CSL302 - Database Engineering

Unit 1

Introduction to DBMS

Data

- Raw, unorganized facts that need to be processed.
- Data can be something simple and seemingly random and useless until it is organized.

- 19UCS19
- Sachin
- 7.8
- Anand
- Chidambaram
- Nirmal

Information

- When data is processed, organized, structured or presented in a given context
- so as to make it useful, it is called information.

PRN	Name of Student	CGPA	Attendance %	Address
19UCS19	Sachin	7.8	80	101, Ichalkaranji

Data Vs Information

PRN	Name of Student	CGPA	Attendance %	Address
19UCS19	Sachin	7.8	80	101, Ichalkaranji

- Scenario 1
- Faculty want to find ‘Who is topper in the class’
- PRN, Name of Student and CGPA
- Scenario 2
- Clerk want to send Attendance Letter
- PRN, Name of Student, Attendance and Address

Introduction

- Database
- collection of related data

- Database application
- A program that interacts with the database at some point in its execution.

Database Management System (DBMS)

- Software that manages and controls access to the database.
- Collection of interrelated data (database) and set of programs to access those data.

Primary Goals

- Store and retrieve data in convenient and efficient manner
- Define structure of information
- Mechanism for manipulation of data
- Safety against system crash and unauthorized access
- Avoid anomalous result while sharing

Database System Applications

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Purpose of Database System

Database Systems Versus Traditional File based Systems

- In Early days, database applications were built on top of file systems
- It suffered from lot of limitations

Example – College Database

- Department
  - PRN, NameOfStudent, Class, Attendance, Address, MobileNum
- Library
  - PRN, NameOfStudent, Class, BooksIssued, Address, MobileNum
- Office
  - PRN, NameOfStudent, Class, FeePaid, FeeDue, Address, MobileNum
- Exam
  - ExamNo, PRN, NameOfStudent, CGPA, Address, MobileNum

Drawbacks of using file systems to store data

- Data isolation
  - Multiple files and formats
  - Inconvenience in data sharing

- Data redundancy and inconsistency
  - Multiple file formats
  - duplication of information in different files

- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
  - E.g. two people reading a balance and updating it at the same time

- Integrity problems
  - Integrity constraints (e.g. account balance < 0) become part of program code
  - Hard to add new constraints or change existing ones

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - E.g. transfer of funds from one account to another should either complete or not happen at all

- Security problems
  - Due to concurrent access and no access control mechanism

Summary

- The limitations of file-based approach can be attributed to two factors:

1. The definition of the data is embedded in the application programs, rather than being stored separately and independently

2. There is no control over the access and manipulation of data beyond that imposed by the application programs.

Database Management System (DBMS)

- DBMS overcome all the limitations of file-based systems.
- In Database Management System, data definition get stored separately and independently from application programs.
- There exist access control mechanism in DBMS to prevent unauthorized access to data.

History of Database Management Systems

- In 1960, Charles W. Bachman designed the Integrated Database System called as Integrated Data Store (IDS), the “first” DBMS.
- It was based on network data model.

- In 1960's, IBM (International Business Machines Corporation) developed the Integrated Management Systems (IMS)
- which is the standard database system used till date in many places.
- It was developed based on the hierarchical database model.

- Both database systems are described as the fore-runners of navigational databases

- By the mid-1960s, as computers developed speed and flexibility,
- and started becoming popular, many kinds of general use database systems became available.
- As a result, customers demanded a standard be developed
- Formed Database Task Group lead by Bachman.
- This group took responsibility for the design and standardization of a language called Common Business Oriented Language (COBOL).

- The Database Task Group presented this standard in 1971, which also came to be known as the "CODASYL approach."
- Conference/Committee on Data Systems Languages
- The CODASYL approach was a very complicated system and required substantial training.
- Eventually, the CODASYL approach lost its popularity as simpler, easier-to-work-with systems came on the market.

- In 1973, Michael Stonebraker and Eugene Wong (both then at UC Berkeley) made the decision to research relational database systems.
- The project was called INGRES (Interactive Graphics and Retrieval System), and successfully demonstrated a relational model could be efficient and practical.
- INGRES worked with a query language known as QUEL

- IBM developed ‘Structured Query Language’ (SQL) in 1974,
- which was more advanced (SQL became ANSI and OSI standards in 1986 and 1987).
- SQL quickly replaced QUEL as the more functional query language.

- The relational model is still in use by many people in the market.

- Then, processing speeds got faster, and “unstructured” data (art, photographs, music, etc.) became much more common place.
- Unstructured data is both non-relational and schema-less, and Relational Database Management Systems simply were not designed to handle this kind of data
- Further, there were many other models with rich features like complex queries, datatypes to insert images and many others.

Commercial DBMS products

- The most prominent commercial DBMS products based on the relational model
- IBM DB2
- Oracle Database
- SQL Server
- MySQL

Example – College Database

- Department
  - PRN, NameOfStudent, Class, Attendance, Address, MobileNum
- Library
  - PRN, NameOfStudent, Class, BooksIssued, Address, MobileNum
- Office
  - PRN, NameOfStudent, Class, FeePaid, FeeDue, Address, MobileNum
- Exam
  - ExamNo, PRN, NameOfStudent, CGPA, Address, MobileNum

Example – College Database

- Department
  - PRN, Attendance
- Library
  - PRN, BooksIssued
- Office
  - PRN, NameOfStudent, Class, FeePaid, FeeDue, Address, MobileNum
- Exam
  - ExamNo, PRN, CGPA

Advantages of DBMS

- Control of data redundancy
  - Traditional file-based systems waste space by storing the same information in more than one file.

Advantages of DBMS

- Data consistency
  - By eliminating redundancy, reduce the risk of inconsistencies occurring.
  - If a data item is stored only once in the database, any update to it has to be performed only once and the new value is available immediately to all.

- More information from the same amount of data
  - With the integration of the operational data, it may be possible for the organization to derive additional information from the same data.

- Sharing of data
  - Database belongs to the entire organization and can be shared by all authorized users.

- Improved data integrity
  - Database integrity refers to the validity and consistency of stored data.
  - Integrity is expressed in terms of constraints
  - Constraints are consistency rules that the database is not permitted to violate.
  - Constraints may apply to data items within a single record or they may apply to relationships between records.

- Improved security
  - Database security is the protection of the database from unauthorized users.

- Enforcement of standards
  - Integration allows the DBA to define and enforce the necessary standards.
  - These may include departmental, organizational, national, or international standards.

- Economy of scale
  - Combining all the organization's operational data into one database and creating a set of applications that work on this one source of data, can result in cost savings.

- Improved data accessibility and responsiveness
  - As a result of integration, data that crosses departmental boundaries is directly accessible to the end-users.
  - This provides a system with potentially much more functionality

- Increased productivity
  - DBMS provides many of the standard functions that the programmer would normally have to write in a file-based application.
  - At a basic level, the DBMS provides all the low-level file-handling routines that are typical in application programs.

- Increased concurrency
  - Many users can access database simultaneously
  - Many DBMS manage concurrent database access and ensure problems cannot occur.

- Improved backup and recovery services
  - Modern DBMSs provide facilities to minimize the amount of processing that is lost due to failure.
  - Take backup regularly.
  - Assure minimum down-time

Disadvantages of DBMS

- Complexity
  - The provision of the functionality expected from a good DBMS makes the DBMS an extremely complex piece of software.
  - Database designers, developers, database administrators, and end-users must understand this functionality to take full advantage of it.
  - Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.

- Memory Size
  - The complexity and breadth of functionality makes the DBMS an extremely large piece of software,
  - occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

- Cost of DBMS
  - The cost of DBMS varies significantly, depending on the environment and functionality provided.
  - A single-user DBMS for a personal computer may only cost US$100.

- Additional hardware costs
  - The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space.
  - To achieve the required performance, it may be necessary to purchase a higher configuration machine.

- Cost of conversion
  - In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on the new DBMS and hardware.
  - This cost also includes the cost of training staff to use these new systems.

- Performance
  - A file-based system is written for a specific application, such as invoicing.
  - As a result, performance is generally very good.
  - However, the DBMS is written to be more general, to cater for many applications rather than just one.
  - The effect is that some applications may not run as fast as they used to.

- Higher impact of a failure
  - The centralization of resources increases the vulnerability of the system.
  - Since all users and applications rely on the availability of the DBMS, the failure of certain components can bring operations to a halt.

View of Data
Levels of Data Abstraction

- Complex data structures used for efficiency
- Hides certain details of how data are stored and maintained
- Provides abstract view
- Hides complexity.

View of Data
Levels of Data Abstraction

- 3 Levels of Data Abstraction
- Physical Level
- Logical Level
- View Level

Physical Level

- Lowest level of abstraction
- Describes how data are actually stored
- Describes complex low-level data structures in detail
- Blocks of consecutive storage locations- words, byte etc
- Administrator may be aware of certain details of the physical organization

Logical Level

- Next higher level of abstraction
- Describes what data are stored in the database
- And what relationship exists among those data
- Describes entire database in terms of a small number of relatively simple structures
- Do not bother with complex physical storage

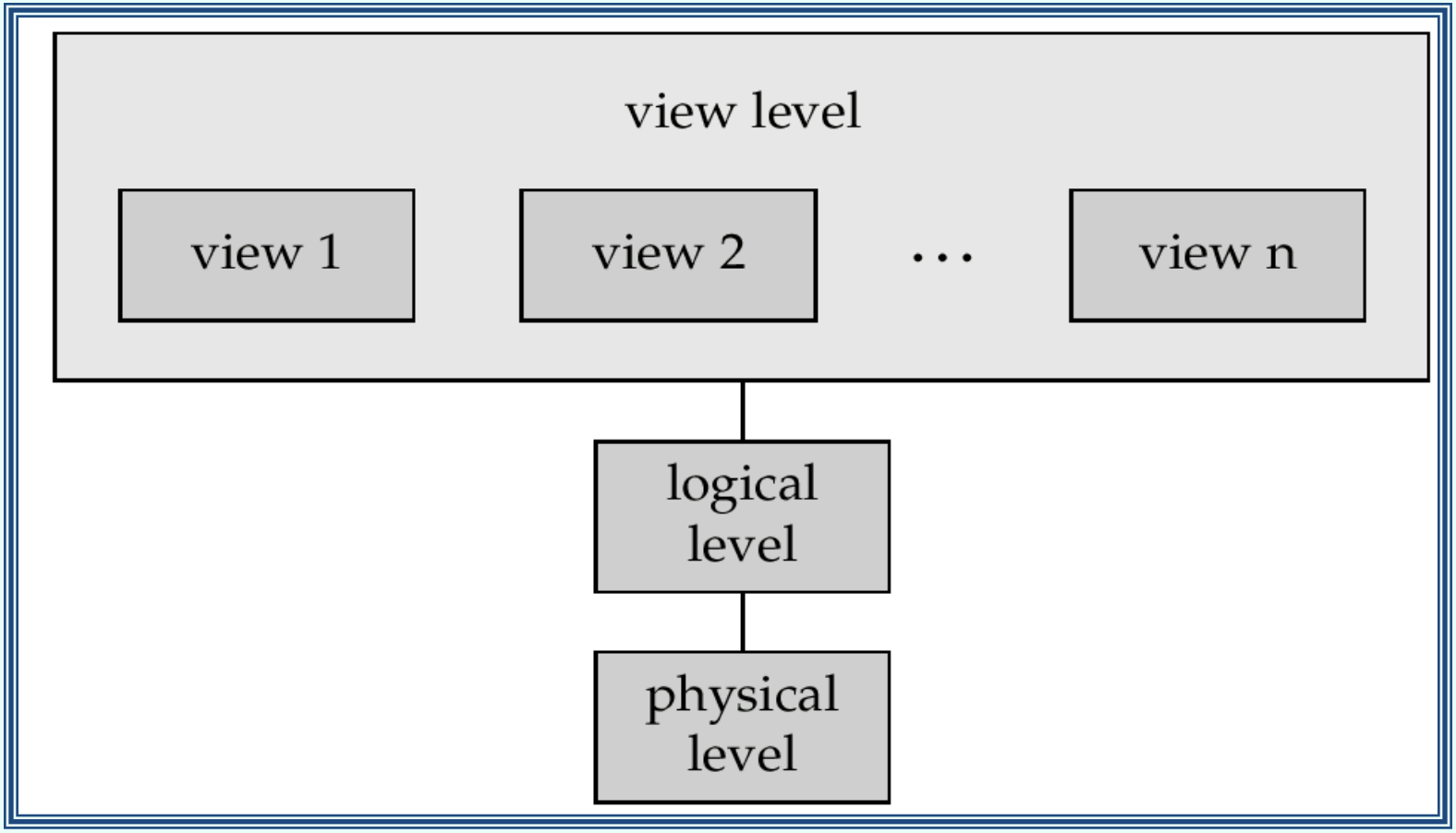
Logical Level

- Database administrator uses logical level
- Who decides what information to keep in database
- Each record is described by a type definition
- Programmers also work at logical level.

View Level

- Highest level of abstraction
- Describes only part of entire database
- Exists to simplify interaction with the system
- System may provide many views for the same database
- Set of application programs that hide details of the data types
- Views also provide security mechanisms to prevent users from accessing certain parts of database

View of Data
Levels of Data Abstraction



Example – College Database

- Department
  - PRN, Attendance
- Library
  - PRN, BooksIssued
- Office
  - PRN, NameOfStudent, Class, FeePaid, FeeDue, Address, MobileNum
- Exam
  - ExamNo, PRN, CGPA

Different Views on College Database

- Departmental Clerk
- Class Teacher
- Accountant
- HOD
- COE

Schema and Instance

- Similar to data types and variables in programming languages

- Schema
  - The logical structure of the database
  - e.g. the database consists of information about a set of customers and accounts and the relationship between them

- Database systems have several schemas
- Partitioned according to levels of abstraction
- Physical Schema –
  - describes database design at physical level
- Logical Schema –
  - describes database design at logical level
- Several schemas at view level, called as sub-schemas
  - Describes different views of database

- Analogous to type information of a variable declared in a program
- The value of the variable in a program at a point in time correspond to an instance.

- In DBMS, Schema is the overall design of the database.
- Instance is the information stored in the database at a particular moment.

Physical Data Independence

- Logical schema is most important.
- Applications depend on the logical schema
- The ability to modify the physical schema without changing the logical schema is called Physical Data Independence
- The interfaces between the various levels and components should be well defined
- so that changes in some parts do not seriously influence others.

Data Models

- Data model describes underlying structure of a database.
- A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

Data Models

- Relational Data Model
- Hierarchical Data Model
- Network Data Model
- Object-Oriented Data Model
- Object-Relational Data Model
- Semi-structured Data Model
- Entity-Relationship Model

Relational Data Model

- Uses a collection of tables to represent both data and the relationships among those data.
- Each table has multiple columns, and each column has a unique name.
- Relational model is an example of a record-based model.
- Record-based models are so named because the database is structured in fixed-format records of several types.
- Each table contains records of a particular type.

- Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type.
- The relational data model is the most widely used data model
- a vast majority of current database systems are based on the relational model.
- The relational model is at a lower level of abstraction than the E-R model.

customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

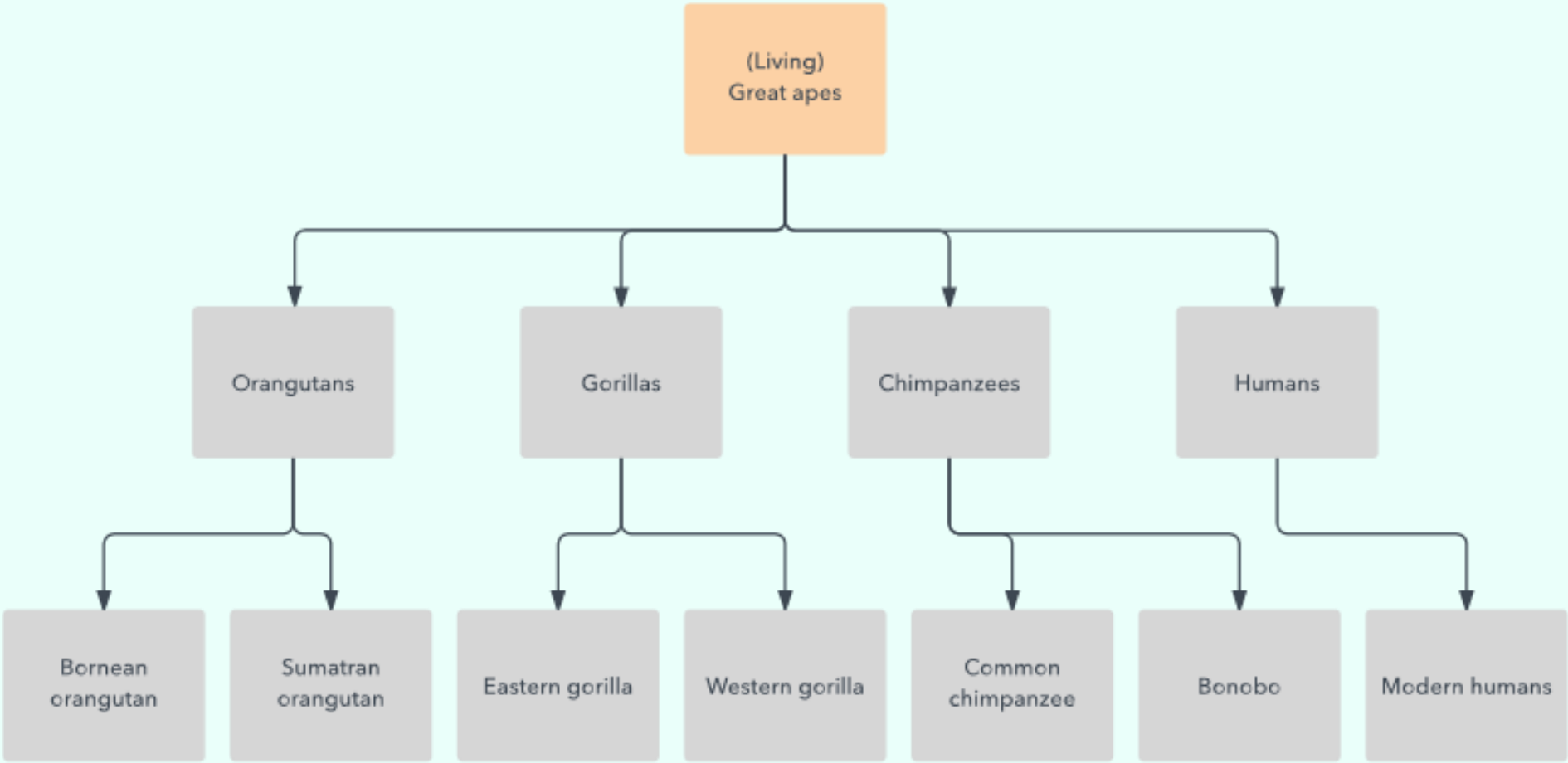
(c) The *depositor* table

A sample relational database.

