**Chapter 3: Database Design** - conceptual database design; **Application Design** - programs and interfaces that access the database. **Methodologies for Conceptual DB Design**: ER, EER, Design Tools, UML. **Entity**: object (thing) with independent physical or conceptual existence in the real world. specific entity - will have a value for each of its attributes. **Entity type** - collection of entities that have same attributes. can have more than one key. **Entity Set** - collection of all entities in a particular entity type in database at any point in time. (Entity Collection) current state of the entities of that type that are stored in the database. **Attribute**: each real-world entity (thing) has certain properties that represent its significance in real world or describes it, these properties of entity are known as attribute. **Attribute value**: associated with each real-world entity are certain attribute that described that entity value of these attributes for any entity is called attribute value. **Simple Attribute**: Each entity has a single atomic value for the attribute. **Composite attribute**: The attribute may be composed of several components. **Multi valued**: An entity may have multiple values for that attribute. **Derived attribute** - attribute whose value is calculated (derived) from other attributes. need not be physically stored within the database; instead, it can be derived by using an algorithm. **Key attribute** - An attribute of an entity type for which each entity must have a unique value. an entity type has one or more attributes whose values are distinct for each individual entity in an entity set can be composite. **Value set (domain)** - set of values associated with an attribute. (Datatypes). **Relationship** - relates two or more distinct entities with a specific meaning. **Relationship type** - Relationships of the same type are grouped or typed into a relationship type. Also, a) Is the schema description of a relationship b) Identifies the relationship name and the participating entity types c) Also identifies certain relationship constraints. **Relationship set** - The current set of relationship instances represented in the database. The current state of a relationship type. **Participation role** - Each entity type that participates in a relationship type plays a particular role in the relationship. **Recursive relationships** - It necessary to use role names in the description of relationship types, in some cases the same entity type participates more than once in relationship type in different roles. In such cases the role name is essential for distinguishing the meaning of each participation. A relationship type between the same participating entity type in distinct roles. **ER model** has **three main concepts**: Entities, Attributes, Relationships. **Constraints on Relationship Types (ratio constraints):** a) **Cardinality Ratio** (maximum participation) 🡪 1:1, 1:N or N:1, M:N. b) **Existence Dependency Constraint** (specifies minimum participation) (also called **participation constraint**) 🡪 i) zero (optional participation, not existence-dependent) ii) one or more (mandatory participation, existence-dependent). **Weak entity type** - An entity that does not have a key attribute and that is identification dependent on another entity type must participate in an identifying relationship type. Weak entities are identified by the combination of: i) A partial key of the weak entity type ii) The particular entity they are related to in the identifying relationship type. **UML**: Represent classes (similar to entity types) as large rounded boxes with three sections: i) entity type (class) name ii) Attributes iii) class operations (operations are not in basic ER model)

