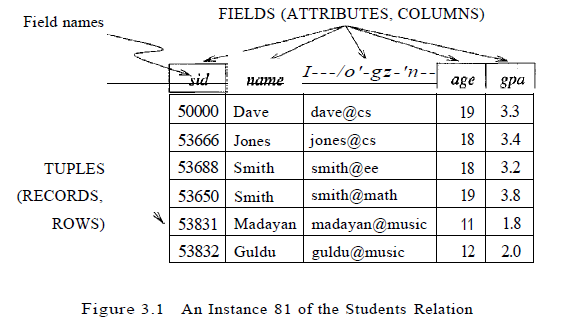
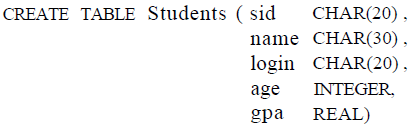
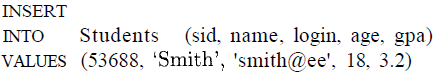
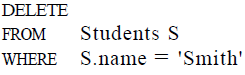
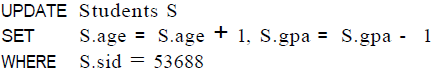
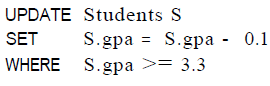
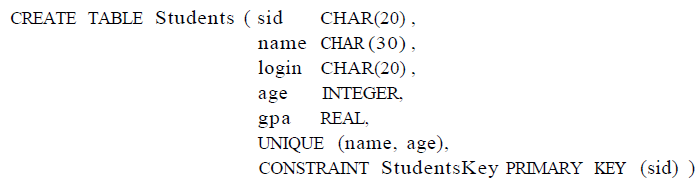
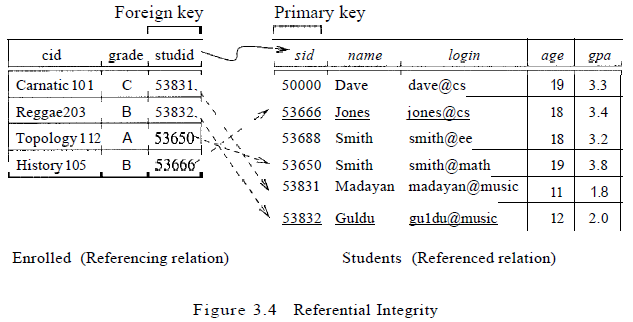
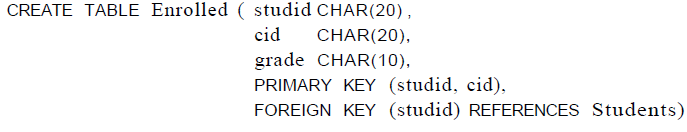
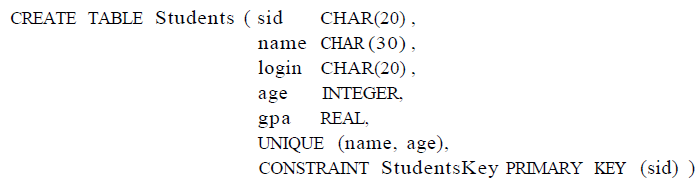
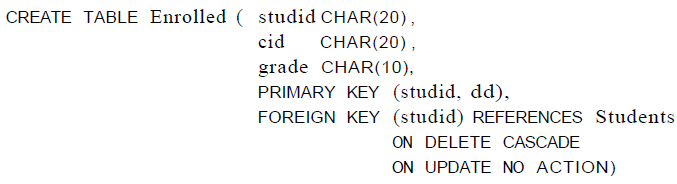
Relational model (Chapter 3)

