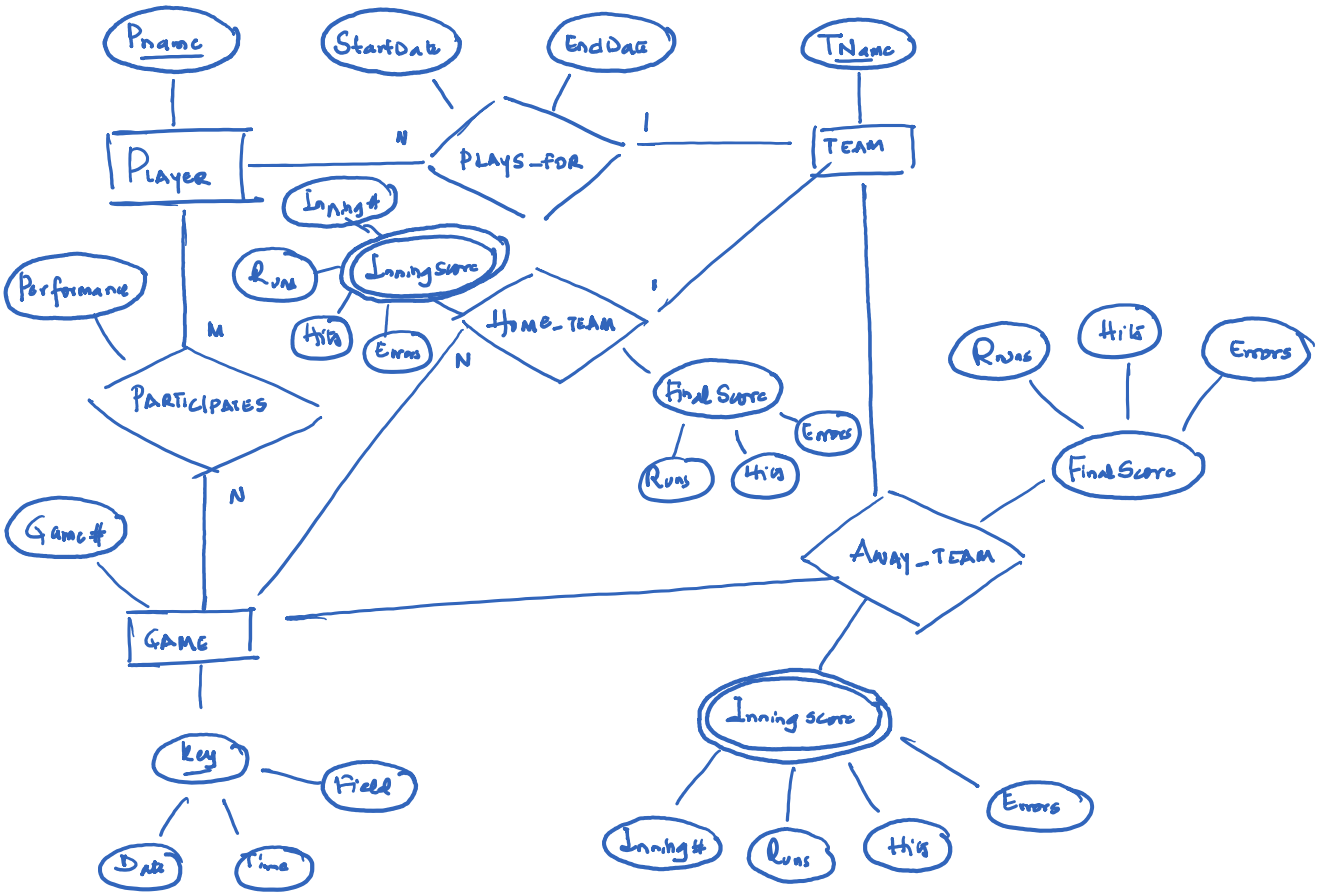
1. Hitting,
2. Pitching,
3. Defense.

