Department Of Computer Science
B.Sc-I
Semester-II
Subject-Relational DataBase Management System
Unit-I
DataBase Design Entity Relationship(ER)

Basic Structure of Entity Relationship

Entity Relationship diagram is used to represent data object.

It is developed by Charles Bachman.

ER model first introduce 1976.

It is a graphical representation of entity and their relationship in a database structure.

ER model normally represented in an Entity Relationship Diagram(ERd) with user graphical representation on to a model database component.

ER Diagram diagrammatic representation of entity, their attributes and relationship between the entities.

Notation Used For ER Diagram for below fig.

Department Of Computer Science
B.Sc-I
Semester-II
Subject-Relational DataBase Management System
Unit-I
DataBase Design Entity Relationship(ER)

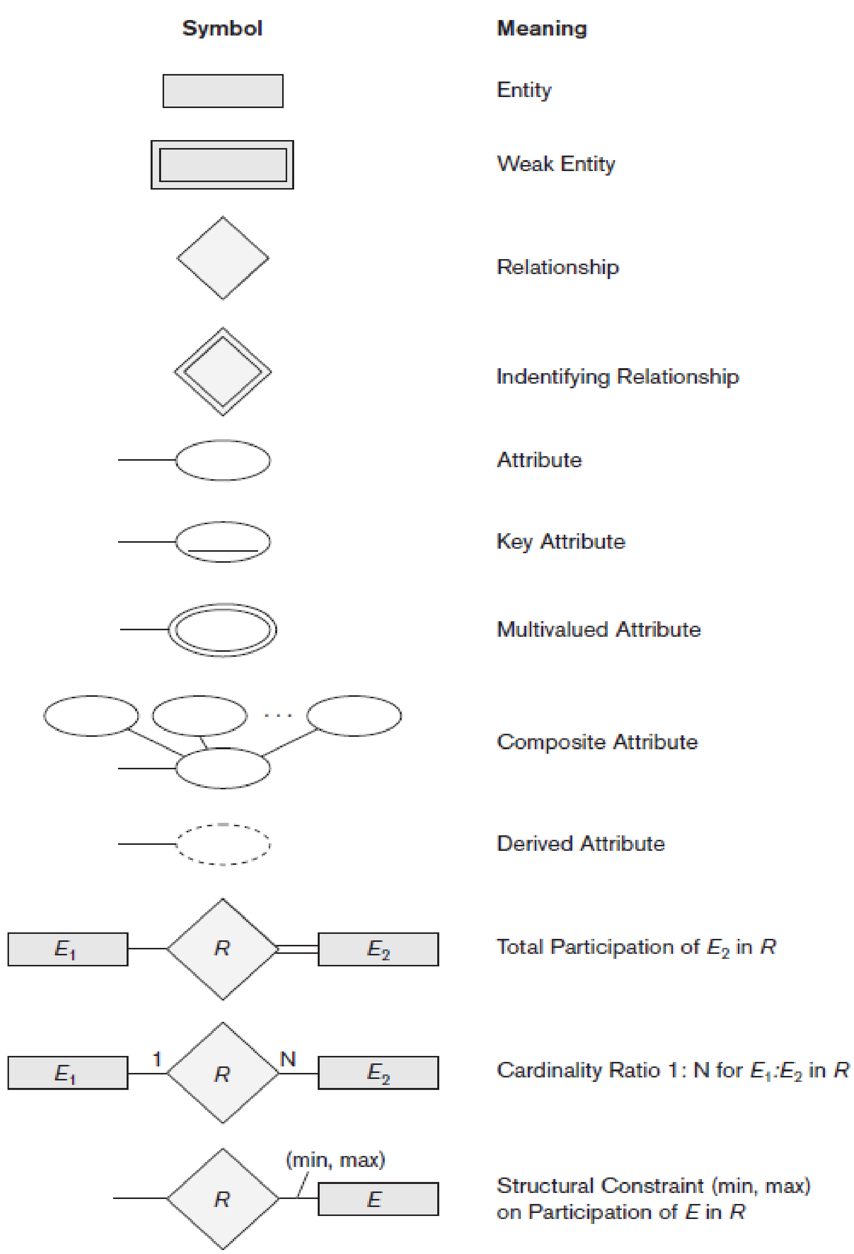


Figure 7.14
Summary of the notation for ER diagrams.