Hierarchical Data Model

- The hierarchical model organizes data into a tree-like structure, where each record has a single parent or root.
- Sibling records are sorted in a particular order.
- That order is used as the physical order for storing the database.
- This model is good for describing many real-world relationships.

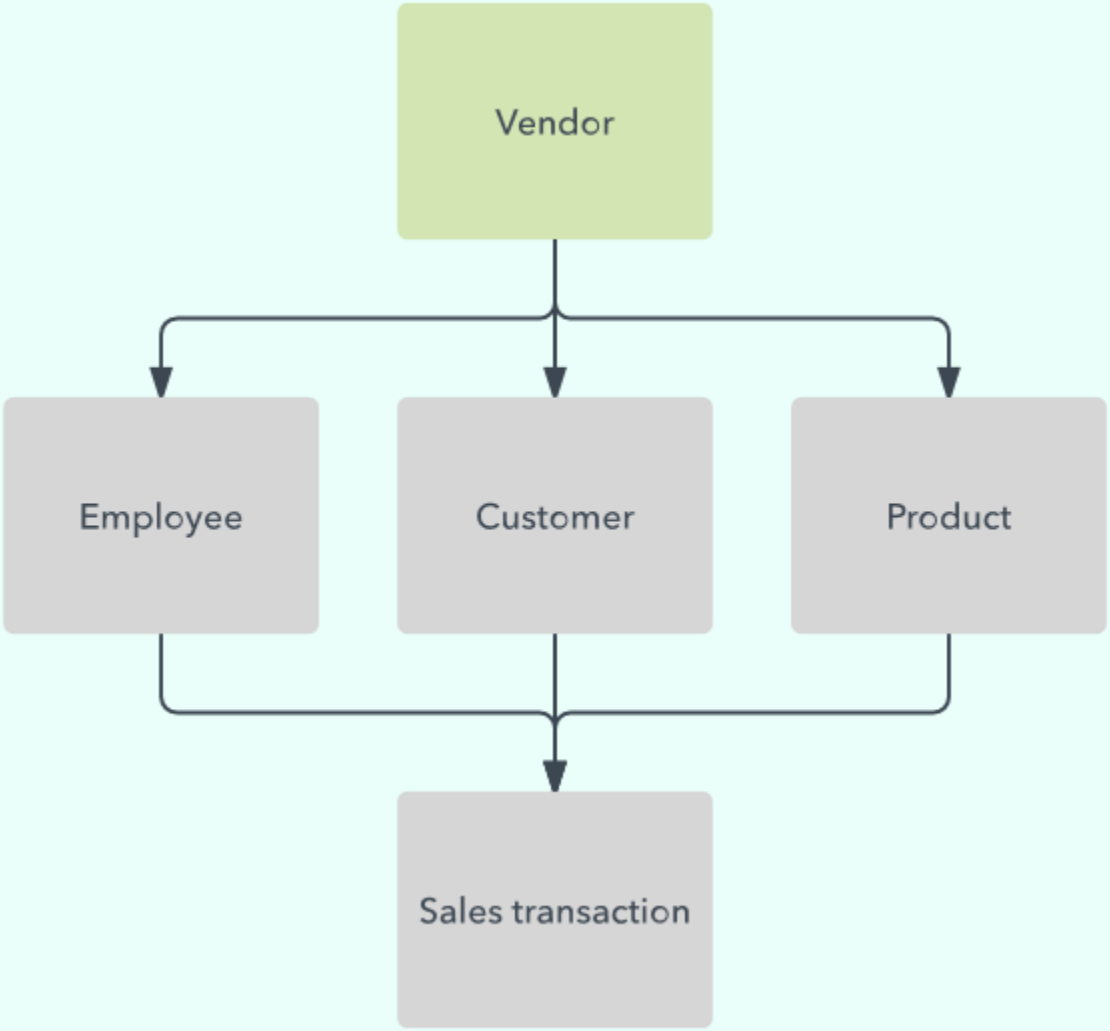
- Hierarchical model was primarily used by IBM's Information Management Systems(IMS) in the 60s and 70s,
- but they are rarely seen today due to certain operational inefficiencies.


(Living)
Great apes
Orangutans
Gorillas
Chimpanzees
Humans
Bornean orangutan
Sumatran orangutan
Eastern gorilla
Western gorilla
Common chimpanzee
Bonobo
Modern humans

Network Data Model

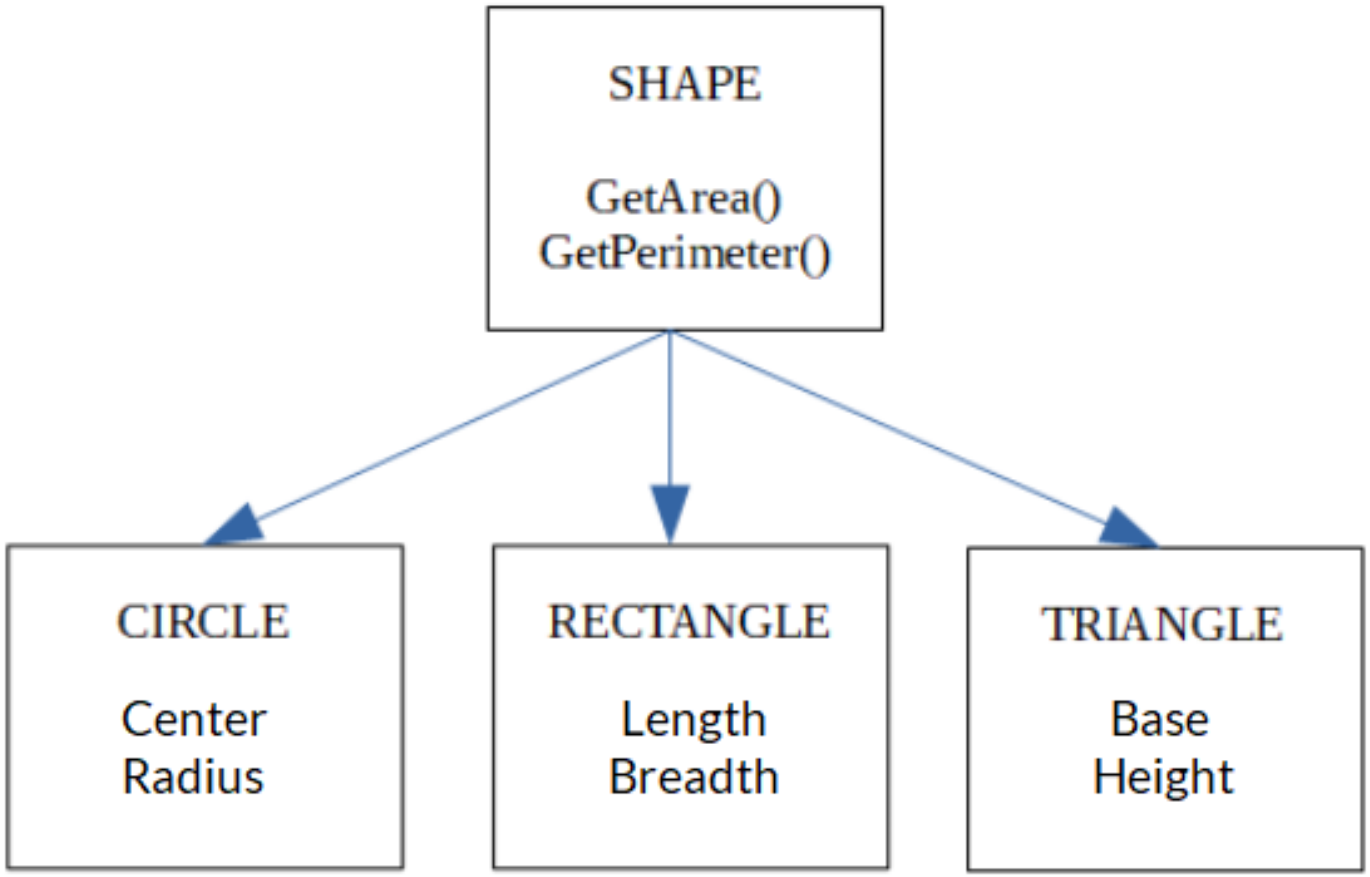
- The network model builds on the hierarchical model by allowing many-to-many relationships between linked records, implying multiple parent records.
- Based on mathematical set theory, the model is constructed with sets of related records.

- Each set consists of one owner or parent record and one or more member or child records.
- A record can be a member or child in multiple sets, allowing this model to convey complex relationships.
- It was most popular in the 70s after it was formally defined by the Conference on Data Systems Languages (CODASYL).


Vendor
Employee
Customer
Product
Sales transaction

Object-oriented Data Model

- This model defines a database as a collection of objects.
- Uses Object-oriented programming language concepts like
- Class
- Object
- Encapsulation
- Inheritance


SHAPE
GetArea()
GetPerimeter()
CIRCLE
Center
Radius
RECTANGLE
Length
Breadth
TRIANGLE
Base
Height

Object-Relational Data Model

- It combines features of the object-oriented data model and relational data model
- It allows designers to incorporate objects into the familiar table structure.

Semi-structured Data Model

- In semi-structured model no separation between the data and the schema, and the structure used depends on the purpose.
- It permit the specification of data where individual data items of the same type may have different sets of attributes.
- This is in contrast with the data models mentioned earlier where every data item of a particular type must have the same set of attributes.

- It provides a flexible format for data exchange between different types of databases.
- The schema can easily be changed.
- The data transfer format may be portable.
- The primary trade-off being made in using a semi-structured database model is that queries cannot be made as efficiently as in a more constrained structure, such as in the relational model.
- The extensible markup language (XML) is widely used to represent semi-structured data.

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Entity-Relationship (E-R) Model

- Entity – Thing or Object in the real world
  - Distinguishable from other objects
  - E. g. person, bank account
- Entities are described in database by a set of attributes
- E. g. Student – PRN, Name, Class, CGPA, Address

- Some attributes used to uniquely identify the entities.
  - PRN
- Some attributes used to group the similar entities in an entity set.
  - Class

- Relationship –
  - Association among several entities
- E. g.
  - Faculty teaches Student
  - Customer has Account
  - Manager manages Company

Entity Set

- Set of entities of the same type that share the same properties, or attributes.
- Student – PRN, Name, Class, CGPA, Address
- Students having same value for attribute ‘Class’ forms an entity set.

- Customers having account at same branch forms an entity set.

- Not all entity sets can be disjoint
  - Person can be Employee
  - Person can be Customer

Relationship Set

- A relationship is an association among several entities.
- A relationship set is a set of relationships of the same type.
- E.g. Customer has Account
- Set of all customers – Entity set
- Set of all accounts – Entity set

- One customer can have only one account
- One customer can have one or more accounts
- Many customers can have a joint account
- Many customers can have disjoint accounts.

- The relationship set ‘has’ represents such association.

Attributes

- An entity is represented by a set of attributes
- Attributes are descriptive properties possessed by each member of an entity set.
- Each entity has a value for each of its attributes
- For each attribute, there is a set of permitted values, called the domain or value set of that attribute.
- An attribute can be characterized by the following attribute types

Simple and Composite Attributes

- Simple attributes
  - They can't be divided into subparts.
- Composite attributes
  - They can be divided into subparts (other attributes).
- E.g. attribute name could be structured as a composite attribute consisting of first-name, middle-name and last-name
- composite attribute may appear as a hierarchy

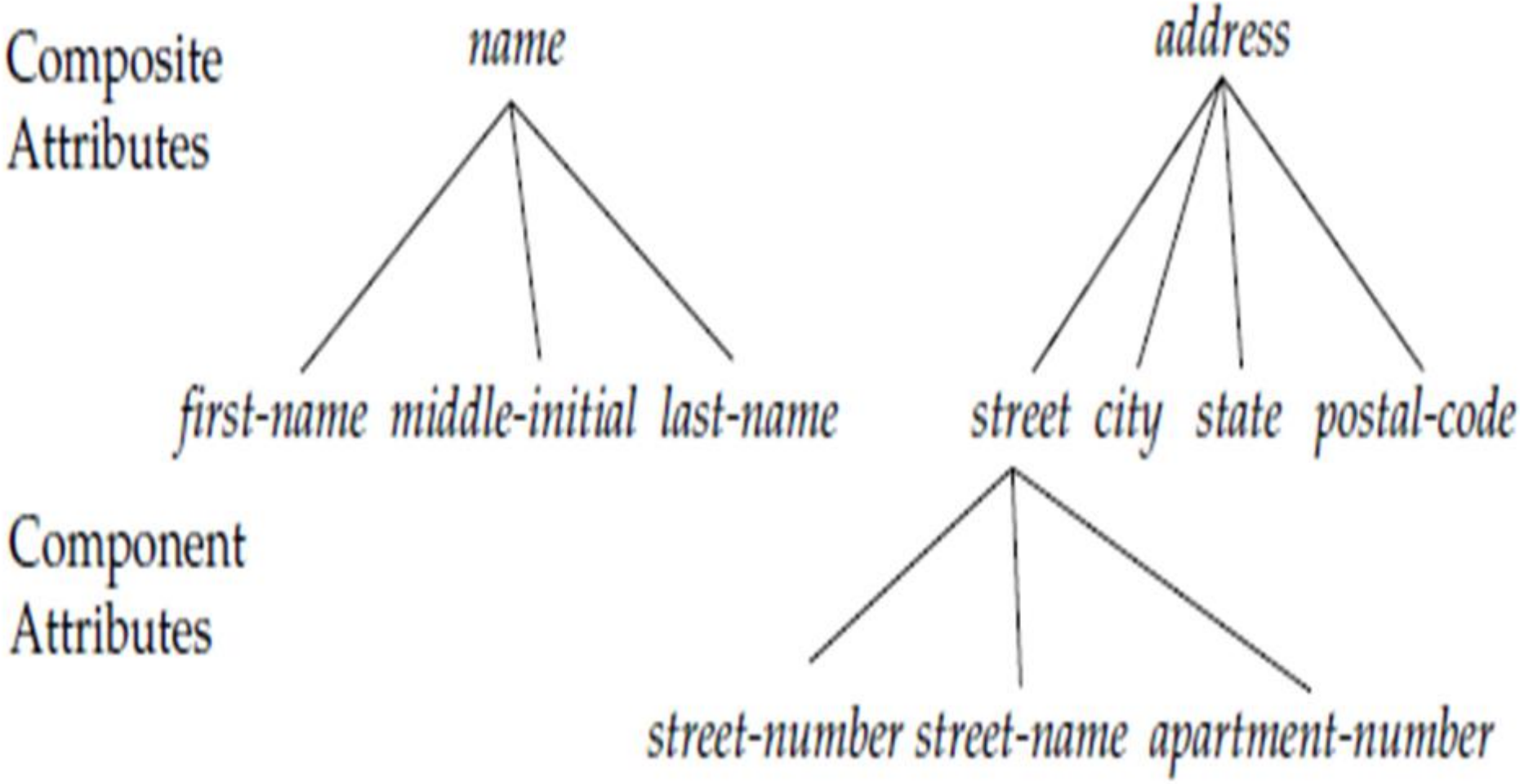


Figure Composite attributes *customer-name* and *customer-address*.