Hitting has a value for each AtBat of a player, and records the HitType (suitable coded; for example, 1 for single, 2 for double, 3 for triple, 4 for home run, 0 for walk, -1 for strikeout, -2 for fly out, ...) and other information concerning the AtBat.

Pitching has a value for each inning during which the player pitched.

Defense has a value for each inning a player played a fielding position.

We can have a less detailed or a more detailed design for the performance of a player in each game, depending on how much information we need to keep in the database.


Pname
StartDate
EndDate
TName
Player
N
PLAYS_FOR
1
TEAM
Inning#
Runs
Inning Score
Performance
Hits
Errors
HOME_TEAM
1
M
N
PARTICIPATES
Final Score
Errors
Runs
Hits
Runs
Hits
Errors
Final Score
Game#
N
AWAY_TEAM
GAME
Inning Score
key
Field
Date
Time
Inning#
Runs
Hits
Errors

7.23
Consider the ER diagram shown in Figure 7.21 for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.
a. List the strong (nonweak) entity types in the ER diagram.
b. Is there a weak entity type? If so, give its name, partial key, and identifying relationship.
c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?
d. List the names of all relationship types, and specify the (min, max) constraint on each participation of an entity type in a relationship type. Justify your choices.
e. List concisely the user requirements that led to this ER schema design.
f. Suppose that every customer must have at least one account but is restricted to at most two loans at a time, and that a bank branch cannot have more than 1,000 loans. How does this show up on the (min, max) constraints?

SOLUTION

(a) Entity types: BANK, ACCOUNT, CUSTOMER, LOAN

(b) Weak entity type: BANK-BRANCH, Partial key: BranchNo., Identifying relationship: BRANCHES.

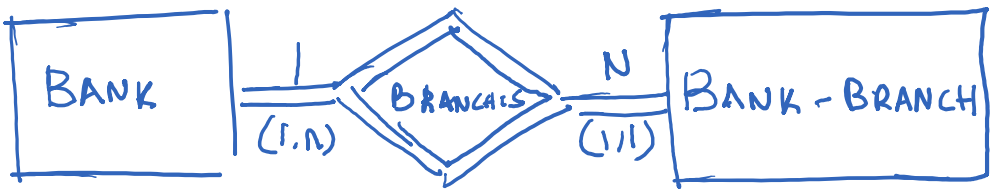
(c)
- The partial key BranchNo in BANK-BRANCH specifies that the same BranchNo value may occur under different BANKs.
- The identifying relationship BRANCHES specifies that BranchNo values are uniquely assigned for those BANK-BRANCH entities that are related to the same BANK entity.

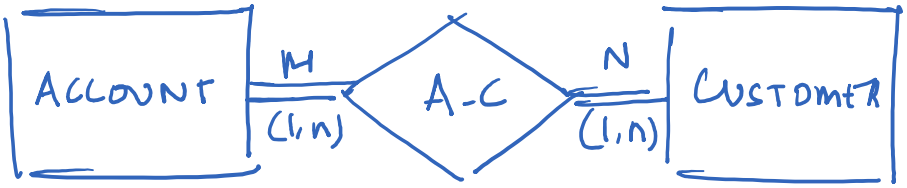
Hence, the combination of BANK Code and BranchNo together constitute a full identifier for a BANK-BRANCH.

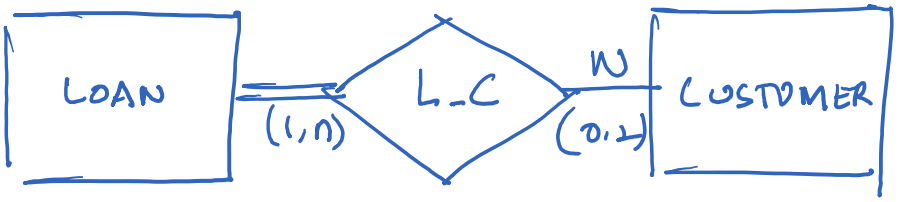
(d) Relationship Types: BRANCHES, ACCTS, LOANS, A-C, L-C. The (min, max) constraints are shown below.