**Chapter 4: EER Model Concepts** a) Includes all modeling concepts of basic ER b) Additional concepts: i) Subclasses/superclasses ii) Specialization/generalization iii) categories (UNION types) iv) attribute and relationship inheritance c) Constraints on Specialization/Generalization. **Subclass**-an entity type has numerous subgroupings or subtypes of its entities that are meaningful. - IS-A relationships SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE-Attributes of a subclass are called **specific or local attributes**. An entity that is member of a subclass inherits a) All attributes of the entity as a member of the superclass b) All relationships of the entity as a member of the superclass. **Super Class**: A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class. **Specialization** - Process of defining a set of subclasses of a superclass. top down conceptual refinement process. Types are a) **Predicate-defined** (or condition-defined): based on some predicate. E.g., based on value of an attribute, say, Job-type, or Age. b) **Attribute-defined**: shows the name of the attribute next to the line drawn from the superclass toward the subclasses. c) **User-defined**: membership is defined by the user on an entity by entity basis. **Generalization** - The reverse of the specialization process. Several classes with common features are generalized into a superclass; original classes become its subclasses. Note: Generalization usually is total because the superclass is derived from the subclasses. -bottom up conceptual synthesis process. **Two basic constraints can apply to a specialization/generalization**: a) **Disjointness Constraint** -subclasses of the specialization must be disjoint. an entity can be a member of at most one of the subclasses of the specialization. -Overlapping - same entity may be a member of more than one subclass of the specialization. b) **Completeness Constraint**-Exhaustiveness. Total specifies that every entity in the superclass must be a member of some subclass. -Partial allows an entity not to belong to any of the subclasses. **Four types of specialization/generalization**: a) Disjoint, total b) Disjoint, partial c) Overlapping, total d) Overlapping, partial. **Hierarchy**: -every subclass has only one superclass (called single inheritance); this is basically a tree structure. **Lattice**-A subclass with more than one superclass is called a shared subclass (multiple inheritance). **Category (UNION TYPE)** -a subclass will represent a collection of entities that is a subset of the UNION of entities from distinct entity types; -we call such a subclass a union type or a category. -A category member must exist in at least one (typically just one) of its superclasses. **Specific (local) 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 (or local) attributes of the subclass. **Specific relationships**- a subclass can participate in specific relationship types such as the HOURLY\_EMPLOYEE subclass participating in the BELONGS\_TO relationship. **Difference from shared subclass**, which is a: a) subset of the intersection of its superclasses b) shared subclass member must exist in all of its superclasses. **Knowledge Representation** - Deals with modeling and representing a certain domain of knowledge. **COMMON FEATURES between KR and Data Models**: a) Both use similar set of abstractions – classification, aggregation, generalization, and identification. B) Both provide concepts, relationships, constraints, operations and languages to represent knowledge and model data. DIFFERENCES: -KR has broader scope: tries to deal with missing and incomplete knowledge, default and common-sense knowledge etc. -include rules and reasoning mechanisms for inferencing -involve data and metadata. -used in conjunction with artificial intelligence systems to do decision support applications. **TYPES OF DATA ABSTRACTIONS**: a) CLASSIFICATION and INSTANTIATION b) AGGREGATION and ASSOCIATION (relationships) c) GENERALIZATION and SPECIALIZATION d) IDENTIFICATION. **CONSTRAINTS**: a) CARDINALITY (Min and Max) b) COVERAGE (Total vs. Partial, and Exclusive (Disjoint) vs. Overlapping. **Specification** - language and vocabulary (data model concepts) used. **Conceptualization** - description (schema) of the concepts of a particular field of knowledge and the relationships among these concepts. **Ontology** - set of concepts and interrelationships among those concepts.

**Chapter 5:** **Relational Model**-based on the concept of a Relation. **Relation** -concept based on the ideas of sets. -table of values, set of rows (rows are called tuples) -elements in each row represents a real-world entity or relationship -column header is attribute (attribute name) - gives meaning. **Key** -uniquely identifies each row in a table. artificial key or surrogate key- row-ids or sequential numbers to identify the rows in a table. **Schema** -name of a relation. Tuple -ordered set of values. Values are atomic(indivisible). -null value represents values that are unknown or not available or inapplicable. Domain -contains the set of all possible values the attribute can take. **Relation State** -subset of the Cartesian product of the domains of its attributes. **Constraints** -determine which values are permissible and which are not in the database. -conditions that must hold on all valid relation states. **1)** **Inherent or Implicit Constraints** -based on the data model itself. (E.g., relational model - does not allow a list as a value for any attribute). **2)** **Schema-based or Explicit Constraints**-Expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model). -**three main types** are a) Key b) Entity Integrity constraints c) Referential integrity constraints. **3) Application based or semantic constraints** - These are beyond the expressive power of the model and must be specified and enforced by the application programs. Must be specified and enforced by the application programs. A constraint specification language expresses these. - Eg: CREATE TRIGGER and CREATE ASSERTION. **1) Key Constraints**: **Superkey** - Is a set of attributes SK of R with the following condition: a) No two tuples in any valid relation state r(R) will have the same value for SK b) That is, for any distinct tuples t1 and t2 in r(R), t1[SK] ≠ t2[SK] c) This condition must hold in any valid state r(R). **Key** - A "minimal" superkey. That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property). A Key is a Superkey but not vice versa. **Primary Key** - uniquely identify each tuple in a relation. Provides tuple identity. smallest of candidate key. **2) Entity integrity**- primary key cannot have null values in any relation schema. **3) Referential integrity** - constraint involving two relations. (Foreign Key - can be value of a PK or null) - specify a relationship among tuples between referencing relation and the referenced relation & domain constraint is a schema-based constraint. **Relational Database Schema** -Set of relation schemas of same DB. (Name of DB). **Relational database state** -a set of relation states. (relational database snapshot). union of all the individual relation states. **Basic Operations for changing DB**- Integrity constraints should not be violated. (Update Actions – Cancel (REJECT/RESTRICT), Inform User, Trigger (CASCADE, SET NULL), Error Correction by User Operations). INSERT - May violate Domain/Key/Referential/Entity Constraints. DELETE - May violate only Referential Integrity Constraint. Remedied by RESTRICT (reject deletion), CASCADE(Propagate), SET NULL (FK to Null). MODIFY - May violate domain constraint and NOT NULL. **Updating the primary key (PK):** Similar to a DELETE followed by an INSERT. Need to specify similar options to DELETE. **Updating a foreign key (FK):** May violate referential integrity. **Updating an ordinary attribute** (neither PK nor FK): Can only violate domain constraints.