Single-valued and multivalued attributes

- Single-valued attributes
  - always attribute has only one value
- Multi-valued attributes
  - attribute can have more than one value
- E.g. Mobile Number attribute of Person

Derived attribute

- Value of the attribute can be derived from the values of other related attributes
- E.g. Age attribute of person
- The value of a derived attribute is not stored, but is computed when required.
- Age of Person is not stored in database, but can be calculated using date_of_birth and todays_date

- An attribute takes a null value when an entity does not have a value for it.
- The null value may indicate “not applicable”—the value does not exist for the entity.
- For example, one of the person may have no middle name.
- Null can also designate that an attribute value is unknown.
- Null and 0 are not same.
- 0 is a value.

Components of E-R Diagram

- Rectangle – represent entity set
- Ellipse – represent attribute
- Diamond – represent relationship among entity set
- Lines – link attribute to entity set and entity set to relationship
- Double Ellipse – Multivalued Attribute
- Dashed Ellipse – Derived Attribute
- Double lines - indicate total participation of an entity in a relationship set
- Each entity is labeled with name

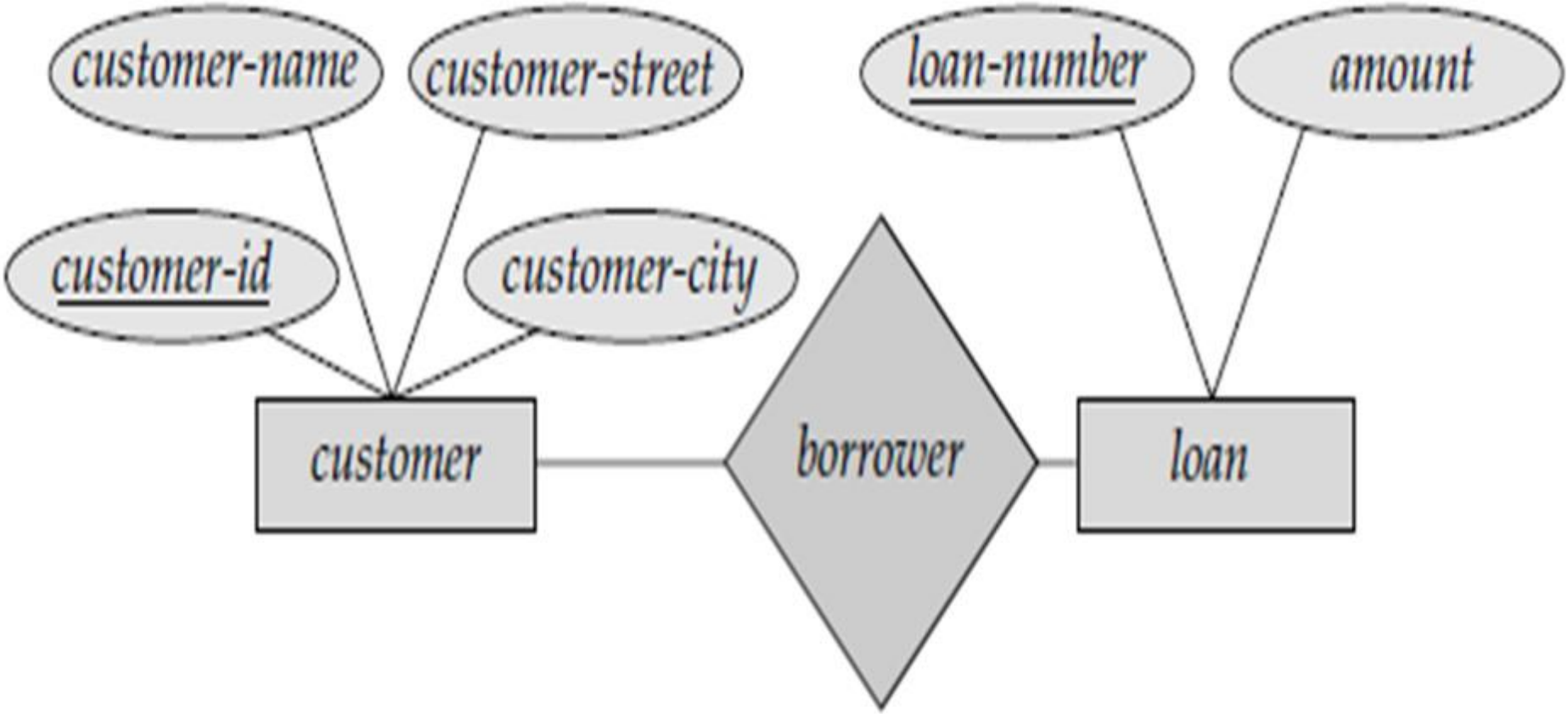


Figure E-R diagram corresponding to customers and loans.

Constraints

- Conditions/Rules to which the contents of a database must conform
- Mapping Cardinalities / Cardinality Ratio
- Participation
- Keys

Mapping Cardinalities / Cardinality Ratios

- Express the number of entities to which other entities can be associated via a relationship set
- E.g. each account must belong to only one customer

- For a binary relationship set R between entity sets A and B,
- the mapping cardinality must be one of the following:

- One to one
- An entity in A is associated with at most one entity in B and
- an entity in B is associated with at most one entity in A.
- E.g. One Customer – one account

- One to many
- An entity in A is associated with any number (zero or more) of entities in B.
- An entity in B, can be associated with at most one entity in A.
- E.g. One Customer – many accounts

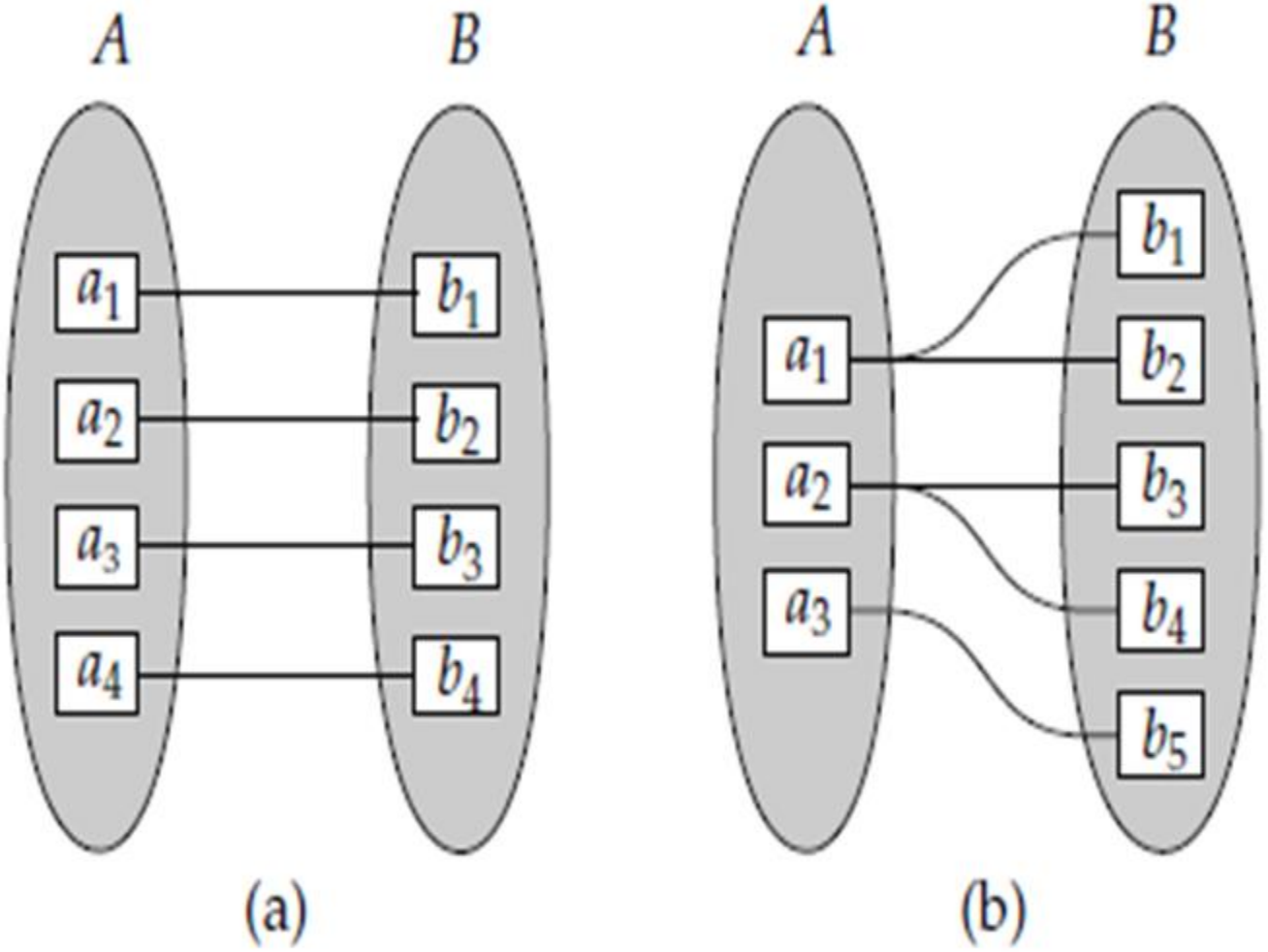


Figure Mapping cardinalities. (a) One to one. (b) One to many.

- Many to one
- An entity in A is associated with at most one entity in B.
- An entity in B, however, can be associated with any number (zero or more) of entities in A.
- E.g. 3 Customers have joint account

- Many to many
- An entity in A is associated with any number (zero or more) of entities in B, and
- an entity in B is associated with any number (zero or more) of entities in A.
- E.g. Many customers many accounts

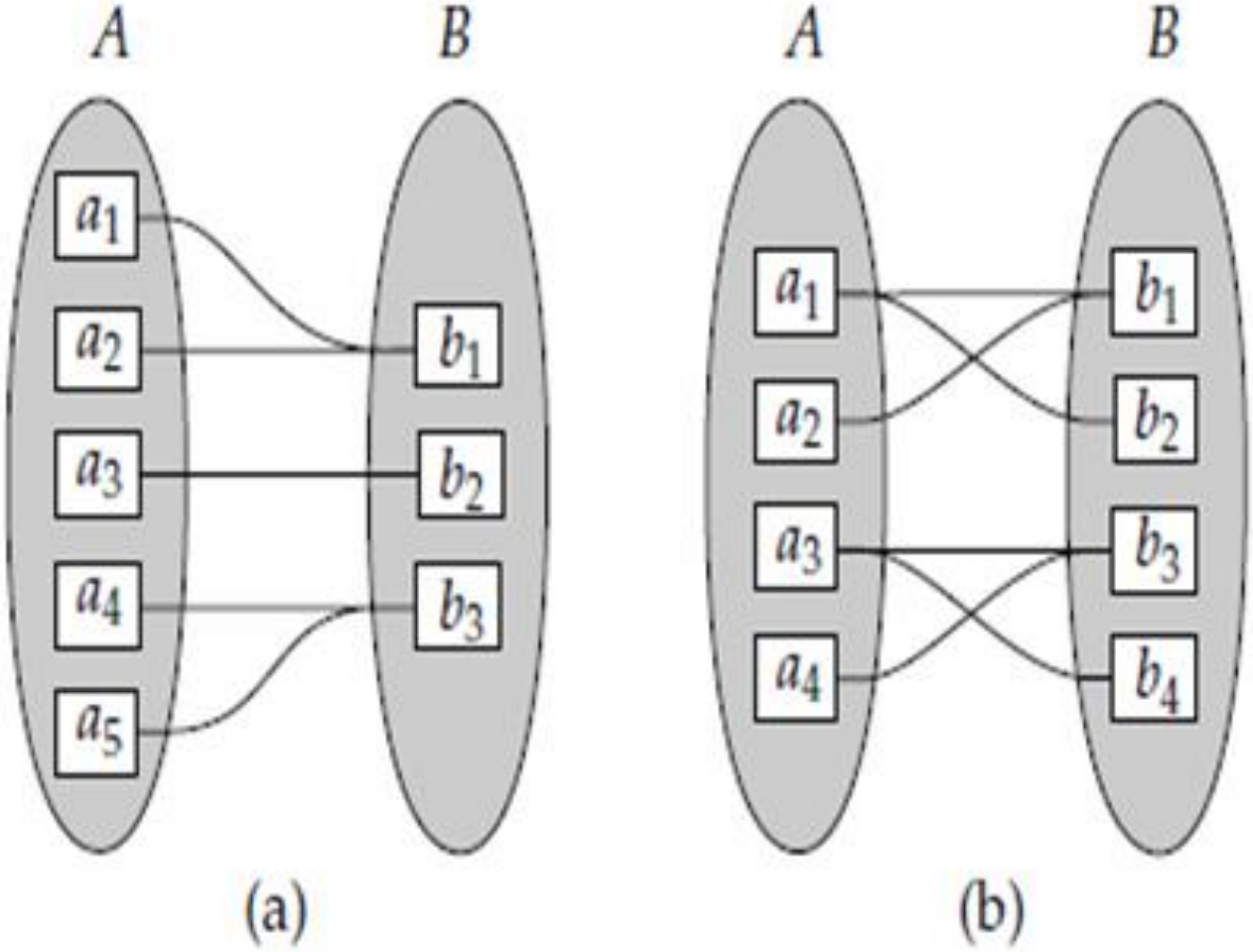


Figure Mapping cardinalities. (a) Many to one. (b) Many to many.

Participation

- The association between entity sets is referred to as participation
- The entity sets E1, E2, . . .,En participate in relationship set R
- The function that an entity plays in a relationship is called that entity's role.

Participation

- Total Participation
- The participation of an entity set E in a relationship set R is said to be total participation
- if every entity in E participates in at least one relationship in R.
- E.g. In Customer has loan_account
- Every loan_account must have at least one customer
- So loan account entity take part in at least one relationship
- So participation of loan_account is total participation

- Partial Participation
- If only some entities in E participate in relationships in R,
- the participation of entity set E in relationship R is said to be partial participation
- E.g. In Customer has loan_account
- Not necessary that, every customer must have loan_account
- So every customer entity may or may not take part in the relationship
- So participation of customer is partial participation

Keys

- It is necessary to distinguish among different entities in an entity set.
- Entity is described by a set of attributes.
- Some of the attributes are used to group similar entities together in an entity set.
- Some of the attributes are used to distinguish entities from each other
- A key allows us to identify a set of attributes that suffice to distinguish entities from each other.

- Super Key
- Candidate Key
- Primary Key

Superkey

- A superkey is a set of one or more attributes that, taken collectively,
- allow us to identify uniquely an entity in the entity set.
- For example- the customer-id attribute of the entity set customer is sufficient to distinguish one customer entity from another.
- Thus, customer-id is a superkey
- PRN is sufficient to distinguish one student from others

- Similarly, the combination of customer-name and customer-id is a superkey for the entity set customer.
- The customer-name attribute of customer is not a superkey, because several people might have the same name.
- PRN and student_name can be used to distinguish one student from others.
- But only Student_Name may not distinguish one student from others.
- Two students may have same Student_Name

- One relation (table) can have one or more superkey.
- One or more attributes can form a superkey.

Candidate Key

- Superkey may contain extraneous attributes.
- If K is a superkey, then so is any superset of K.
- Superkeys for which no proper subset is a superkey.
- Such minimal superkeys are called candidate keys.
- several distinct sets of attributes could serve as a candidate key.

- E.g. Student – PRN, Student_Name, Class, CGPA, Address
- 1] PRN and Student_Name forms a superkey
- 2] Only PRN also forms superkey

- Student_Name is extraneous attribute
- Key1 is not minimal, it can have proper subset
- But key2 is minimal, no proper subset is possible for key2
- Key2 is called candidate key.

- One relation (table) can have one or more candidate keys.
- One or more attributes can form a candidate key.

Primary Key

- Primary key is one of the candidate keys selected by database designer to uniquely identify an entity in entity set.
- E.g. PRN, UID, Class and RollNo.

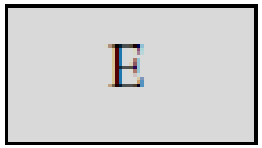
- One relation (table) can have only one primary key.
- One or more attributes can form a primary key.

- A key (primary key, candidate key, and superkey) is a property of the entity set rather than, of the individual entities.
- Any two individual entities in the set are prohibited from having the same value on the key attributes at the same time.
- Candidate keys must be chosen with care

- The primary key should be chosen such that its attributes are never or very rarely, changed.
- For instance, the address field of a student should not be part of the primary key
- since it is likely to change.
- PRN, on the other hand, are guaranteed to never change or rarely change.