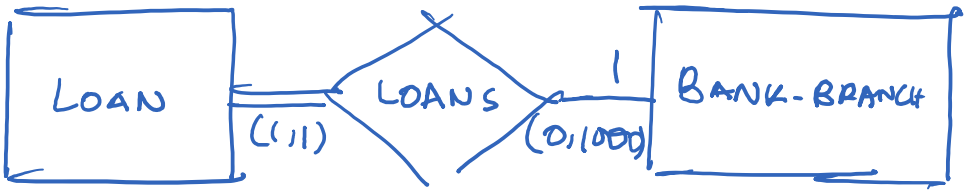
(e) REQUIREMENTS:
- Each BANK has a unique Code, as well as a Name and Address.
- Each BANK is related to one or more BANK-BRANCHes, and the BranhNo is unique among each set of BANK-BRANCHes that are related to the same BANK.
- Each BANK-BRANCH has an Address.
- Each BANK-BRANCH has zero or more LOANS and zero or more ACCTS.
- Each ACCOUNT has an AcctNo (unique), Balance, and Type and is related to exactly one BANK-BRANCH and to at least one CUSTOMER.
- Each LOAN has a LoanNo (unique), Amount, and Type and is related to exactly one BANK-BRANCH and to at least one CUSTOMER.
- Each CUSTOMER has an SSN (unique), Name, Phone, and Address, and is related to zero or more ACCOUNTs and to zero or more LOANs.

(f) The (min, max) constraints would be changed as follows:


BANK
1
(1,N)
BRANCHES
N
(1,1)
BANK-BRANCH


ACCOUNT
M
(1,n)
A-C
N
(1,n)
CUSTOMER


LOAN
(1,n)
L-C
N
(0,1)
CUSTOMER


LOAN
(1,1)
LOANS
1
(0,1000)
BANK-BRANCH

7.27

Cardinality ratios often dictate the detailed design of a database. The cardinality ratio depends on the real-world meaning of the entity types involved and is defined by the specific application. For the following binary relationships, suggest cardinality ratios based on the common-sense meaning of the entity types. Clearly state any assumptions you make.

SOLUTION

	Entity 1	Cardinality Ratio	Entity 2
1.	Student	1-many A student may have more than one social security card (legally with the same unique social security number), and every social security number belongs to a unique student.	SocialSecurityCard
2.	Student	Many-many Generally students are taught by many teachers and a teacher teaches many students.	Teacher
3.	ClassRoom	Many-many Don't forget that the wall is usually shared by adjacent rooms.	Wall
4.	Country	1-1 Assuming a normal country under normal circumstances having one president at a time.	CurrentPresident
5.	Course	Many-many A course may have many textbooks and a text book may be prescribed for different courses.	TextBook
6.	Item (that can be found in an order)	Many-many Assuming the same item can appear in different orders.	Order
7.	Student	Many-many One student may take several classes. Every class usually has several students.	Class
8.	Class	Many-to-1 Assuming that every class has a unique instructor. In case instructors were allowed to team teach, this will be many-many.	Instructor
9.	Instructor	1-1 Assuming every instructor has only one office and it is not shared. In case of offices shared by 2 instructors, the relationship will be 2-1. Conversely, if an instructor has a joint appointment (in two departments) and offices in both departments, then the relationship will be 1-2. In a very general case, it may be many-many.	Office
10.	E-bay Auction item	1-many 1 item has many bids and a bid is unique to an item (assuming a regular auction format).	E-bay bid

7.28
Consider the ER schema for the MOVIES database in Figure 7.24. Assume that MOVIES is a populated database. ACTOR is used as a generic term and includes actresses. Given the constraints shown in the ER schema, respond to the following statements with True, False, or Maybe. Assign a response of Maybe to statements that, while not explicitly shown to be True, cannot be proven False based on the schema as shown. Justify each answer.