Department Of Computer Science
B.Sc-I
Semester-II
Subject-Relational DataBase Management System
Unit-I
DataBase Design Entity Relationship(ER)

Construction Of ER Diagram :

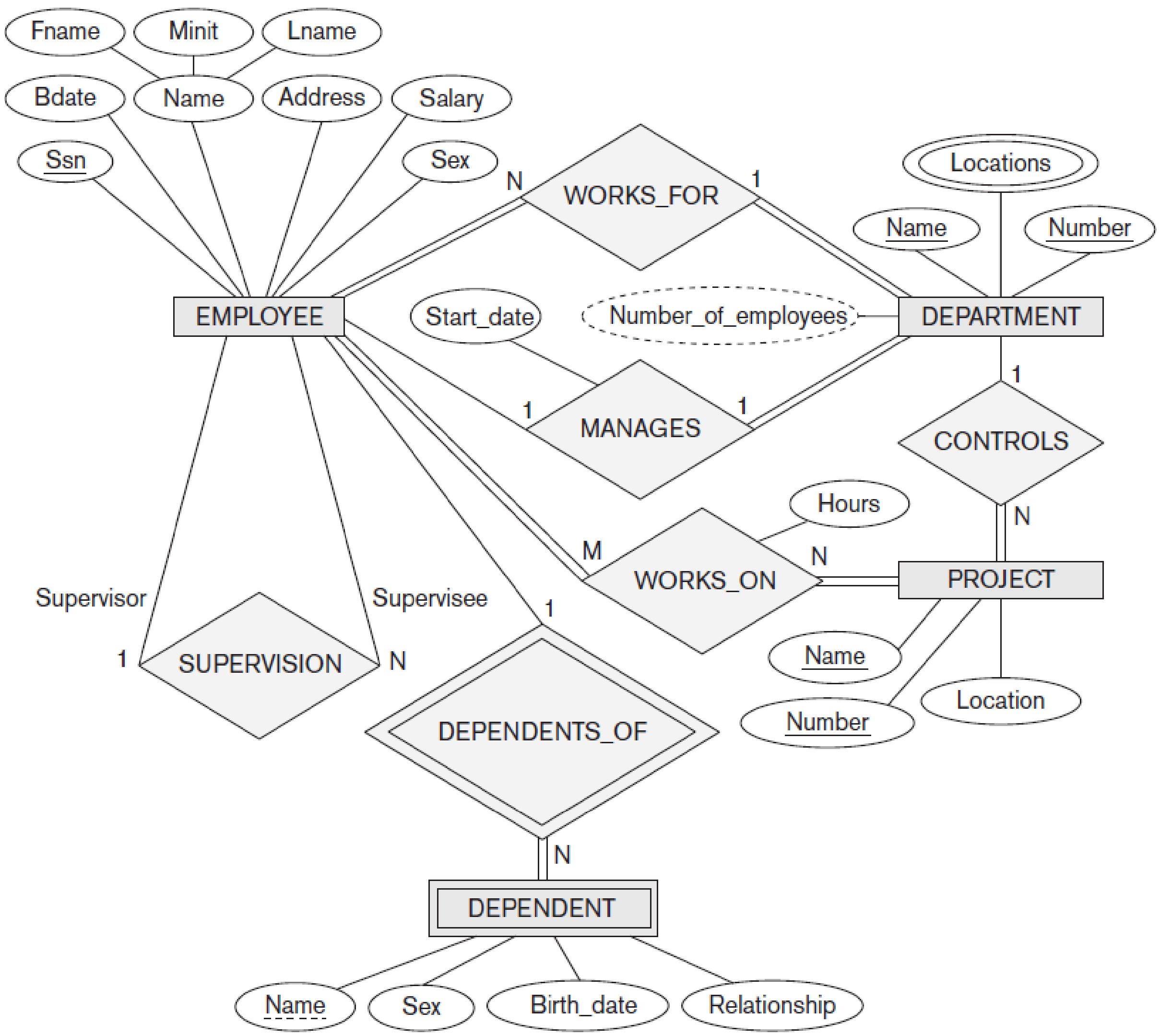
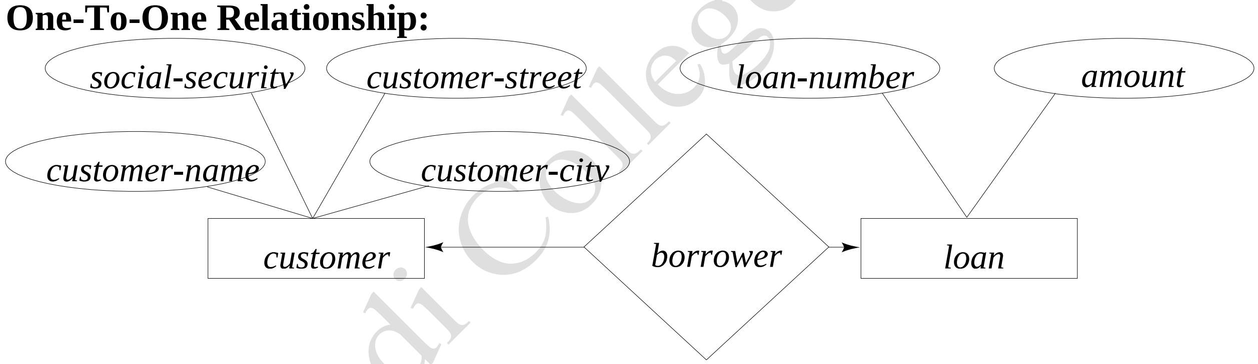