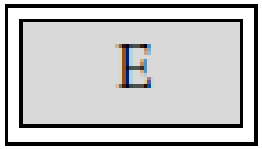
Components of E-R diagram

- Rectangle – represent entity set
- Ellipse – represent attribute
- Diamond – represent relationship among entity set
- Lines – link attribute to entity set and entity set to relationship
- Double Ellipse – Multivalued Attribute
- Dashed Ellipse – Derived Attribute
- Double lines - indicate total participation of an entity in a relationship set
- Each entity is labeled with name



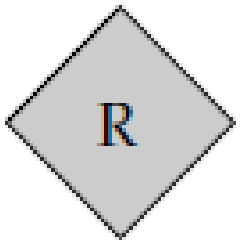
entity set

attribute



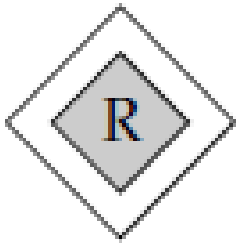
weak entity set

multivalued attribute

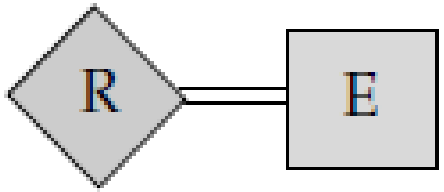


relationship set

derived attribute



identifying relationship set for weak entity set

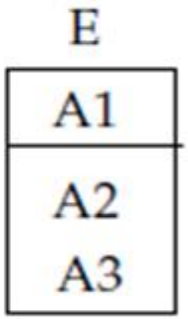


total participation of entity set in relationship

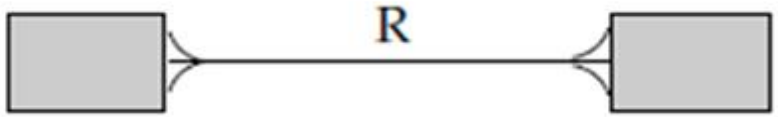
primary key

discriminating attribute of weak entity set

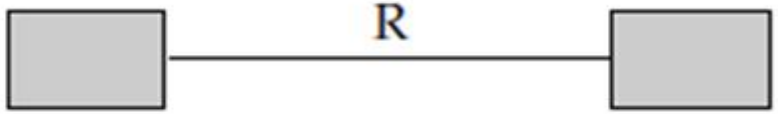
entity set E with
attributes A1, A2, A3
and primary key A1



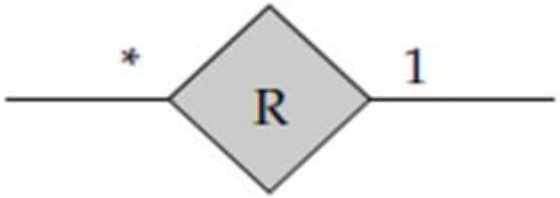
many-to-many
relationship

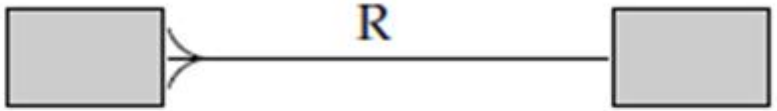


one-to-one
relationship



many-to-one
relationship





Sample E-R Diagrams

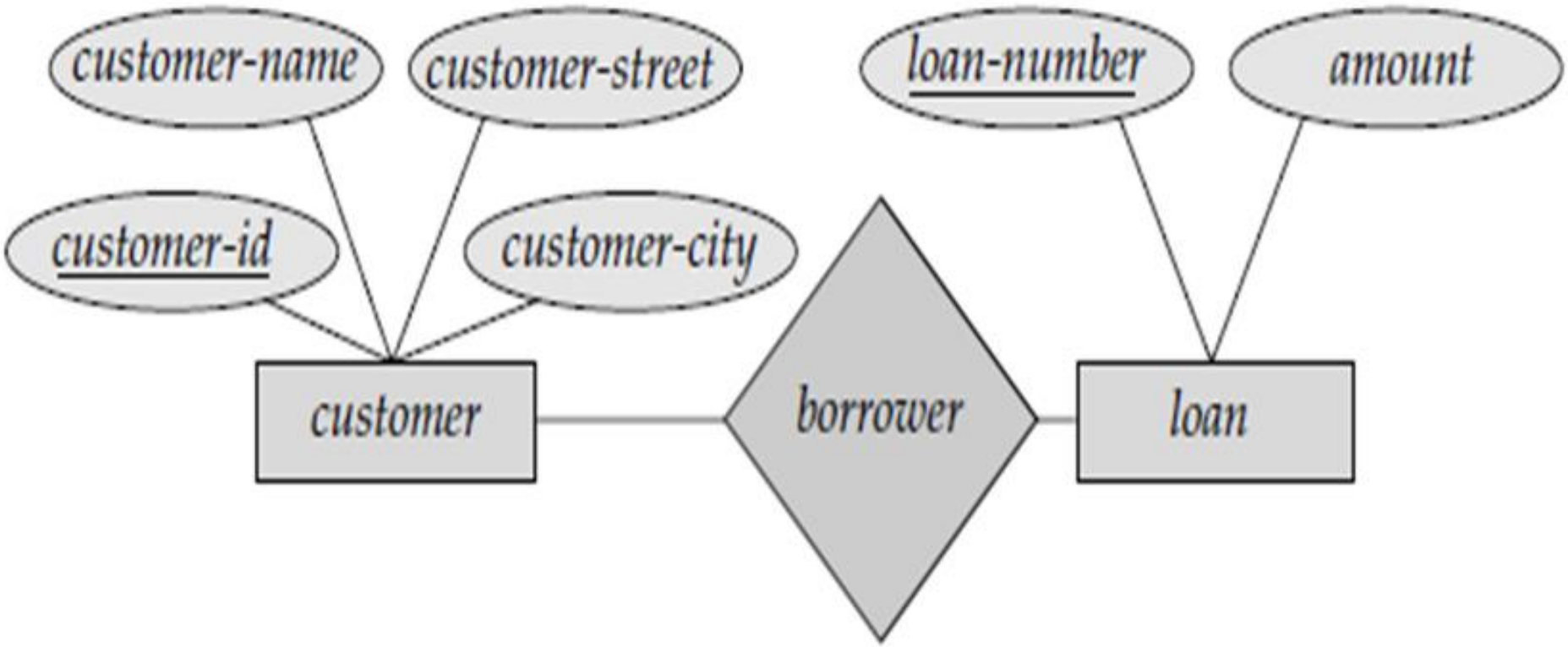
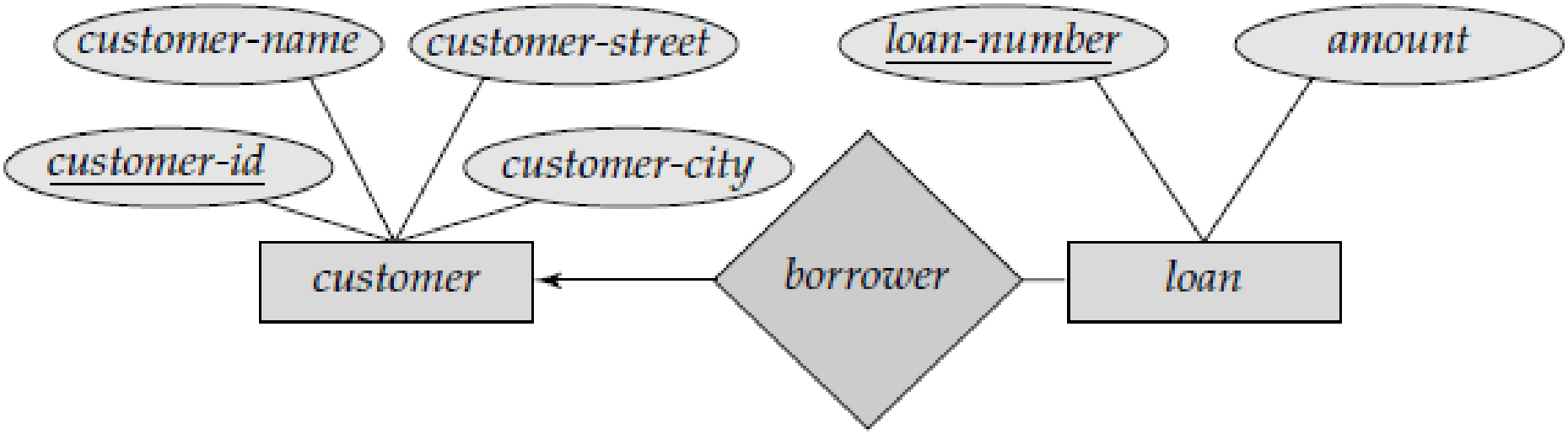
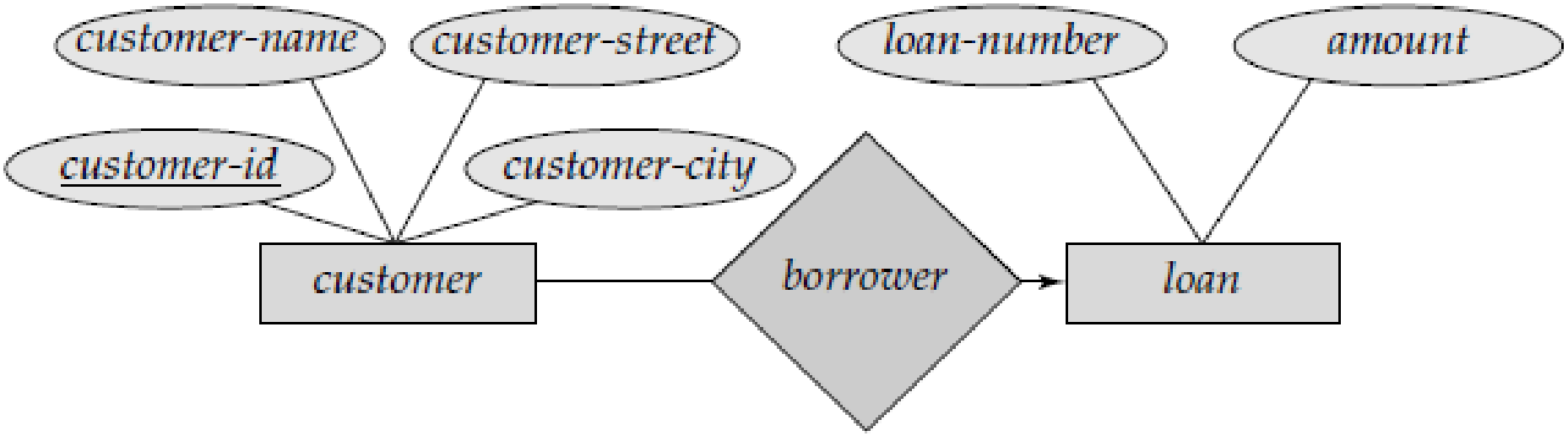


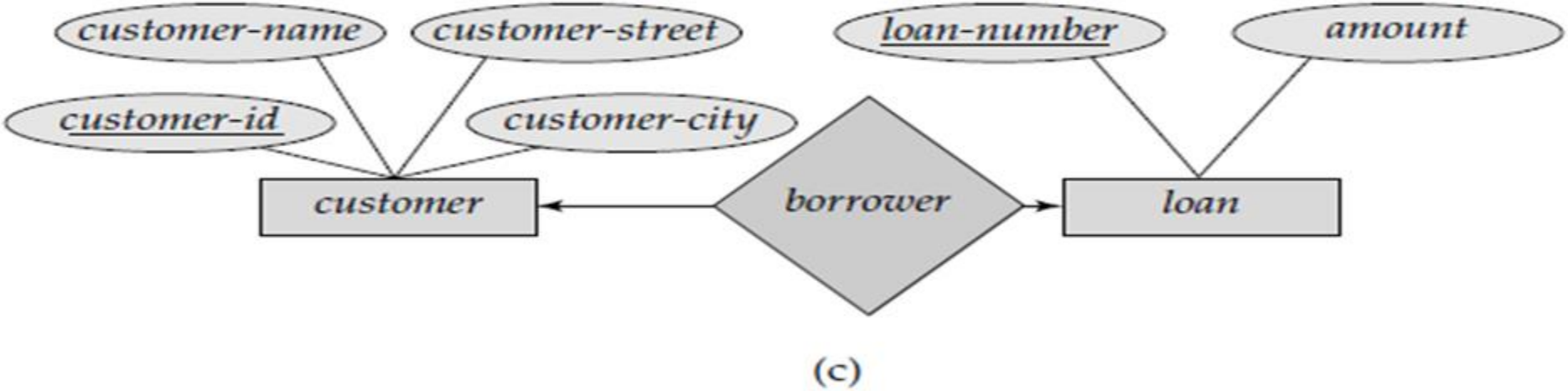
Figure E-R diagram corresponding to customers and loans.


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

(a)


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

(b)



(c)

Figure Relationships. (a) one to many. (b) many to one. (c) one-to-one.

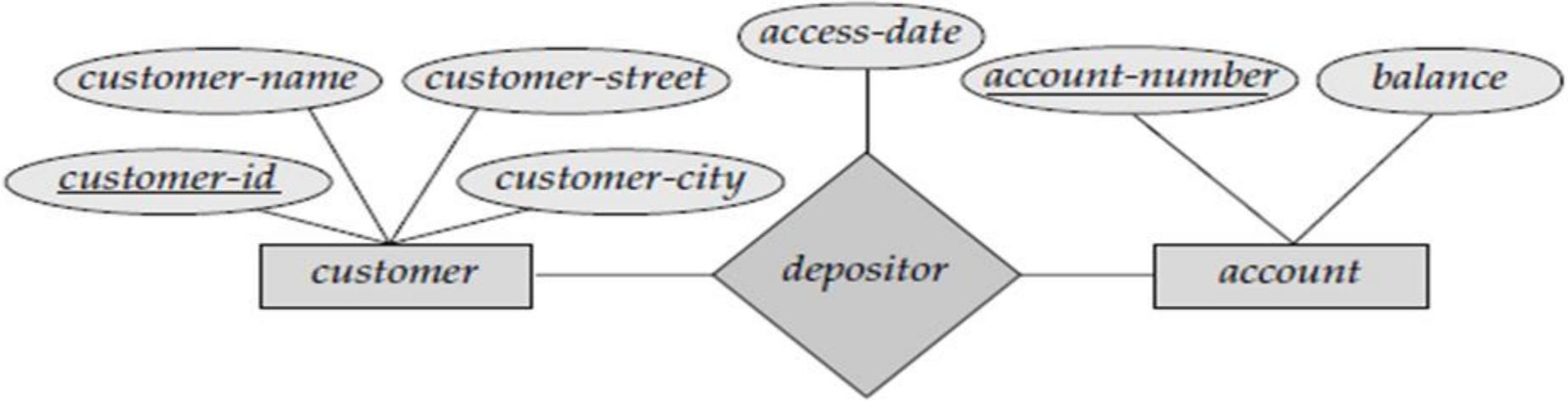


Figure E-R diagram with an attribute attached to a relationship set.

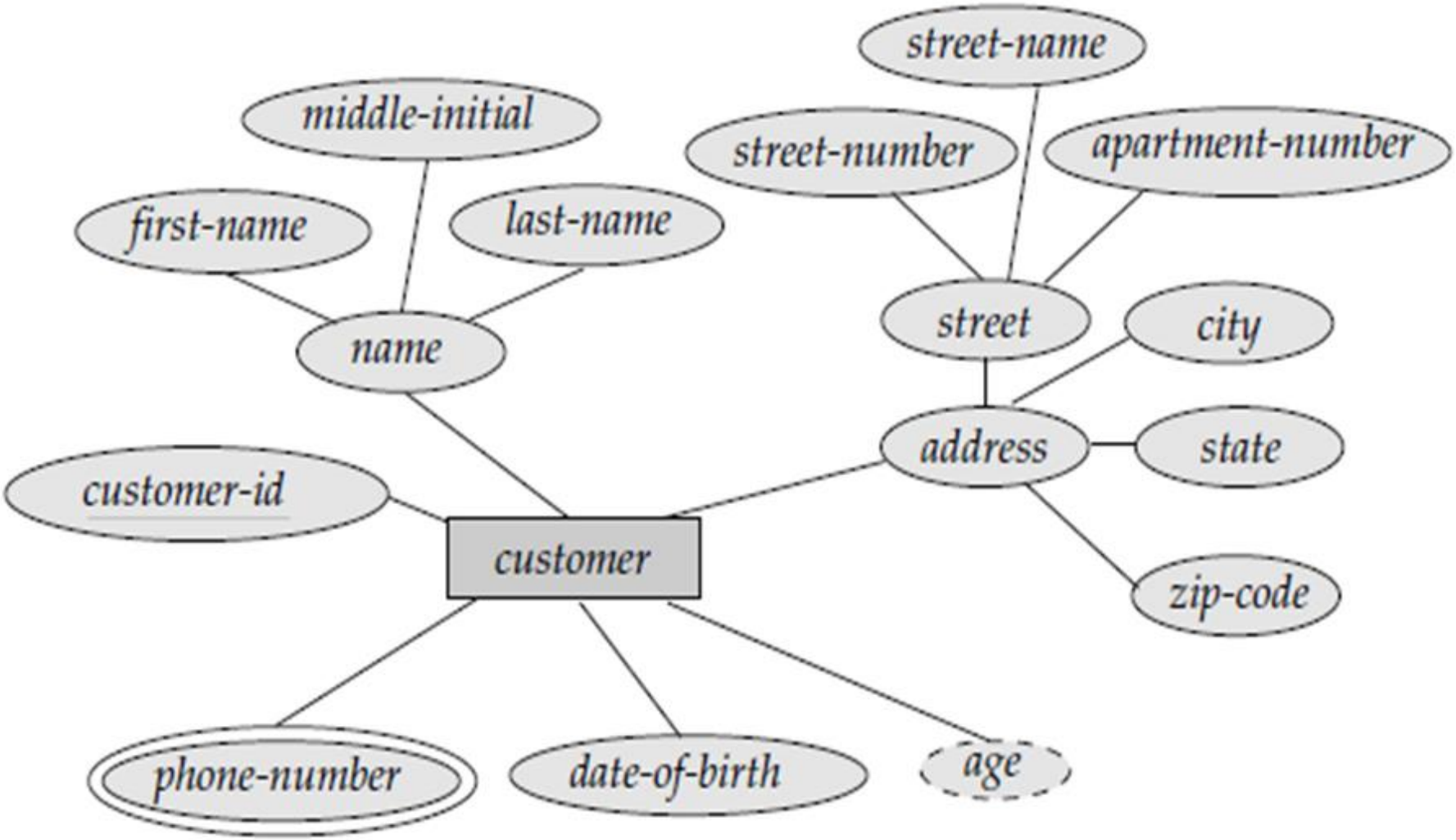


Figure E-R diagram with composite, multivalued, and derived attributes.