**Chapter 6: SQL**: Comprehensive language for relational database management. **CREATE statement**: Main SQL command for data definition. **SQL schema**: Identified by a schema name. Includes an authorization identifier and descriptors for each element. Schema elements include Tables, constraints, views, domains, and other constructs. **Catalog:** Named collection of schemas. **Specifying a new relation** 1) Provide name of table 2) Specify attributes, their types and initial constraints. **Base tables (relations):** Relation and its tuples are actually created and stored as a file by the DBM. SQL Standards: core specification and specialized extensions. **Virtual relations (views)**: Created through the CREATE VIEW statement. Do not correspond to any physical file. **Basic data types**: **Numeric data types**: INTEGER, INT, and SMALLINT, FLOAT or REAL, and DOUBLE PRECISION. **Character-string**: CHAR(n), CHARACTER(n), VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n). **Bit-string**: BIT(n), BIT VARYING(n). **Boolean**: TRUE or FALSE or NULL. **DATE**: YYYY-MM-DD (Mapping Functions avalaible in DBMS). **Timestamp**: Includes the DATE and TIME also TIMEZONE. **INTERVAL**: increment or decrement an absolute value of a date, time, or timestamp. **Domain**: Name used with the attribute specification. Makes it easier to change the data type for a domain that is used by numerous attributes. Improves schema readability. Example: CREATE DOMAIN SSN\_TYPE AS CHAR(9). **UDT's: TYPE**: User Defined Types are supported for object-oriented applications. **Basic Constraints**: **Key constraint**: A primary key value cannot be duplicated. Specifies one or more attributes that make up the primary key of a relation. UNIQUE clause Specifies alternate (secondary) keys (called CANDIDATE keys in the relational model). **Entity Integrity Constraint**: A primary key value cannot be null. **Referential integrity constraints**: “foreign key” must have a value that is already present as a primary key, or may be null. **PRIMARY KEY clause**: Specifies one or more attributes that make up the primary key of a relation. **UNIQUE clause**: Specifies alternate (secondary) keys (called CANDIDATE keys in the relational model). **FOREIGN KEY clause**: Default operation is reject update on violation. Attach referential triggered action clause a. Options include SET NULL, CASCADE, and SET DEFAULT b. Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE c. CASCADE option suitable for “relationship" relations**.** **CHECK clauses** at the end of a CREATE TABLE statement- Apply to each tuple individually. CHECK (Dept\_create\_date <= Mgr\_start\_date); **Attribute Constraints**: DEFAULT <value> , NULL (NOT NULL), CHECK (Apply to each tuple individually). **Using the keyword CONSTRAINT**: a. Name a constraint b. Useful for later altering. **SELECT statement**: One basic statement for retrieving information from a database. **Projection attributes** are: a) Attributes whose values are to be retrieved b) **Selection condition**: Boolean condition that must be true for any retrieved tuple. Selection conditions include join conditions when multiple relations are involved.