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

ER to Relational Mapping

Express the number of entities to which another entity can be associated via a relationship set.
Most useful in describing binary relationship sets.
For a binary relationship set the mapping cardinality must be one of the following types:
One to one
One to many
Many to one
Many to many
We distinguish among these types by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

One-To-One Relationship:



A customer is associated with at most one loan via the relationship borrower

A loan is associated with at most one customer via borrower

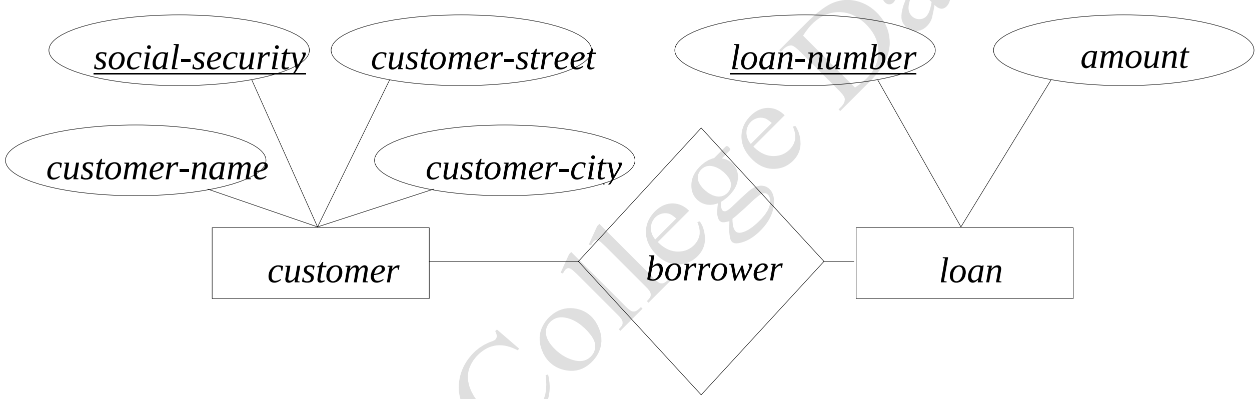
One-To-Many and Many-to-One Relationship