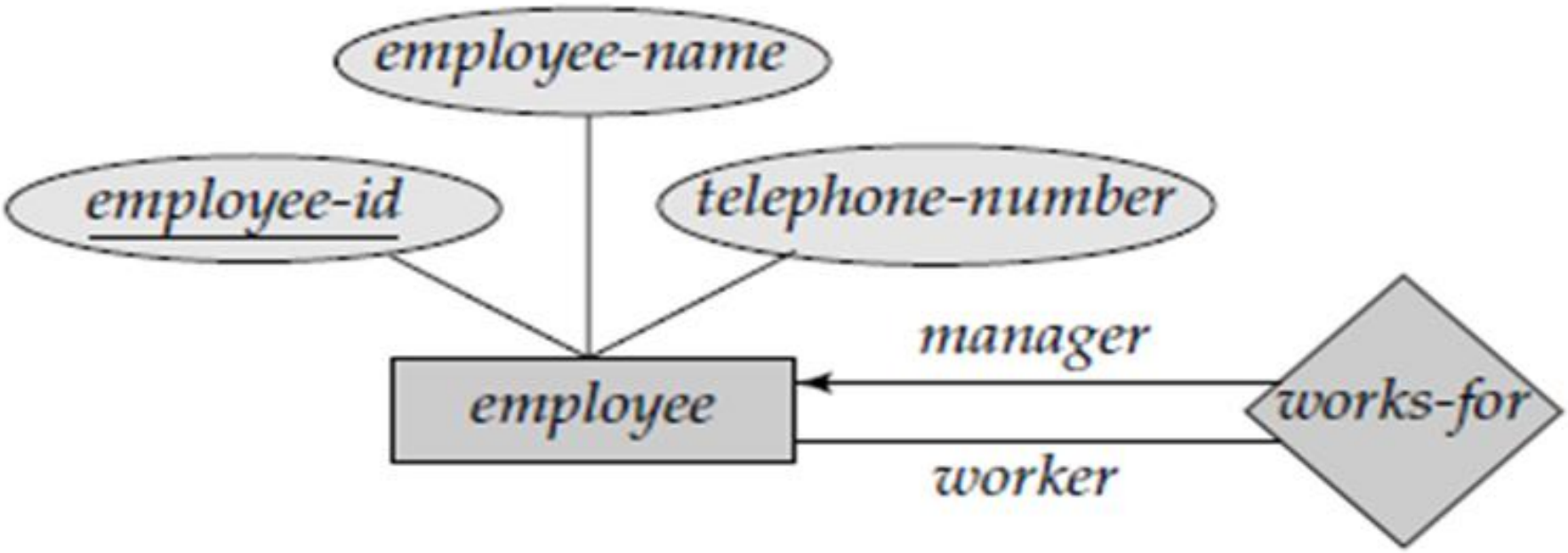


Figure E-R diagram with role indicators.

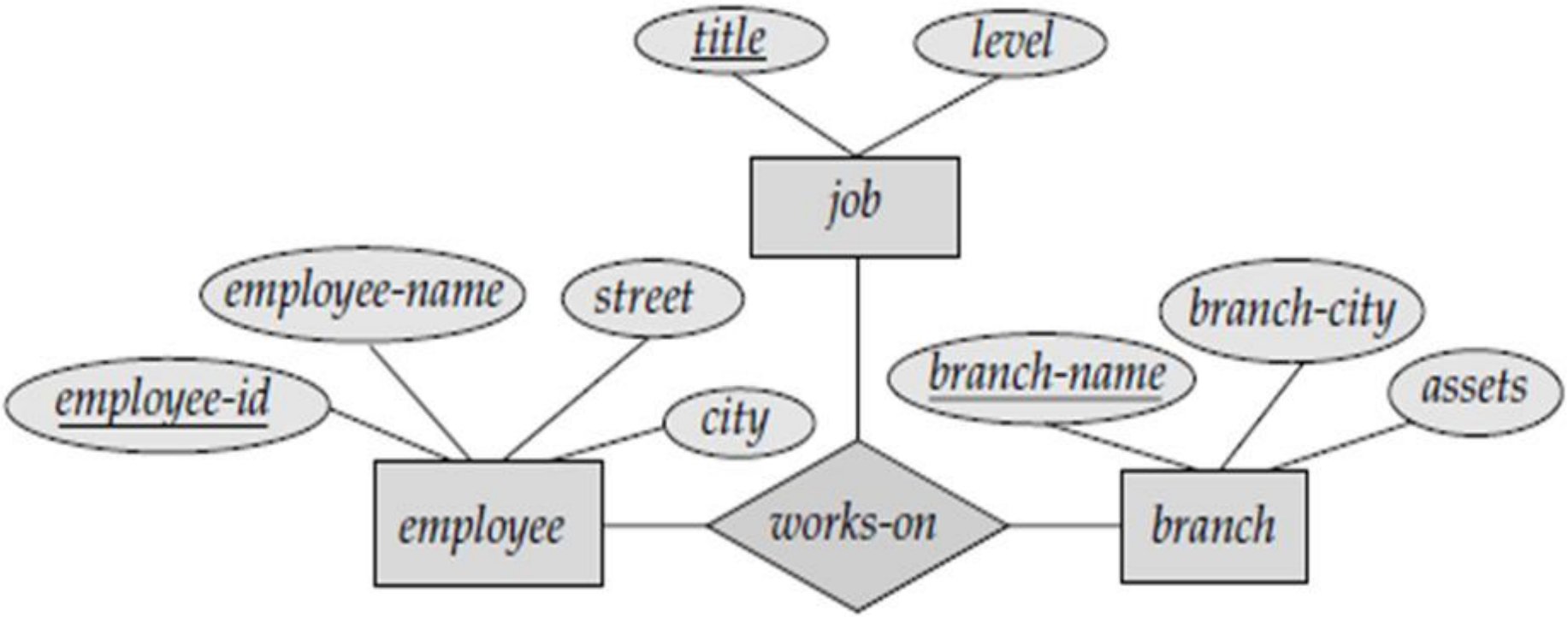


Figure E-R diagram with a ternary relationship.

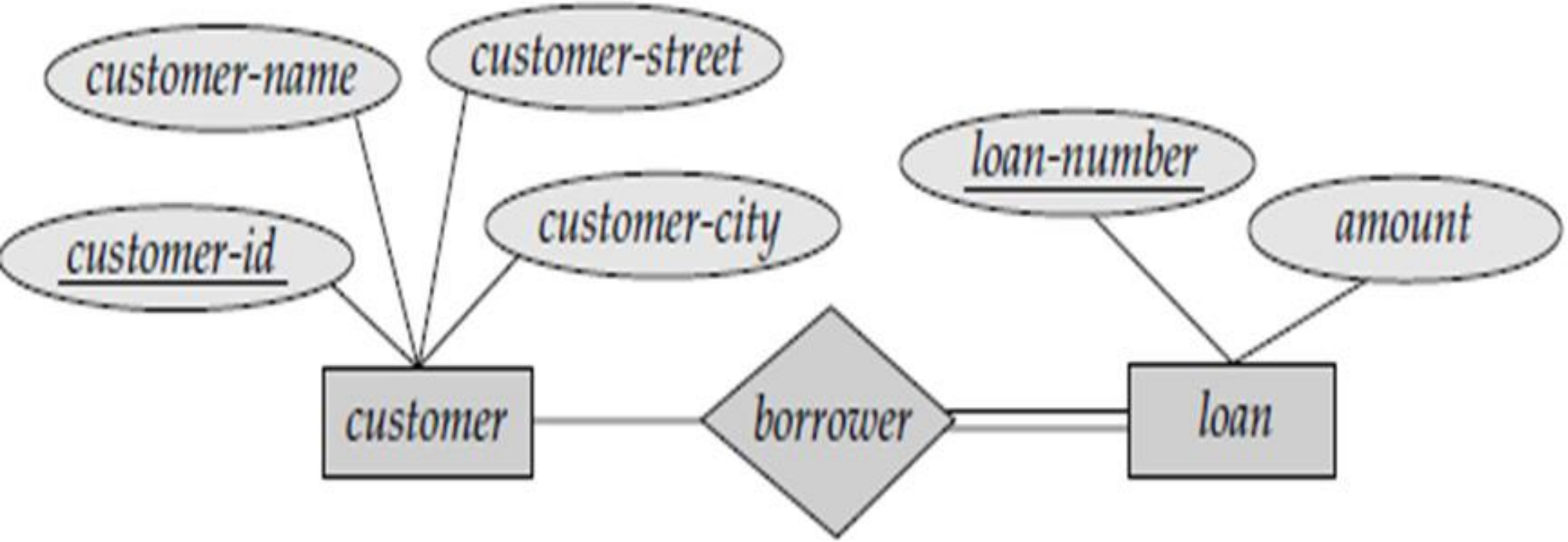


Figure Total participation of an entity set in a relationship set.

Tools To Use To Draw E-R Diagram

- http://dia-installer.de/download/index.html.en
- https://www.diagrameditor.com/

Strong Entity Sets and Weak Entity Sets

- Strong Entity Set
  - An entity set having sufficient attributes to form a primary key.
  - Such an entity set is termed as a Strong Entity Set.
- E.g. Customer Entity Set has attribute customer_id to work as primary key.
- Student Entity Set can have attribute PRN or Class and Roll_No to work as primary key.

- Weak Entity Set
  - An entity set may not have sufficient attributes to form a primary key.
  - Such an entity set is termed as a weak entity set.
- E.g. Cashier in Bank records the details about all the payments only using Payment_Date, Chalan_Number (Payment_Number) and amount.
- It may consist of payment done towards saving account, loan account or current account.
- Consider- Same Payment_Number used to pay to multiple loan accounts.

- No sufficient attribute to form the primary key
- Payment_Number is not sufficient.
- This is Weak Entity Set
- For a weak entity set to be meaningful, it must be associated with another entity set, called the identifying or owner entity set.
- The weak entity set is said to be existence dependent on the identifying entity set.

- The identifying entity set is said to own the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is identifying relationship.
- The identifying relationship is many to one
- from the weak entity set to the identifying entity set, and the participation of the weak entity set in the relationship is total.

- In this example, the identifying entity set for payment is loan/Account
- and a relationship loan-payment that associates payment entities with their corresponding loan entities is the identifying relationship.
- The primary key of a weak entity set is formed by the primary key of the identifying entity set, plus the weak entity set's discriminator.
- In the case of the entity set payment, its primary key is {loan-number, payment-number}

- In E-R diagrams, a doubly outlined box indicate a weak entity set
- a doubly outlined diamond indicates the corresponding identifying relationship.
- In Figure the weak entity set payment depends on the strong entity set loan via the relationship set loan-payment.
- The figure also illustrates the use of double lines to indicate total participation
- The participation of the (weak) entity set payment in the relationship loan-payment is total

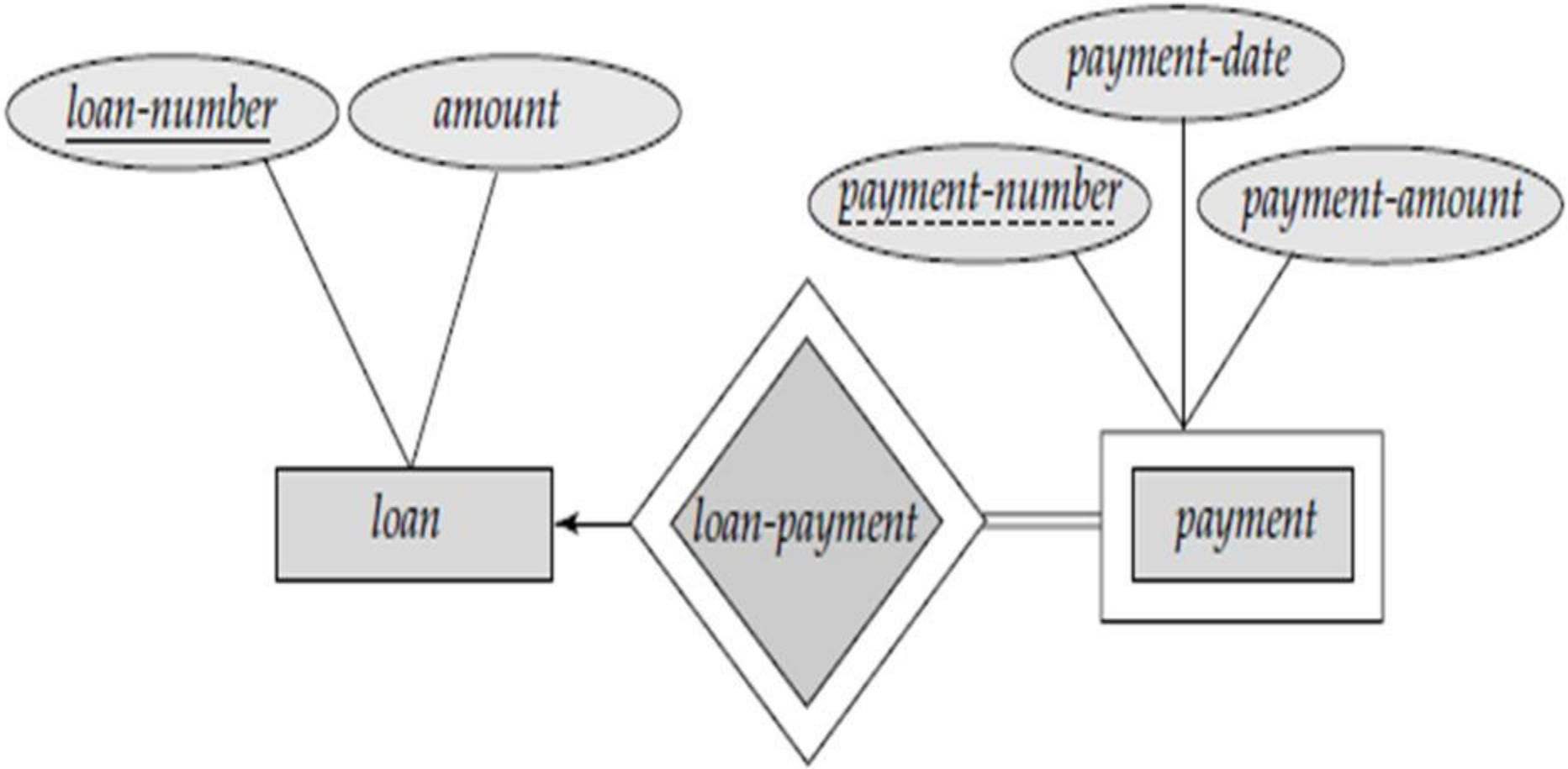


Figure E-R diagram with a weak entity set.

Extended E-R Features

- Specialization
- Generalization
- Attribute Inheritance
- Aggregation

Specialization

- An entity set may include subgroupings of entities that are distinct in some way from other entities in the set.
- For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
- The E-R model provides a means for representing these distinctive entity groupings.

- Consider an entity set person with attributes name, street, and city.
- A person may be further classified as one of the following:
- Customer
- Employee
- Each of these person types is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes.

- For example, Customer entities may be described further by the attribute customer-id
- whereas employee entities may be described further by the attributes employee-id and salary.
- The process of designating subgroupings within an entity set is called specialization.
- The specialization of person allows us to distinguish among persons according to whether they are employee or customer.

- Another Example
- The bank wishes to divide accounts into two categories, Current account and Savings account.