**Query 0**. Retrieve the birth date and address of the employee(s) whose name is ‘John B. Smith’- Q0: **SELECT Bdate, Address FROM EMPLOYEE WHERE Fname = ‘John’ AND Minit = ‘B’ AND Lname = ‘Smith’**; **Query 1**. Retrieve the name and address of all employees who work for the ‘Research’ department. Q1: **SELECT Fname, Lname, Address FROM EMPLOYEE, DEPARTMENT WHERE Dname = ‘Research’ AND Dnumber = Dno**; We can also write **SELECT EMPLOYEE.Fname, EMPLOYEE.LName, EMPLOYEE.Address FROM EMPLOYEE, DEPARTMENT WHERE DEPARTMENT.DName = ‘Research’ AND DEPARTMENT.Dnumber = EMPLOYEE.Dno**;

**Aliases or tuple variables**: Declare alternative relation names. Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables. asterisk (\*):Retrieve all the attribute values of the selected tuples. \* can be prefixed by the relation name. **DISTINCT**: Only distinct tuples should remain in the result. **Set operations**: UNION, EXCEPT (difference), INTERSECT. **Corresponding multiset operations**: UNION ALL, EXCEPT ALL, INTERSECT ALL. Type compatibility is needed for these operations to be valid. **Query 4**. Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project. **Q4A: ( SELECT DISTINCT Pnumber FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE Dnum = Dnumber AND Mgr\_ssn = Ssn AND Lname = ‘Smith’ ) UNION ( SELECT DISTINCT Pnumber FROM PROJECT, WORKS\_ON, EMPLOYEE WHERE Pnumber = Pno AND Essn = Ssn AND Lname = ‘Smith’ )**; **LIKE (comparison operator)**: Used for string pattern matching. % replaces an arbitrary number of zero or more characters. underscore (\_) replaces a single character. **BETWEEN** is also a comparison operator. **WHERE(Salary BETWEEN 30000 AND 40000) AND Dno = 5**; **Query 13.** Show the resulting salaries if every employee working on the ‘ProductX’ project is given a 10 percent raise. **Q13: SELECT E.Fname, E.Lname, 1.1 \* E.Salary AS Increased\_sal FROM EMPLOYEE AS E, WORKS\_ON AS W, PROJECT AS P WHERE E.Ssn = W.Essn AND W.Pno**; **ORDER BY clause**: DESC result in descending order of values. ASC result in ascending order. Typically placed at the end of the query. **INSERT** typically inserts a tuple (row) in a relation (table). INSERT is used for bulk-loading. **UPDATE** may update a number of tuples (rows) in a relation (table) that satisfy the condition. DELETE may also update a number of tuples (rows) in a relation (table) that satisfy the condition. **Additional Features of SQL**: CREATE INDEX, Transaction control commands; granting and revoking of privileges; Constructs for creating trigger; User Defined Types (CREATE TYPE); XML and OLAP.

**Chapter 5,6,9 Review**: **Domain:** set of atomic values. Each value in domain is indivisible. contains the set of all valid values the attribute can take. **Attribute**: Each column has a column header that gives an indication of the meaning of the data items in that column. In the formal model, the column header is called an attribute name (or just attribute). **n-tuple**: if a tuple has n attributes. **relation schema**: Is made up of relation name and a list of attributes A1.....An. **Relation State**: subset of the Cartesian product of the domains of its attributes. **Degree of a relation**: is the no of attributes of its relational schema. (Arity). **Relational Database Schema**: set of Tables, columns and relationships. Relational Database State: State of the set of relations at any point of time. **Tuples in a relation not ordered**: A relation is defined as a set of tuples. Mathematically, elements of a set have no order among them; hence, tuples in a relation do not have any particular order. **Duplicate tuples not allowed in a relation**: Because it violates the specifications of the relational integrity constraints, particularly the key constraint which states that no two tuples can have the same values for their attributes at any relation state of a database. **Key and a superkey**: **Key**: uniquely identifies each row in a table. **Superkey**: set of attributes of a Relation schema. No two tuples in a valid r(R) will have same value for SK. **Candidate keys of a relation to be the primary key**? it becomes fairly easier to deal with a database when we can have a single distinct key for a particular relation instead of having more than one key. **Discuss the characteristics of relations that make them different from ordinary tables and files**: The first is that a relation is not sensitive to the ordering of tuples. The second involves the ordering of attributes in a relation schema and the corresponding ordering of values within a tuple. **Occurrence of NULL values in relations**: A special null value is used to represent values that are unknown or not available or inapplicable in certain tuples. NULL values should be avoided as much as possible. **Entity Integrity**: The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to identify the individual tuples. **Referential Integrity**: A constraint involving two relations. **Foreign Key**: Used to relate two tuples from different relations and attribute present in referencing relation is foreign key. **Transaction**: an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database. At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema. Update operation is performed on the tuple(s) of a relation whereas transaction is performed on the database.