In the one-to-many relationship (a), a loan is associated with at most one customer via borrower; a customer is associated with several (including 0) loans via borrower

In the many-to-one relationship (b), a loan is associated with several (including 0) customers via borrower; a customer is associated with at most one loan via borrower

Many-To-Many Relationship:

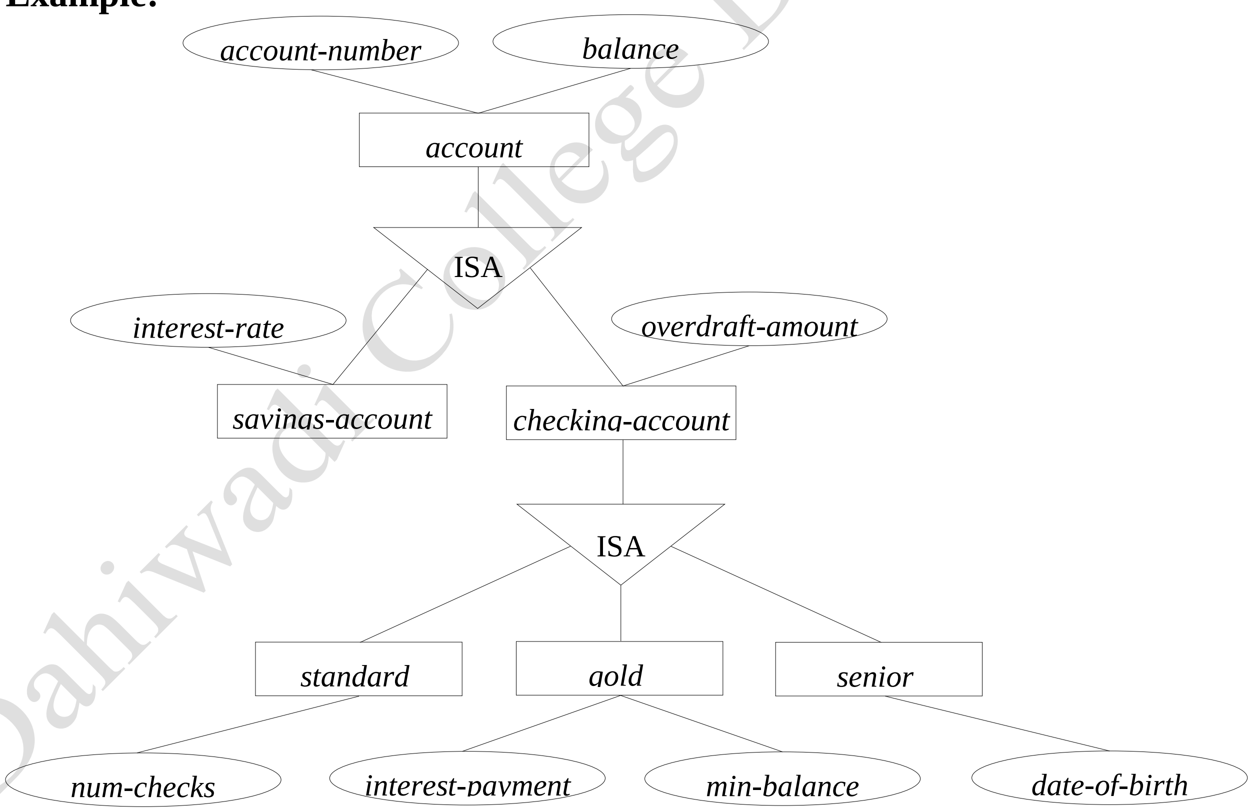
A customer is associated with several (possibly 0) loans via borrower • A loan is associated with several (possibly 0) customers via borrower