SOLUTION

a. There are no actors in this database that have been in no movies.
TRUE

b. There are some actors who have acted in more than ten movies.
MAYBE

c. Some actors have done a lead role in multiple movies.
TRUE

d. A movie can have only a maximum of two lead actors.
TRUE

e. Every director has been an actor in some movie.
FALSE

f. No producer has ever been an actor.
FALSE

g. A producer cannot be an actor in some other movie.
FALSE

h. There are movies with more than a dozen actors.
MAYBE

i. Some producers have been a director as well.
TRUE

j. Most movies have one director and one producer.
MAYBE

k. Some movies have one director but several producers.
TRUE

l. There are some actors who have done a lead role, directed a movie, and produced some movie.
FALSE

m. No movie has a director who also acted in that movie.
FALSE

CHAPTER 8

8.2
Define the following terms: superclass of a subclass, superclass/subclass relationship, IS-A relationship, specialization, generalization, category, specific (local) attributes, and specific relationships.

SOLUTION [from Textbook]

Superclass of a subclass
- A meaningful subgrouping of entities of a particular type.
- The 'parent' entity type to which a subclass belongs to is called a superclass.

Superclass/subclass relationship
- Entities of the subclass inherit the attribute(s) and relationships of the superclass
- A member entity of the subclass represents the same real-world entity as some member of the superclass, but in a distinct specific role
- An entity cannot exist in the DB merely by being a member of a subclass, it must also be a member of a superclass

IS-A relationship
- A class/subclass relationship is often called an IS-A (or IS-AN) relationship because of the way we refer to the concept.
- We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE

Specialization
- Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.
- The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass.

For example, the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the job type of each employee entity.

Generalization
- It is the inverse of specialization
- The process of defining a generalized entity type from the given entity types
- A reverse process of abstraction in which we supress the differences among several entity types, identify their common features, and generalize them into a single superclass ["generalized superclass"] of which the original entity types are special subclasses

Category
A subclass that represents a collection of entities that is a subset of the UNION of entities from distinct entity types

Specific attributes
- Attributes that apply only to entities of a particular subclass—such as TypingSpeed of SECRETARY—are attached to the rectangle representing that subclass.
- These are called specific attributes

Specific relationships
Relationships that are specific to a subclass of a superclass

8.3
Discuss the mechanism of attribute/relationship inheritance. Why is it useful?

SOLUTION

- The type of each entity is defined by the set of attributes and the relationship types.
- The members of the subclass entity inherit the attributes and the relationships of the superclass entity.
- This mechanism is useful because, the attributes in the subclass possess the characteristics of the superclass