**SELECT <attribute list> FROM <Table List> WHERE <condition> ORDER BY <attribute list>** **Select**: list of attribute names to be received by the query **From**: the tables that these attributes with be retrieved from **Where**: conditional Boolean expression to identify certain tuples to be retrieved (optional) **Order by**: attribute list to order the result by (optional). **SQL differ from relation**: sql allows a table (relation) to have 2 or more tuples that are identical in all their attri values. Sql tables is not set of tuples becoz a set does not allow 2 identical members. It is a multiset of tuples. Some SQL relations are constrained to be sets because a key constraint has been declared or becoz of DISTINCT option has been used in SELECT stmt. In relation is a set of tuples that is same values are not allowed for any tuple.

**Entity type** - entity relation, 1:1 or 1:10 relationship type- foreign key (relationship relation), M:N relationship type - relationship relation and 2 foreign keys, n-ary relationship type – relationship relation foreign keys, simple attribute-attribute, composite attribute -set of simple component attributes, multivalued attribute-relation and foreign key, value set-domain, key attribute -primary or secondary key.

**Entity integrity & referential integrity constraints**: Entity integrity constraints: In this, it states that no PK value can be NULL as PK value is used to identify individual tuples in a relation. And, having NULL values implies that we can't identify some tuples. **Referential integrity constraints:** It is specified between 2 relations & is used to maintain consistency among tuples in the 2 relations & referential integrity constraints states that, a tuple in one relation that refers to another relation & must refer to an existing tuple in that relation.

**Referential triggered actions**: Schema designer can specify an alternative action to taken when a referential integrity constraint is violated by attaching a referential triggered action clause to any FK constraint.

**ER-to-Relational mapping**:1: Mapping of Regular Entity Types 2: Weak entity types 3: Binary 1:1 relationship types 4: Binary 1:N relationship types 5: Binary M:N relationship types 6: Multivalued attributes 7: N-ary relationship types. **EER-to-Relational mapping**: 8: Specialization & Generalization 8A. Multiple relations-Superclass & subclasses. 8B. Multiple relations-subclass relations only. 8C. Single relation with one type attribute. 8D. Single relation with multiple type attributes. 9: Union types.

**Chapter 3,4 Review**: subclass and why needed? an entity type has numerous subgroupings or subtypes of its entities that are meaningful and need to be represented explicitly because of their significance to the database application. Subset of the entities that belongs to the super class entity set. Subgroupings of entities that are meaningful. Subclass can define:Specific attributes, Specific relationship types, easy way to define inheritance between two entity types. **Superclass of a subclass, superclass/subclass relationship, IS-A relationship, specialization, generalization, category, specific (local) attributes, and specific relationships**. A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class. state an example. subclass inherits the attributes of super class. Secretary is-a employee & technician is-a employee. specialization - Specialization is the process of defining a set of subclasses of a superclass. **Generalization** is the reverse of the specialization process. Several classes with common features are generalized into a superclass;original classes become its subclasses. Supress the differences between entity types(subclasses) and identify the common features in them and generlaize them into a super class. category(UNION TYPE) - a subclass will represent a collection of entities that is a subset of the UNION of entities from distinct entity types; we call such a subclass a union type or a category. specific (local) 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 (or local) attributes of the subclass. specific relationships - a subclass can participate in specific relationship types, such as the HOURLY\_EMPLOYEE subclass participating in the BELONGS\_TO relationship. **Inheritance** - when a certain entity can inherit traits from a superclass or entity that already has attributes the subclass should have. It is useful to avoid redundancy and repetitiveness. **3.1 1**-High-level data model do not include implementation details, they are easy to understand and are useful in communicating with non-technical users. 2- it can also be used as reference to ensure that all user requirements are met and that requirements do not conflict with each other. 3- this also enables database designers to concentrate on specifying the properties of data without being concerned with storage details. **3.2** use of NULL value: 1-When value of an attribute is irrelevant for an entity For example: in a schema that stores information about a person if we have an attribute calledcompany, which sores the company name where a person works. Now for a student who is not working, this attribute value will be irrelevant, so we can put in a NULL value at its place. 2-When value of attribute is not known, either because it is not known that value for attribute exist or because existing value is unknown, then we can put NULL as value. **3.7** participation role - Each entity type that participates in a relationship type plays a particular role in the relationship. It necessary to use role names in the description of relationship types, in some cases the same entity type participates more than once in relationship type in different roles. In such cases the role name become essential for distinguishing the meaning of each participation. Such relationship types are called recursive relationships.