- In E-R diagram, specialization is depicted by a triangle component labelled ISA.
- The label ISA stands for "is a"
- For example, a customer "is a" person.
- The ISA relationship may also be referred to as a Superclass-Subclass Relationship.
- Higher- and lower-level entity sets are depicted as regular entity sets - rectangles containing the name of the entity set.

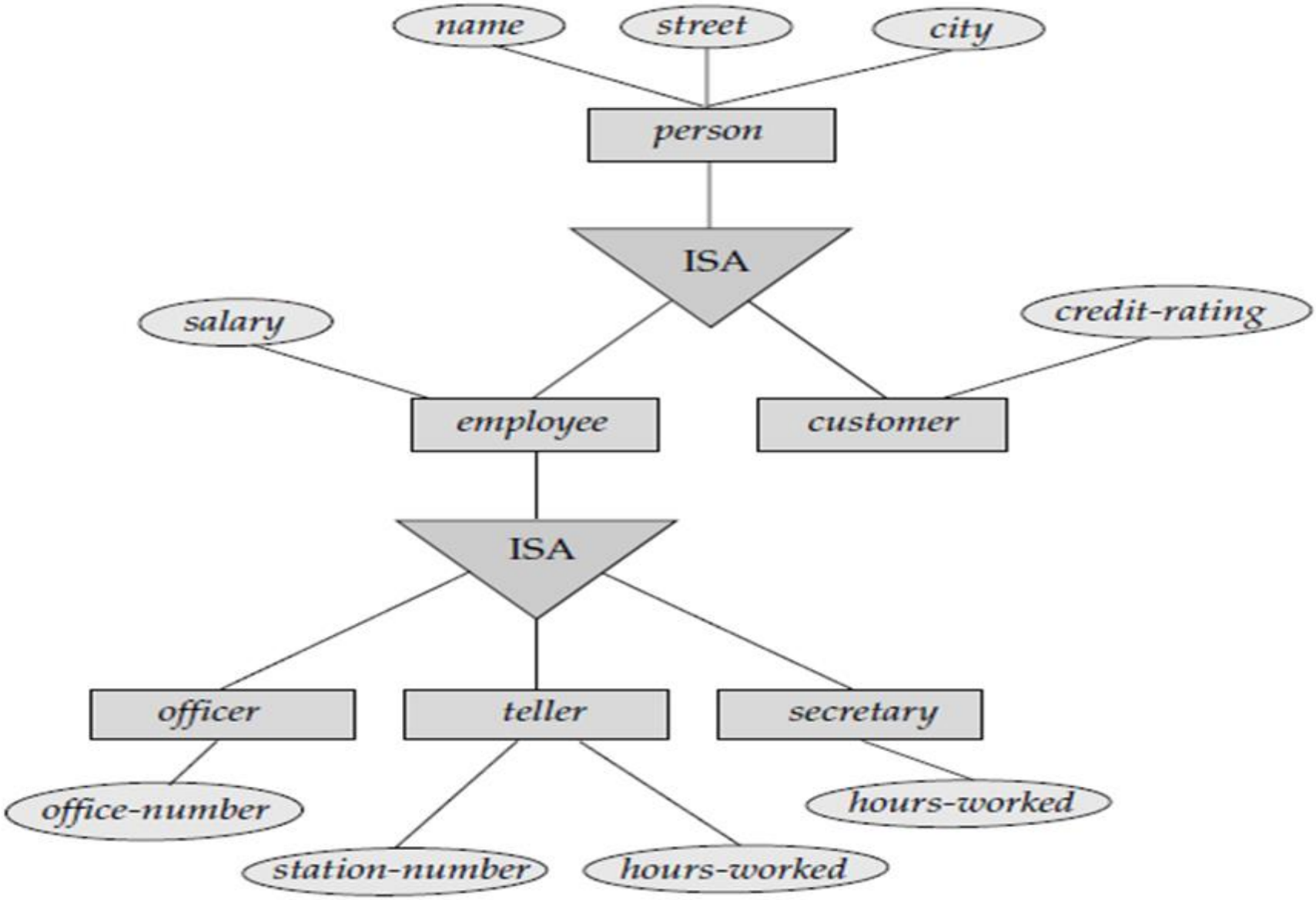


Figure Specialization and generalization.

Generalization

- The refinement from an initial entity set into successive levels of entity subgroupings represents a top-down design process in which distinctions are made explicit.
- It is called Specialization.
- The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.
- It is called Generalization.

- Customer entity set with the attributes name, street, city, and customer-id
- Employee entity set with the attributes name, street, city, employee-id, and salary
- There are similarities between the Customer entity set and the Employee entity set
- They have several attributes in common.
- This commonality can be expressed by Generalization

- Generalization is containment relationship that exists between a higher-level entity set and one or more lower-level entity sets.
- Person is the higher-level entity set and Customer and Employee are lower-level entity sets.
- Higher- and lower-level entity sets may be also designated by the terms Superclass and Subclass, respectively.
- The person entity set is the superclass of the customer and employee subclasses.

- Generalization is a simple inversion of specialization.
- In E-R Diagram both can be represented in combination.

- Differences in the two approaches may be characterized by their starting point and overall goal.

- Specialization stems from a single entity set
- It emphasizes differences among entities within the set by creating distinct lower-level entity sets.
- These lower-level entity sets may have attributes or may participate in relationships, that do not apply to all the entities in the higher-level entity set.

- Generalization proceeds from the recognition that a number of entity sets share some common features

- They are described by the same attributes and participate in the same relationship sets.
- On the basis of their commonalities, generalization synthesizes these entity sets into a single, higher-level entity set.
- Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences
- it also permits an economy of representation in that shared attributes are not repeated.

Attribute Inheritance

- A crucial property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance.
- The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.
- For example, customer and employee inherit the attributes of person.

- Customer is described by its name, street, and city attributes, and additionally a customer-id attribute

- Employee is described by its name, street, and city attributes, and additionally employee-id and salary

- A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.
- The officer, teller, and secretary entity sets can participate in the works-for relationship set
- since the superclass employee participates in the works-for relationship.
- Attribute inheritance applies through all tiers of lower-level entity sets.

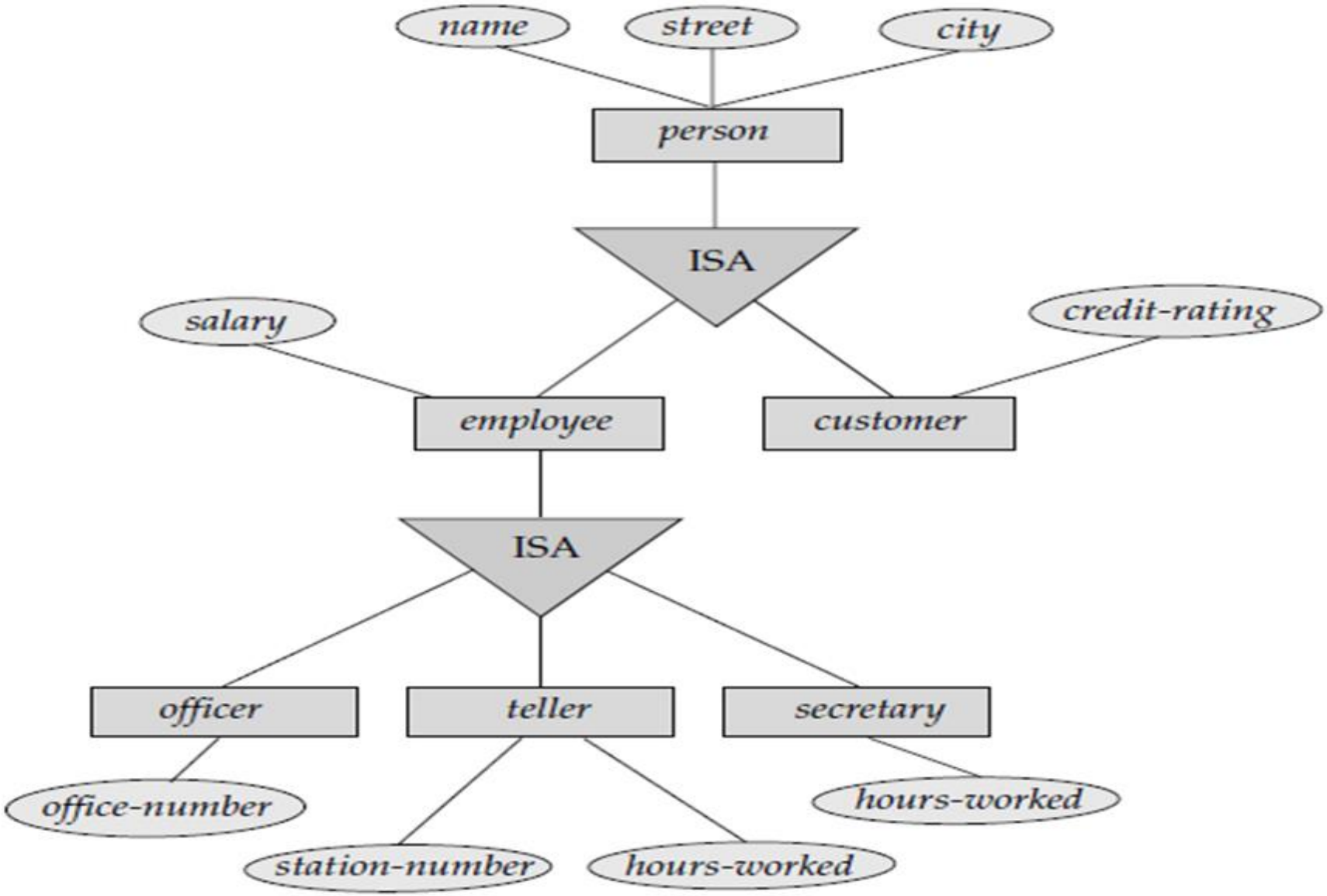


Figure Specialization and generalization.

- Single Inheritance
- Inherited from single superclass

- Multiple Inheritance
- Inherited from multiple superclasses

Constraints on Generalizations

- One type of constraint involves determining which entities can be members of a given lower-level entity set.
- Such membership may be one of the following:
- Condition-defined
- User-defined

- Condition-defined
- In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate.
- E.g. Account – account type is saving or current

- User-defined
- User-defined lower-level entity sets are not constrained by a membership condition
- rather, the database user assigns entities to a given entity set.
- For instance, let us assume that, after 3 months of employment, bank employees are assigned to one of the work teams.

- A second type of constraint relates to whether or not entities may belong to more than one lower-level entity set within a single generalization.
- The lower-level entity sets may be one of the following:
- Disjoint
- Overlapping

- Disjoint
- A disjoint constraint requires that an entity belong to no more than one lower-level entity set.
- E.g. Account can be either saving or current.
- Same account number can’t be in both the low-level entity set

- Overlapping
- In overlapping generalizations, the same entity may belong to more than one lower-level entity set within a single generalization.
- E.g. One of the Faculty is pursuing Ph.D. in same college.
- Same record exist in both the low-level entity sets Student and Faculty

- A final constraint, the completeness constraint on a generalization or specialization,
- It specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/ specialization
- This constraint may be one of the following:
- Total Generalization or Total Specialization
- Partial Generalization or Partial Specialization

- Total Generalization or Total Specialization
- Each higher-level entity must belong to a lower-level entity set
- E.g. The account generalization is total
- All account entities must be either a savings account or a current account

- We can specify total generalization in an E-R diagram by using a double line to connect the box representing the higher-level entity set to the triangle symbol.
- This notation is similar to the notation for total participation in a relationship.

- Partial Generalization or Partial Specialization
- Some higher-level entities may not belong to any lower-level entity set
- Partial generalization is the default.
- The work team entity sets illustrate a partial specialization.
- Since employees are assigned to a team only after 3 months on the job,
- some employee entities may not be members of any of the lower-level team entity sets.

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- Consider the ternary relationship works-on, between a employee, branch, and job
- Suppose we want to record managers for tasks performed by an employee at a branch
- We want to record managers for (Employee, Branch, Job) combinations.

- One alternative for representing this relationship is to create a quaternary relationship manages between Employee, Branch, Job, and Manager.
- But it is difficult to mange.
- The best way to model a situation is to use Aggregation.
- Aggregation is an abstraction through which relationships are treated as higher-level entities.

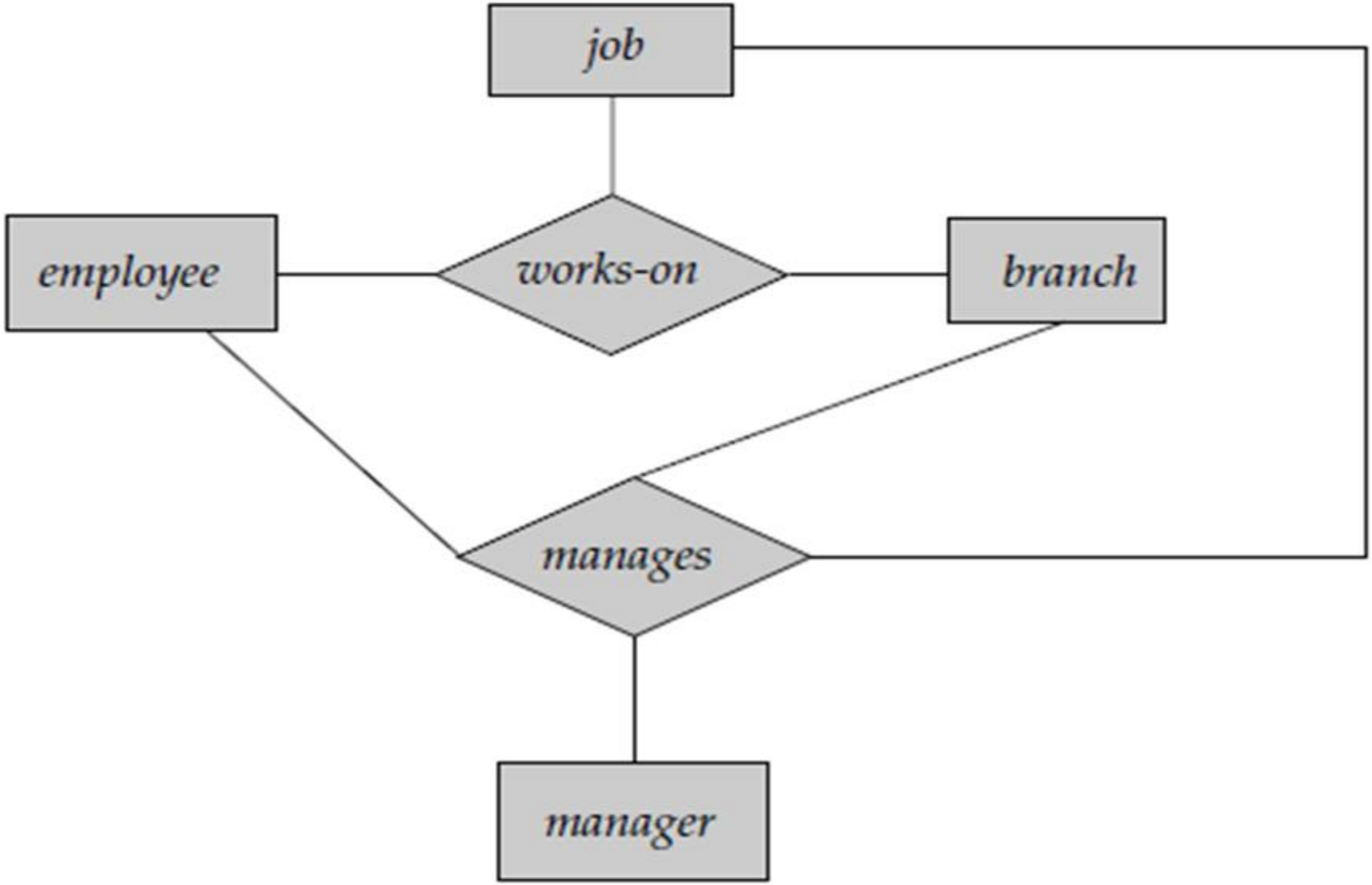


Figure E-R diagram with redundant relationships.