Relationship Diagram : Specialization, Generation and Aggregation

- ### Specialization:

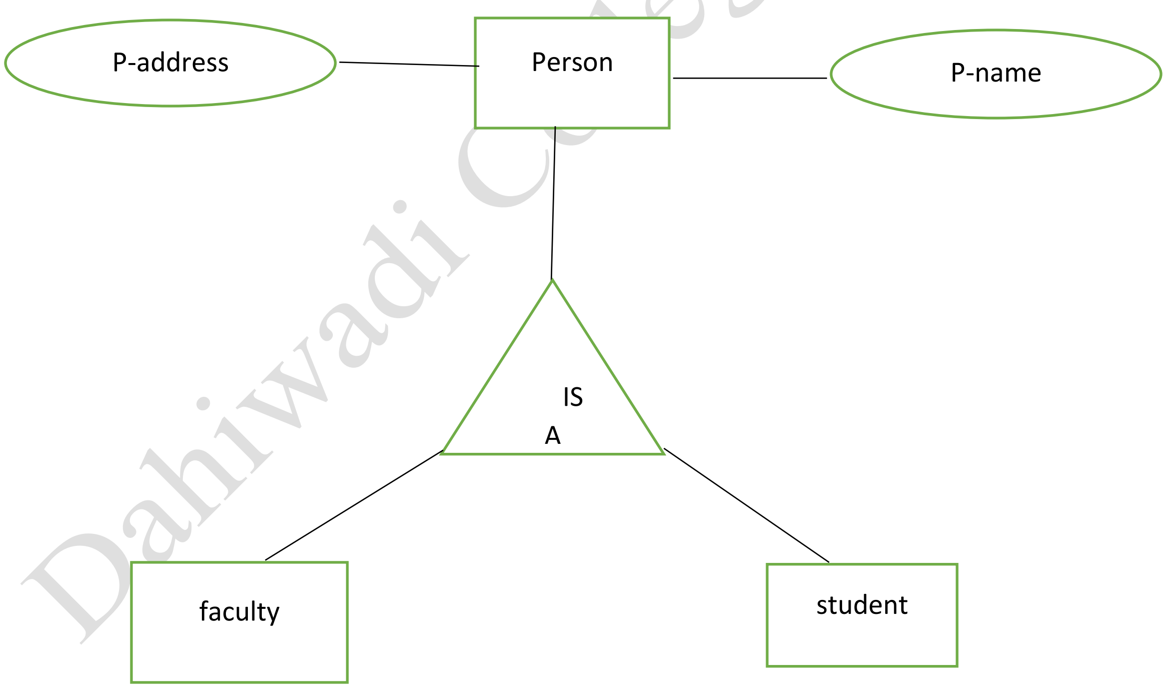
Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
Depicted by a triangle component labeled ISA (i.e., savings-account"is an" account)
Example:



- **Generalization:**

A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set
Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
Attribute Inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it