**Query 2**. For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date. Q2**: SELECT Pnumber, Dnum, Lname, Address, Bdate FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE Dnum = Dnumber AND Mgr\_ssn = Ssn AND Plocation = ‘Stafford’**. **Query 8**. For each employee, retrieve the employee’s first and last name and the first and last name of his or her immediate supervisor. Q8: **SELECT E.Fname, E.Lname, S.Fname, S.Lname FROM EMPLOYEE AS E, EMPLOYEE AS S WHERE E.Super\_ssn = S.Ssn**;

**Step 1: Mapping of Regular Entity Types**. ◼ For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. ◼ Choose one of the key attributes of E as the primary key for R. ◼ If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R. ◼ **Step 2: Mapping of Weak Entity Types** ◼ For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R. ◼ Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s). ◼ The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any. ◼ **Step 3: Mapping of Binary 1:1 Relation Types** ◼ For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. ◼ There are three possible approaches: 1. **Foreign Key ( 2 relations) approach**: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. 2. **Merged relation (1 relation) option**: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total. 3. Cross-reference or relationship relation ( 3 relations) option: The third alternative is to set up a third relation R for the purpose of crossreferencing the primary keys of the two relations S and T representing the entity types. ◼ **Step 4: Mapping of Binary 1:N Relationship Types**. ◼ For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. ◼ Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R. ◼ Include any simple attributes of the 1:N relation type as attributes of S. ◼ **Step 5: Mapping of Binary M:N Relationship Types.** ◼ For each regular binary M:N relationship type R, create a new relation S to represent R. This is a relationship relation. ◼ Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. ◼ Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. ◼ **Step 6: Mapping of Multivalued attributes.** ◼ For each multivalued attribute A, create a new relation R. ◼ This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute. ◼ The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components. **◼ Step 7: Mapping of N-ary Relationship Types**. ◼ For each n-ary relationship type R, where n>2, create a new relationship S to represent R. ◼ Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. ◼ Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S. ◼ **Step8: Options for Mapping Specialization or Generalization.** **Option 8A: Multiple relations-Superclass and subclasses** ◼ Create a relation L for C with attributes Attrs(L) = {k,a1,…an} and PK(L) = k. Create a relation Li for each subclass Si, 1 < i < m, with the attributesAttrs(Li) = {k} U {attributes of Si} and PK(Li)=k. This option works for any specialization (total or partial, disjoint of over-lapping). ◼ **Option 8B: Multiple relations-Subclass relations only** ◼ Create a relation Li for each subclass Si, 1 < i < m, with the attributes Attr(Li) = {attributes of Si} U {k,a1…,an} and PK(Li) = k. This option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses). ◼ **Option 8C: Single relation with one type attribute** ◼ Create a single relation L with attributes Attrs(L) = {k,a1,…an} U {attributes of S1} U…U {attributes of Sm} U {t} and PK(L) = k. The attribute t is called a type (or discriminating) attribute that indicates the subclass to which each tuple belongs ◼ **Option 8D: Single relation with multiple type attributes** ◼ Create a single relation schema L with attributes Attrs(L) = {k,a1,…an} U {attributes of S1} U…U {attributes of Sm} U {t1, t2,…,tm} and PK(L) = k. Each ti, 1 < I < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass Si.