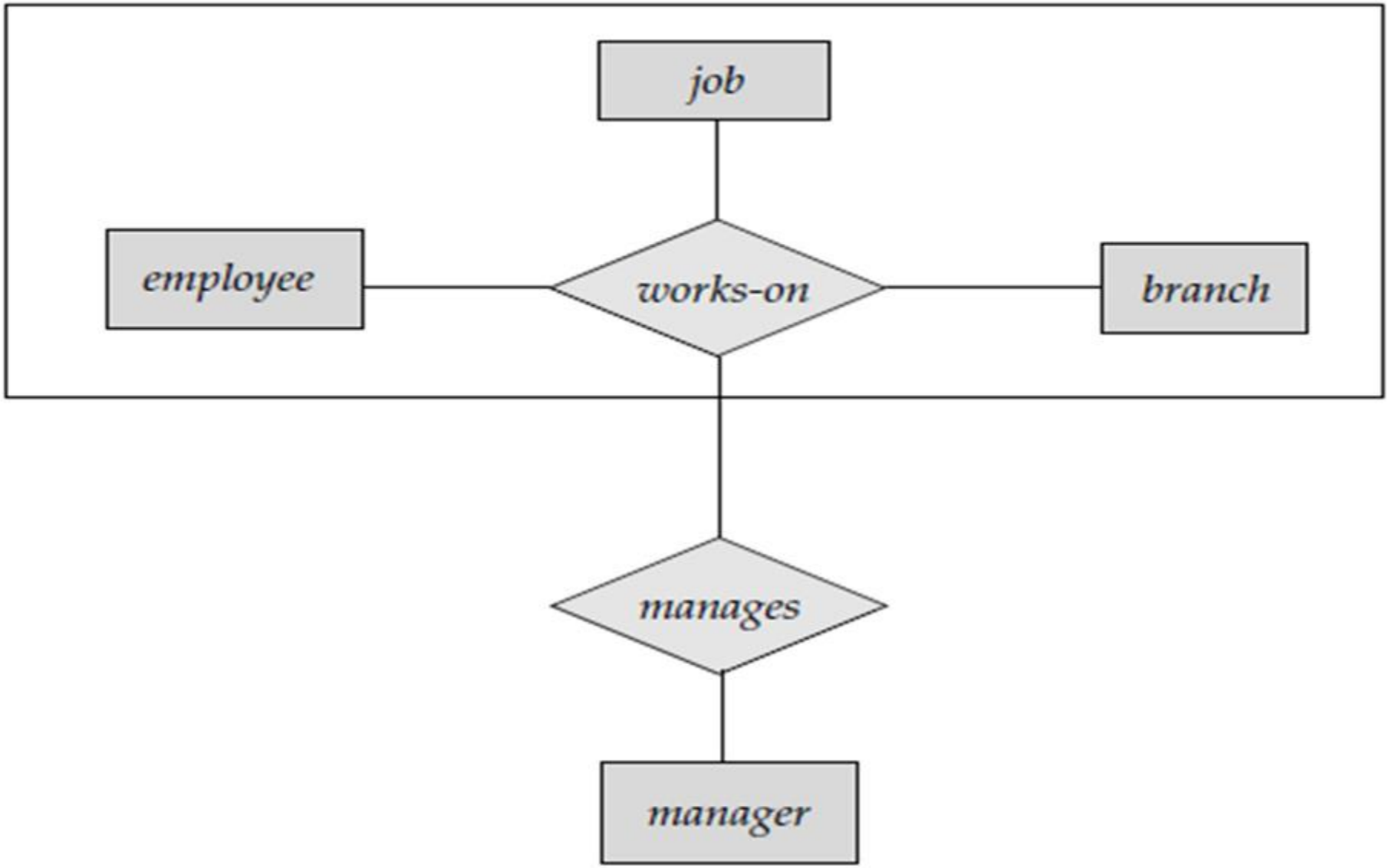


Figure E-R diagram with aggregation.

Design of an E-R Database Schema

- The E-R data model gives us much flexibility in designing a database schema to model a given enterprise.
- How a database designer may select from the wide range of alternatives?
- Among the designer's decisions are:

1] Whether to use an attribute or an entity set to represent an object/thing

2] Whether a real-world concept is expressed more accurately by an entity set or by a relationship set

3] Whether to use a ternary relationship or a pair of binary relationship

4] Whether to use a weak entity set along with owner entity set or convert into strong entity set

5] Whether using generalization is appropriate?

6] Whether using aggregation is appropriate?

Database Design Phases

1] A] Requirement Collection

B] Requirement Analysis

2] A] Logical Design

B] Physical Design

3] Implementation / Coding / Table Creation

Reduction of Schema into Tables

4] Testing – by writing queries

1] A] Requirement Collection

- Collect the requirements from end user.
- Data needs of the prospective database users
- Users describe the kinds of operations (or transactions) that will be performed on the data.

- Bank has 20 branches.
- Bank provide facility for saving account and loan account
- Customer can have both the accounts.
- Customer can have multiple accounts at different branches.

1] B] Requirement Analysis

- Analyse the collected requirements to make it easier for design and implementation.
- Entity, Relationship, Attribute
- In a specification of functional requirements describe the kinds of operations (or transactions) that will be performed on the data

- Two types of account - Saving account and loan account
- Describe the attributes of saving and loan account
- Customer is associated with both the accounts, so store it only at one place.
- Branch is also associated with both the account.

2] A] Logical Design

- Database designer chooses a data model
- Apply the concepts of the chosen data model
- Translates the requirements into a conceptual schema of the database.
- The designer reviews the schema to confirm that all data requirements are indeed satisfied and are not in conflict with one another.
- E.g. E-R Diagram

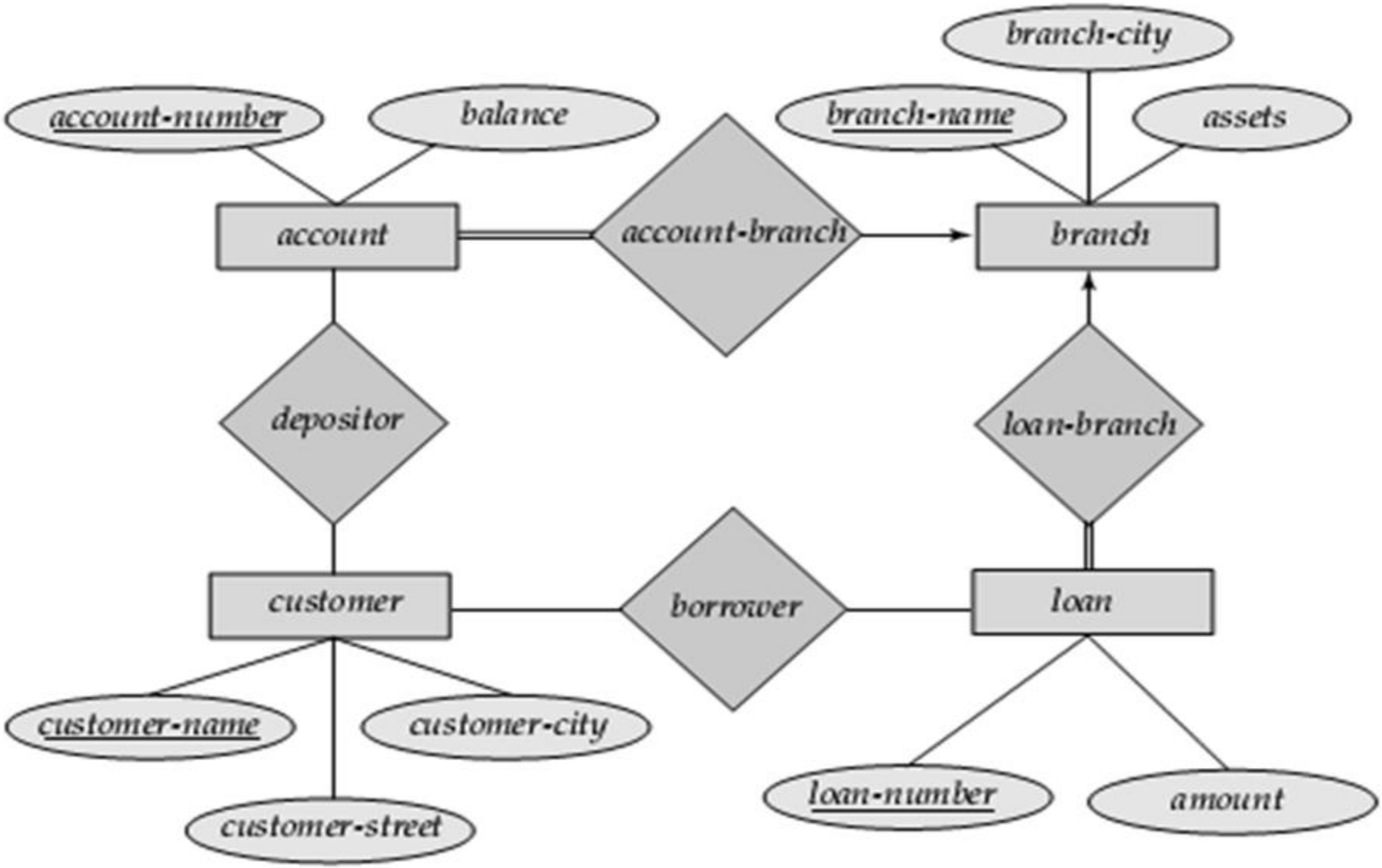


Figure E-R diagram for the banking enterprise.

2] B] Physical Design

- Logical Schema designed in step 2] A] get converted into Physical Schema
- The designer uses the resulting system specific database schema in the subsequent physical-design phase, in which the physical features of the database are specified.
- E.g. Schema Diagram

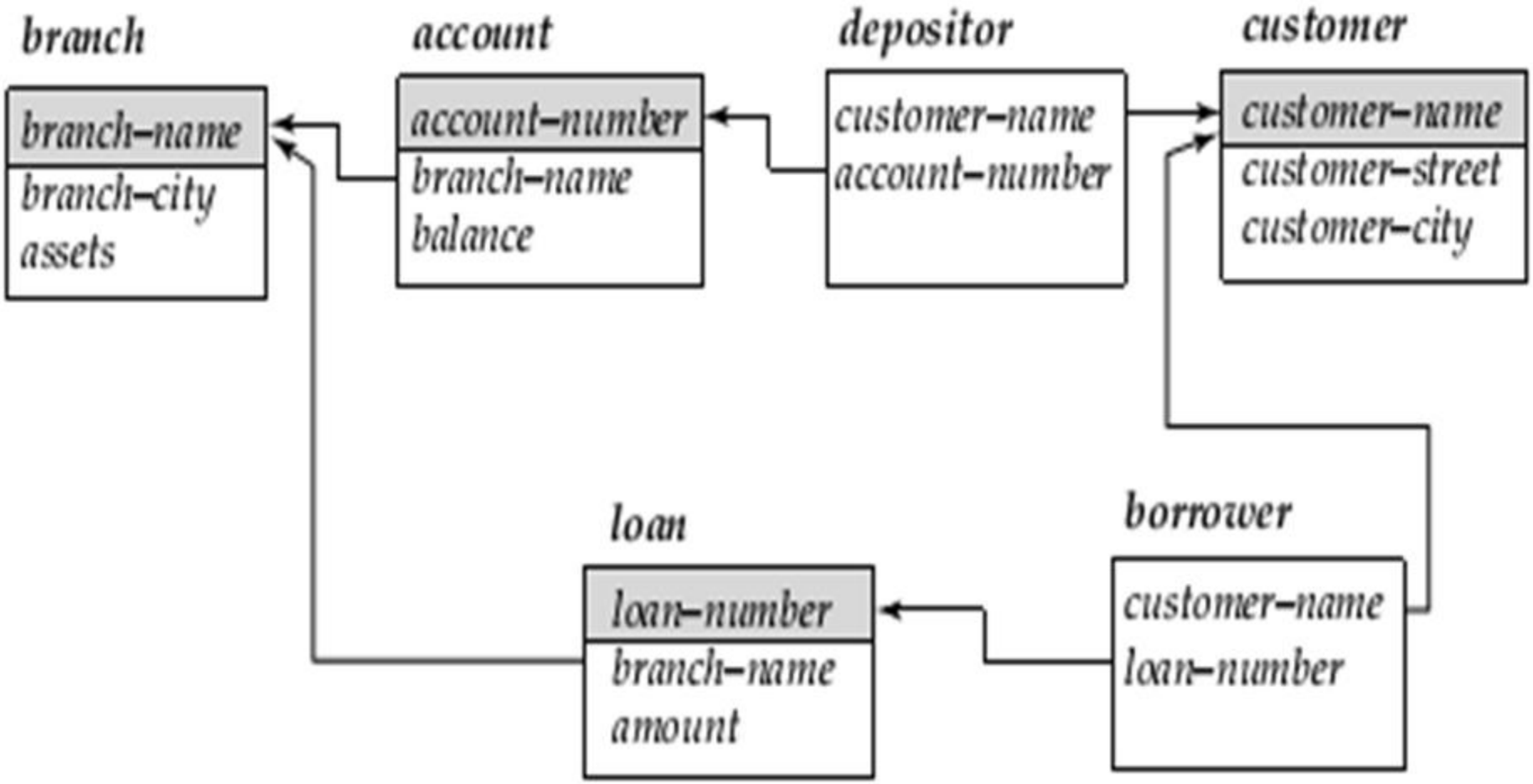


Figure Schema diagram for the banking enterprise.

3] Reduction of E-R Schema into Tables

- For each entity set and for each relationship set, a table is required to be created.
- Assign the name of the corresponding entity set or relationship set to that table.
- Each table has multiple columns, each of which has unique name.

- Tabular Representation of Strong Entity Sets
- Tabular Representation of Weak Entity Sets
- Tabular Representation of Relationship Sets
- Redundancy of Tables
- Combination of Tables
- Composite Attributes
- Multivalued Attributes
- Tabular Representation of Generalization
- Tabular Representation of Aggregation

Tabular Representation of Strong Entity Sets

- Let E be a strong entity set with descriptive attributes $a_1, a_2, \ldots, a_n$.
- We represent this entity by a table called E with n distinct columns,
- each of which corresponds to one of the attributes of E.
- Each row in this table corresponds to one entity of the entity set E.

- E.g. loan with attributes loan number and amount

loan-number	*amount*
L-11	900
L-14	1500
L-15	1500
L-16	1300
L-17	1000
L-23	2000
L-93	500

Figure The *loan* table.

Tabular Representation of Weak Entity Sets

- Let A be a weak entity set with attributes $a_1, a_2,...,a_m$
- Let B be the strong entity set on which A depends.
- Let the primary key of B consist of attributes $b_1, b_2, \ldots, b_n$.
- We represent the entity set A by a table called A with one column for each attribute of the set:
- $\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$

- Loan
- Payment

loan-number	*payment-number*	*payment-date*	*payment-amount*
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

Figure The *payment* table.

Tabular Representation of Relationship Sets

- Let R be a relationship set, let $a_1, a_2, \ldots, a_m$ be the set of attributes formed by the union of the primary keys of each of the entity sets participating in R
- and let the descriptive attributes (if any) of R be $b_1, b_2, \ldots, b_n$.
- We represent this relationship set by a table called R with one column for each attribute of the set:
- $\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$

- E.g. relationship - borrower
- Customer – primary key customer id
- Loan – primary key loan number

customer-id	*loan-number*
019-28-3746	L-11
019-28-3746	L-23
244-66-8800	L-93
321-12-3123	L-17
335-57-7991	L-16
555-55-5555	L-14
677-89-9011	L-15
963-96-3963	L-17

Figure The *borrower* table.

Redundancy of Tables

- Redundancy must be avoided in multiple tables.
- In good database design, only key attributes are stored redundantly in multiple tables.
- Non-key attributes should be stored only at one place.

- Redundancy occurs mostly in case of weak entity set.
- Table used for Identifying relationship is redundant
- E.g. Every(loan-number, payment-number) combination in loan-payment would also be present in the payment table.
- Thus, the loan-payment table is redundant.

Combination of Tables

- Combine table for account-branch with the table for account and require only the following two tables:
- account, with attributes account-number, balance, and branch-name
- branch, with attributes branch-name, branch-city, and assets
- No need of table account-branch representing relationship

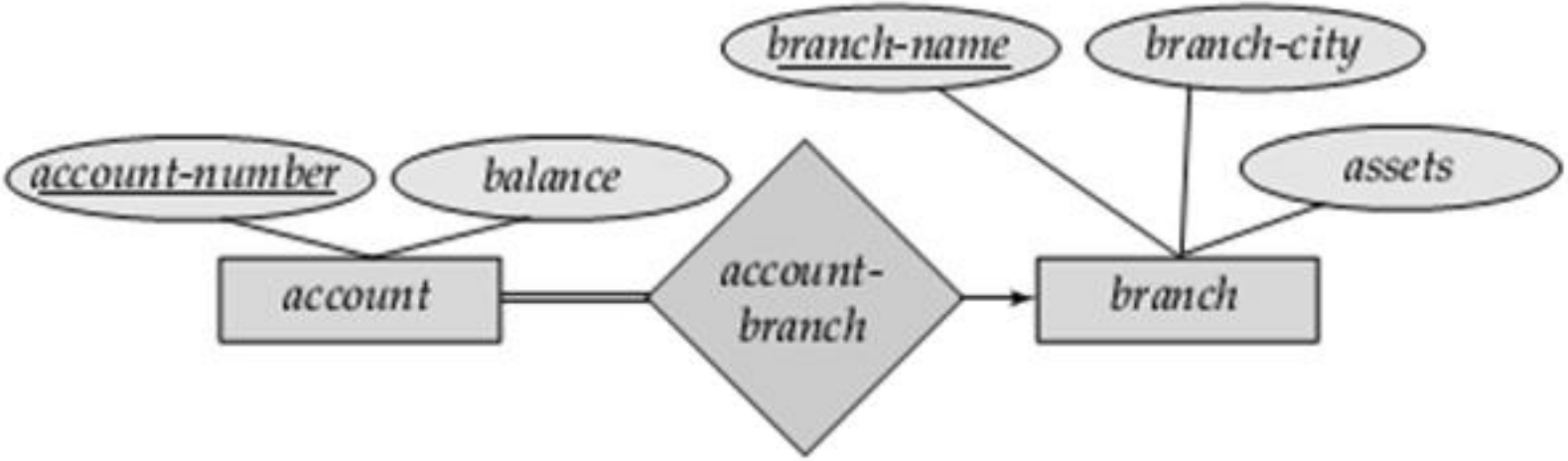


Figure E-R diagram.