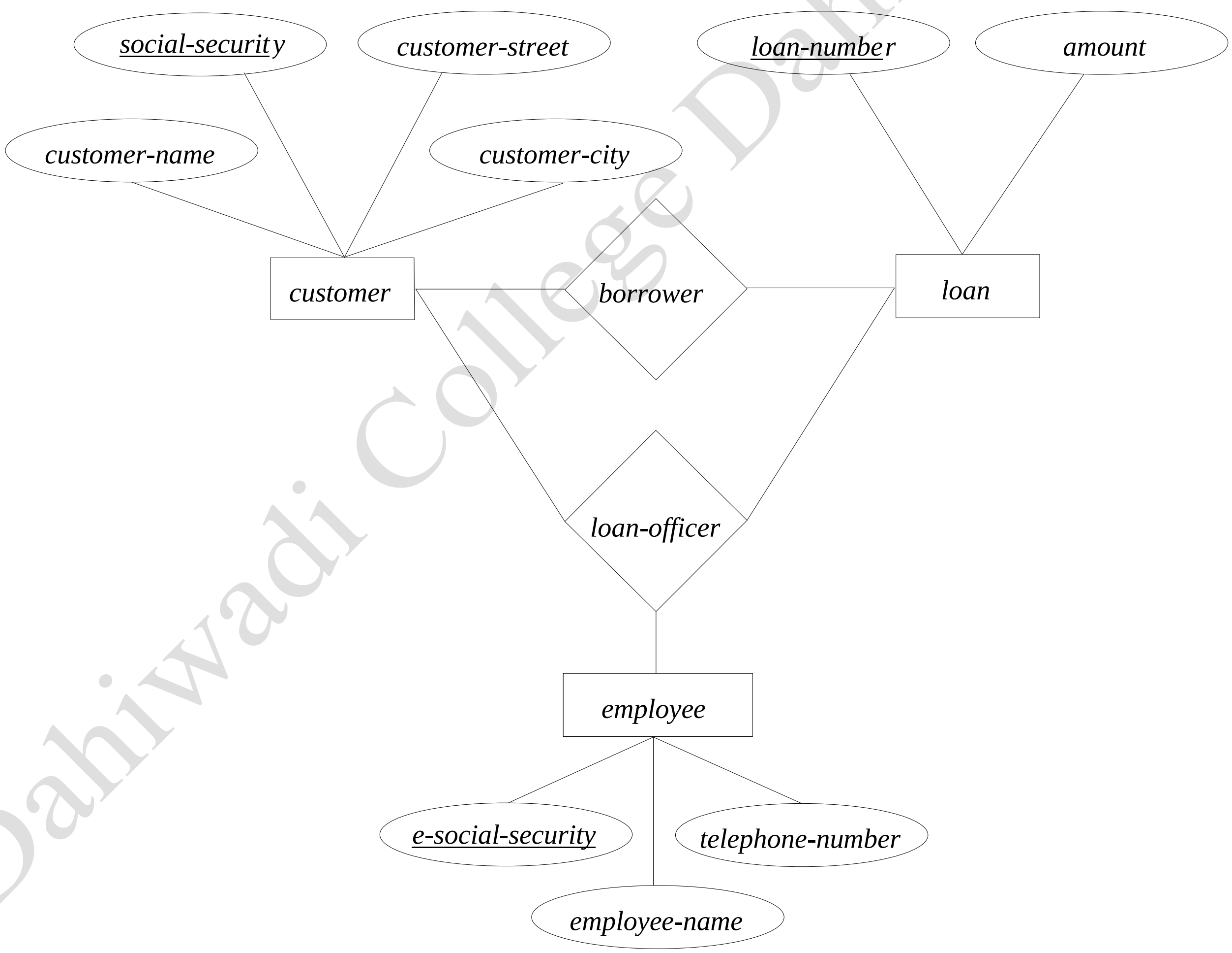
Example:



Aggregation:

• Loan customers may be advised by a loan-officer.

Example:



Functional Dependency:
A functional dependency (abbreviated as FD or f.d.), denoted by X ! Y ,
between two sets of attributes X and Y that are subsets of R = fA1;A2; : : : ;Ang
speci_es a constraint on the possible tuples that can form a relation state r of R.
The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], we
must also have t1[Y] = t2[Y].
_ X ! Y : X functionally determines Y or Y is functionally dependent on X
X functionally determines Y in a relation schema R if and only if , whenever two
tuples
of r(R) agree on their X-values, they must necessarily agree on their Y -values.
{ If X is a candidate key of R, this implies that X ! Y for any subset of attributes
Y of R.
{ If X ! Y in R, this does not say whether or not Y ! X in R.
_ A functional dependency is a constraint that any relation extensions r(R) must
satisfy
the functional dependency constraint at all times.
Candidate Keys and Superkeys
Let R = (U; F) be a relation schema, where U is the set of attributes and F is the set
of
functional dependencies.
_ A subset X of U is a superkey for a relation schema R if (X ! U) is in F +, i.e.,
X+
F = U.
_ A subset X of U is a candidate key for R if X is a superkey and no proper subset
Y
of X is a superkey.
That is, if X is a candidate key , then (X+
F = U) and (69 Y such that Y _ X and Y +
F =
U).
_ Example: Is (ABE) a superkey for R = fU; Fg, where U = fA;B;C;D;E; Ig and
F = fA ! D; AD ! E; BI ! E; CD ! I; E ! Cg
{ Is F j= X ! U ?
(ABE)+

F = ABEDCI = U, therefore, ABE ! U and ABE is a superkey for R.
_ Example: Is (ABE) a candidate key for R ? Where R is as previous example.
{ We know ABE is a superkey.
{ A+
F = ADECI 6= U
B+
F = B 6= U
E+
F = EC 6= U
AB+
F = ABDECI = U) ABE is not a candidate key.
AE+
F =
BE+
F =
ABE+
F =
_ How about _finding all candidate keys for R = fU; Fg, where U = fA;B;C;D;Eg and
F = fAB ! E; E ! AB; EC ! Dg.

Types of Keys:

- **Composite key**: Key that is composed of more than one attribute
- **Key attribute**: Attribute that is a part of a key
- **Entity integrity**: Condition in which each row in the
  - table has its own unique identity
  - _ All of the values in the primary key must be unique
  - _ No key attribute in the primary key can contain a null

Department Of Computer Science
B.Sc-I
Semester-II
Subject-Relational DataBase Management System
Unit-I
DataBase Design Entity Relationship(ER)

Normalization of Relations:

_ The normalization process takes a relation schema through a series of tests to \certify" whether it satisfies a certain normal form. If the schema does not meet the normal form test, the relation is decomposed into smaller relation schemas that meet
the tests.
_ The purpose of normalization is to analyze the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy
and (2) minimizing the insertion, deletion, and update anomalies.
_ 1NF, 2NF, 3NF, and BCNF are based on the functional dependencies among the attributes of a relation.
_ Database design as praticed in industry today pays particular attention to normaliza-
tion only up to 3NF or BCNF (sometimes 4NF).

First Normal Form

- _ 1NF: the domain of an attribute must include only atomic values and the value of any attribute in a tuple must be a single value from the domain of that attribute.

Second Normal Form

- _ 2NF is based on the concept of fully functional dependency.
- _ X ! Y is a full functional dependency if for any attribute A 2 X; (X□fAg) 6! Y .
- _ X ! Y is a partial functional dependency if for some attribute A 2 X; (X □
- fAg) ! Y .
- _ 2NF: a relation schema R is in 2NF if every nonprime attribute A in R is fully functionally dependent on the primary key of R.

Third Normal Form

Department Of Computer Science
B.Sc-I
Semester-II
Subject-Relational DataBase Management System
Unit-I
DataBase Design Entity Relationship(ER)

- _ 3NF is based on the concept of transitive dependency.
- _ X ! Y in a relation schema R is a transitive dependency if there is a set of
- attributes Z that is neither a candidate key nor a subset of any key of R, and both
- X ! Z and Z ! Y hold.
- _ 3NF: a relation schema R is in 3NF if it satises 2NF and no nonprime attribute of R
    - is transitively dependent on the primary key.

Boyce-Codd Normal Form

- _ BCNF is a stronger normal form than 3NF. That is, every relation in BCNF is also in
- 3NF; however, a relation in 3NF is not necessarily in BCNF.
- _ BCNF: a relation schema R is in BCNF if whenever a nontrivial functional dependency
- X ! A holds in R, then X is a superkey of R.