8.5
Discuss user-defined and attribute-defined specializations, and identify the differences between the two.

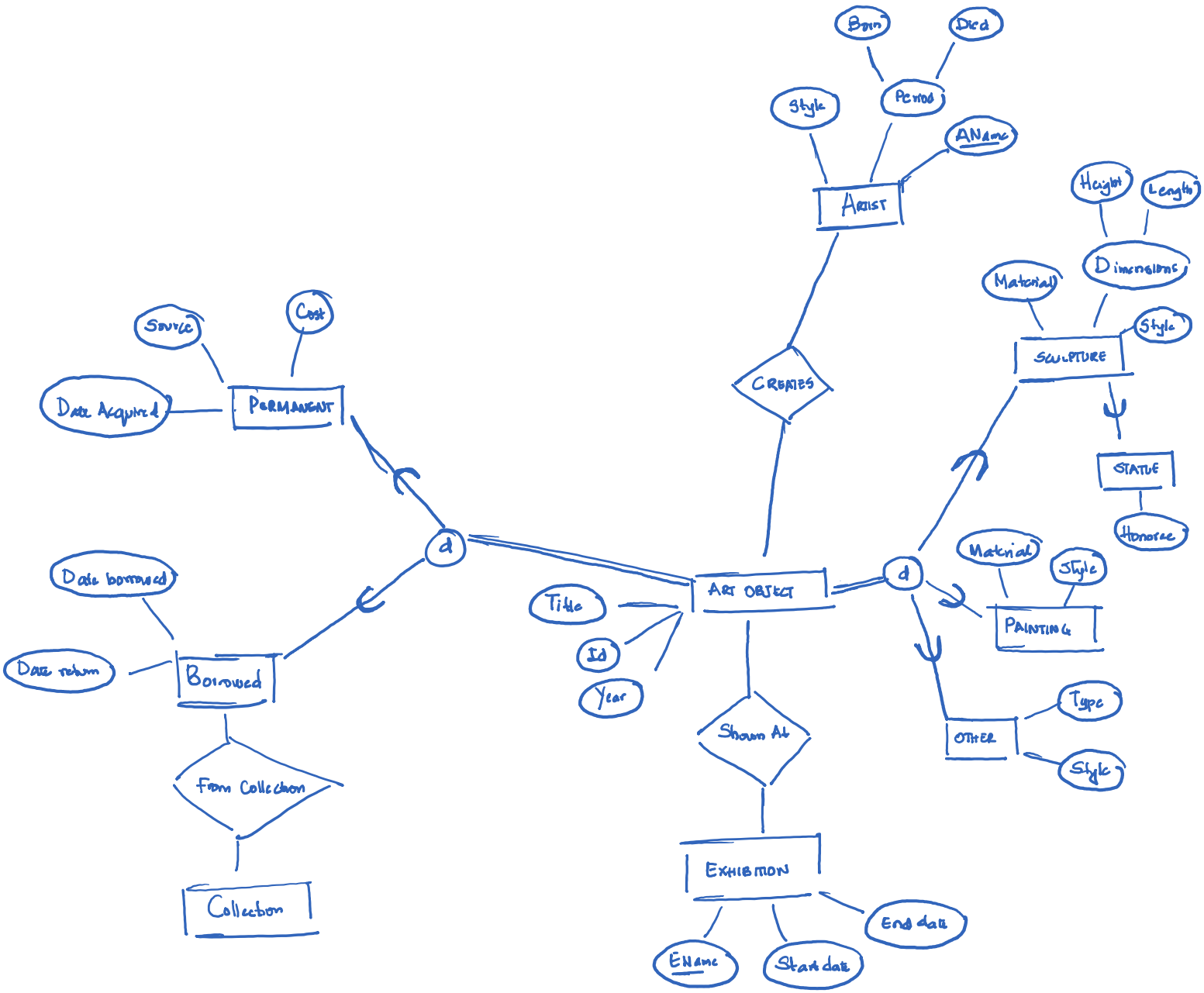
SOLUTION

USER DEFINED SPECIALIZATION	ATTRIBUTE DEFINED SPECIALIZATION
The user is responsible for identifying proper subclass	The value of the same attribute is used in defining predicate for all subclasses
Membership of user-defined defined specialization cannot be decided automatically	Membership of attribute defined specialization can be decided automatically

8.19

Identify all the important concepts represented in the library database case study described below. In particular, identify the abstractions of classification (entity types and relationship types), aggregation, identification, and specialization/generalization. Specify (min, max) cardinality constraints whenever possible. List details that will affect the eventual design but that have no bearing on the conceptual design. List the semantic constraints separately. Draw an EER diagram of the library database.

SOLUTION



8.26

Which of the following EER diagrams is/are incorrect and why? State clearly any assumptions you make.

SOLUTION

Only c is incorrect

Clearly the picture has two subclass entities connected with o in a circle and there is no superclass