Composite Attributes

- Its better to split the composite attribute into simple attributes and represent as a column of table.
- Representing composite attribute as it is may create data retrieval related issue.
- E.g. Name composed of First Name, Middle Name and Last Name
- If Name is stored as a composite attribute, it is difficult to retrieve First Name only

Multivalued Attributes

- Different ways to represent multivalued attributes.
- E.g. multiple phone numbers of person
- Single column storing multiple phone numbers separated by comma – Retrieval is Difficult
- Multiple columns, one for each phone number – may result into insufficient columns or wastage of columns.
- Better to apply Normalization and create one more table.

Tabular Representation of Generalization
Tabular Representation of Specialization

- Create a table for the higher-level entity set.
- For each lower-level entity set, create a table that includes a column for each of the attributes of that entity set.
- plus a column for each attribute of the primary key of the higher-level entity set.
- E.g. Person - Faculty, Student

Tabular Representation of Aggregation

- Separate table required to be created for each of the entity set taking part in aggregation.
- The table for the relationship set manages between the aggregation of works-on and the entity set manager includes
- a column for each attribute in the primary keys of the entity set manager and the relationship set works-on.

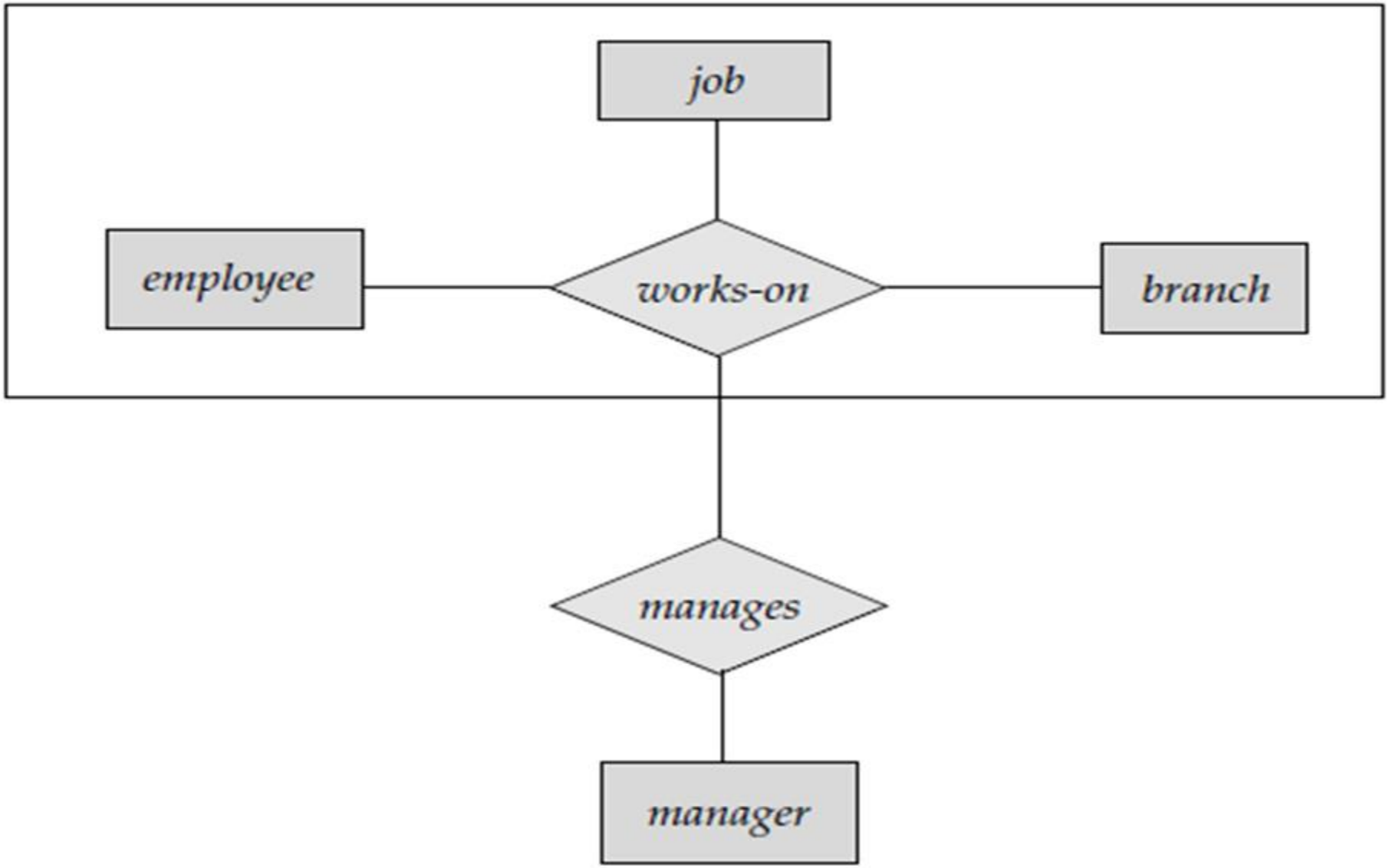
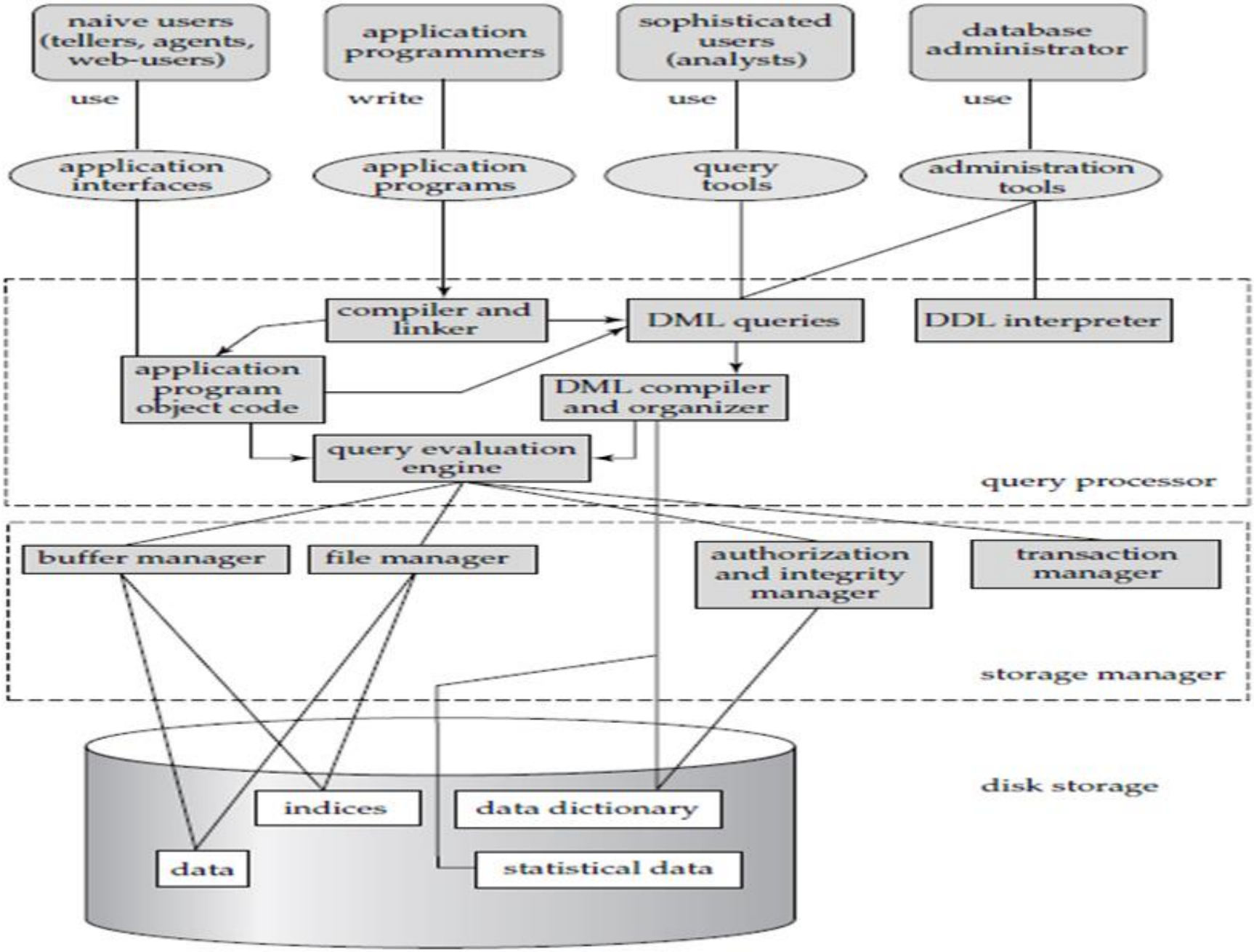


Figure E-R diagram with aggregation.

Database System Structure

- A database system is partitioned into modules
- that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into
  - The Storage Manager
  - The Query Processor


naive users (tellers, agents, web-users)
application programmers
sophisticated users (analysts)
database administrator
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
indices
data dictionary
data
statistical data
disk storage

Storage Manager

- It is a program module 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, which is usually provided by a conventional operating system.

- The storage manager translates the various DML statements into low-level file-system commands.
- The storage manager is responsible for storing, retrieving, and updating data in the database.
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager

- Authorization and Integrity Manager
- Tests for the satisfaction of integrity constraints
- Helps to maintain the integrity of DBMS
- Checks the authority of users to access data.
- Helps to prevent unauthorized data access.

- Transaction Manager
- Ensures that the database always remains in a consistent (correct) state despite system failures
- Ensures concurrent transaction executions proceed without conflicting.
- Helps to maintain ACID properties of Transaction
- A – Atomicity
- C – Consistency
- I – Isolation
- D – Durability

- File Manager
- Data logically represented in form of table get converted into records.
- Records are stored in form of files.
- Blocks of information in files stored in sectors and tracks of disk storage
- 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
- Decide 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 store metadata about the structure of the database, in particular the schema of the database.
- Indices - which provide fast access to data items that hold particular values.
- Statistical Data – Statistical observations related to size of database, number of users, peak time etc.

The Query Processor

- The query processor components include
- DDL Interpreter
- DML Compiler and Organizer
- Query Evaluation Engine
- Compiler and Linker

- DDL Interpreter
- Receive the Commands in form of DDL statements like Create, Alter, Rename, Drop, Truncate
- Interprets DDL statements and records the definitions in the data dictionary.

- DML Compiler and Organizer
- It receives command in form of DML Statements.
- It translates the DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

- Query Evaluation Engine
- It executes low-level instructions generated by the DML Compiler and received from Application Program Object Code

- Compiler and Linker
- It is used to separate application program part and DML statements from application program received as input.

Database Application Architectures

- Database applications are usually partitioned into two or three parts
  - Two-tier architectures
  - Three-tier architectures

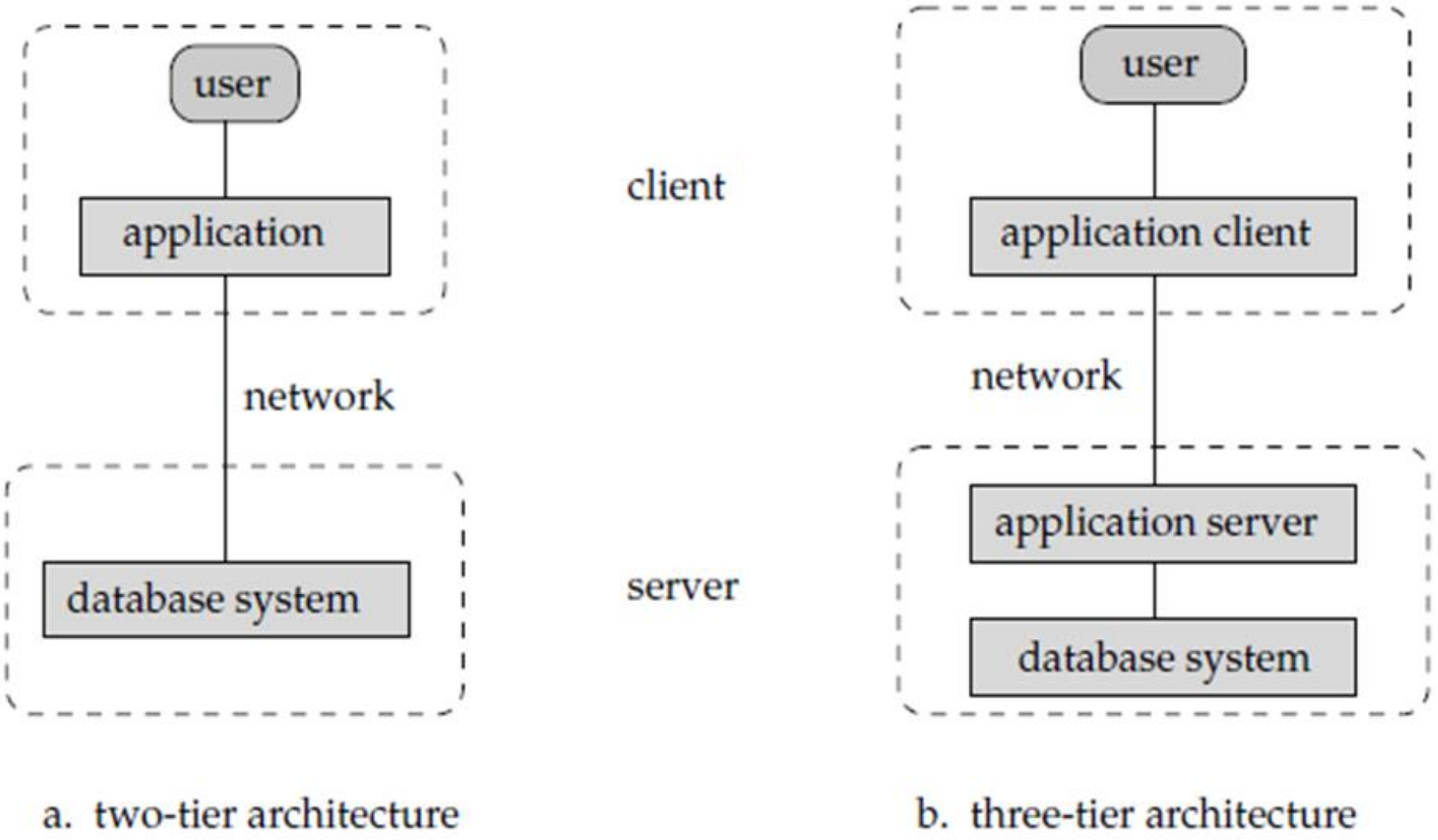


Figure Two-tier and three-tier architectures.

- Two-tier architectures
- The application is partitioned into a component that resides at the client machine
- It invokes database system functionality at the server machine through query language statements.
- Application program interface standards like JDBC are used for interaction between the client and the server.

- Three-tier architectures
- The client machine acts as merely a front end and does not contain any direct database calls.
- Instead, the client end communicates with an application server, usually through a forms interface.
- The application server in turn communicates with a database system to access data.

Roles in the Database Environment

- A primary goal of a database system is to retrieve information from and store new information in the database.
- People who work with a database can be categorized as
  - Database Users
  - Database Administrators

Database Users and User Interfaces

- There are four different types of database-system users
- differentiated by the way they expect to interact with the system.
- Different types of user interfaces have been designed for the different types of users.

- Naive Users
- Unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer.

- Application Programmers
- They are computer professionals who write application programs.
- Application programmers can choose from many tools to develop user interfaces.
- They can write application program using any application development tool.

- Sophisticated Users
- Interact with the system without writing programs.
- Instead, they form their requests in a database query language.
- They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.
- Analysts who submit queries to explore data in the database fall in this category.

- Specialized Users
- They are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.
- Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

Database Administrator

- One of the main reasons for using DBMS is to have central control of both the data and the programs that access those data.
- A person who has such central control over the system is called a database administrator (DBA).
- The functions of a DBA include:

- Schema definition
  - The DBA creates the original database schema by executing a set of DDL Statements
- Storage structure and access-method definition
  - The DBA define the storage mechanisms and access control mechanisms
- Schema and physical-organization modification
  - The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

- Granting of authorization for data access
  - By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
- Routine maintenance
  - Examples of the database administrator's routine maintenance activities are:
  - Periodically backing up the database,
  - Ensuring that enough free disk space is available
  - Monitoring jobs running on the database and ensuring that performance is not degraded

Thank You