* Definition
  + Schema
    - Specifies the relation’s name, the name of each filed and the domain of each filed
  + Domain
    - Is referred to in a relation schema by the domain name
    - Has a set of associated values
  + Instance of a relation
    - Set of tuples in which each tuple has the same number of fields as the relation schema
    - Relation instance that satisfies the domain constraints in the relation schema
    - 
  + Relation schema
    - Specifies the domain of each filed or column in the relation instance
    - 
    - Students (sid: string, name: string, login: string, age: integer, gpa: real)
  + Domain constraints
    - Specify an important condition that we want each instance of the relation to satisfy
  + 
    - |
      * Read as ‘such that’
    - E
      * Read as ‘in’
    - 
      * Condition that must be satisfied by the filed values of each tuple in the set
    - { … }
      * Denote a set
  + Degree of a relation
    - Number of fields
  + Cardinality of a relation instance
    - The number of tuples in the instance
  + Relational database
    - Collection of relations with distinct relation names
  + Relational database schema
    - The collection of schemas for the relations in the database
* Creating and modifying relations using SQL
  + SQL uses ‘table’ to denote relation
  + Definitions
    - Data Definition Language (DDL)
      * Subset of SQL that supports the creation, deletion, and modification of tables
  + SQL statements
    - CREATE TABLE
      * Is used to define a new table
      * 
    - INSERT
      * Is used to insert tuples
      * 
        + Can optionally omit the list of column names in the INTO clause and list the values in the appropriate order
    - DELETE
      * Is used to delete tuples
      * 
    - UPDATE
      * Is used to modify the column values in an existing row
      * , 
        + WHERE is applied first and determines which rows are to be modified
        + SET then determines how these rows are to be modified
* Integrity constraints over relations
  + Definition
    - Integrity constraint (Ie)
      * A condition specified on a database schema
      * Restricts the data that can be stored in a n instance of the database
    - Legal instance
      * Database instance that satisfies all the integrity constraints specified on the database schema
      * DBMS enforces integrity constraints that it permits only legal instances to be stored in the database
  + When to enforce Integrity constraints
    - When the DB or end user defines a database, he or she specifies ICs that must hold on any instance of this database
    - When a database application is run, DBMS checks for violations and disallows changes to the data that violate the specified ICs
* Key constraints
  + Definition
    - Key constraint
      * A statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple
  + Candidate key
    - Set of fields that uniquely identifies a tuple according to a key constraint
    - We often abbreviate candidate key to just key
    - Two distinct tuples in a legal instance cannot have identical values in all the fields of a key
    - No subset of the set of fields in a key is a unique identifier for a tuple
    - A relation may have several candidate keys but the key must identify tuples uniquely in all possible legal instance s of the relation
  + Superkey
    - A set of fields that contains a key
    - {StudentID, Name}
* Specifying key constraints in SQL
  + Statements
    - UNIQUE
      * Is used to declare that a subset of the columns of a table constitute a key
    - Primary KEY
      * At most one of these candidate keys can be declared to be a primary key
  + 
    - Sid is primary key
      * Can name a constraint by preceding it with CONSTRAINT constraint-name
    - Combination of name and age is key
    - If the constraint is violated, the constraint name is returned and can be used to identify error
* Foreign key constraints
  + Definition
    - Foreign key constraint
      * Most common IC involving two relations
      * An IC involving different relations for DBMS to check to keep the data of relations consistent
      * Foreign key in the referencing relation must match the primary key of the referenced relation
        + Foreign key must have the same number of columns and compatible data types, although the column names can be different
    - Example
      * 
        + Every studid value that appears in the instance of the Enrolled table appears in the primary key column of a row in the students table
        + Violations

Cannot insert a tuple with a new sid into Enrolled where there is no tuple in Students with the sid

Cannot delete a tuple with a sid in Students when a tuple with the sid in Enrolled exists

DBMS should disallow deletion or delete the Enrolled tuple that refers to the deleted Students tuple

* + Null value
    - Value in the field is either unknown or not applicable
    - Appearance of null in a foreign key field does not violate the foreign key constraint
    - Null values are not allowed to appear in a primary key
* Specifying foreign key constraints in SQL
  + 
    - Foreign key constraint states that every studid value in Enrolled must also appear in Students, that is, studid in Enrolled is a foreign key referencing Students
    - Primary key constraint for Enrolled states that a student has exactly one grade for each course he or she is enrolled
* General constraints
  + Constraints that go well beyond domain, key, or foreign key constraints can be specified
    - Every student whose age is greater than 18 must have a gpa greater than 3
  + Current relational database systems support such general constraints in the form of *table constraints* and *assertions*
    - Table constraints
      * Are associated with a single table
      * Are checked whenever that table is modified
    - Assertions
      * Involve several tables
      * Are checked whenever any of these tables is modified
* Enforcing integrity constraints
  + Insertion violation
    - Primary key constraint
      * Insertion of a new tuple if there is already a tuple with the primary key
      * Insertion of a new tuple if the primary key contains null
    - Domain constraint
      * Insertion of a tuple with a value in a field that is not in the domain associated with that field
    - Foreign key constraint
      * Insertion of a tuple with a foreign key when the key is not in referenced table
  + Deletion does not cause a violation of domain, primary key or unique constraints
  + Update violation
    - Primary key constraint
      * Update a tuple with a new key when there is already a tuple with the key
  + Special cases
    - Deletions of Enrolled tuples do not violate referential integrity, but insertions of Enrolled tuples could
    - Insertions of Students tuples do not violate referential integrity, and deletions of Students tuples could cause violations
    - Updates on either Enrolled or Students that change the studid (respectively, sid) value could potentially violate referential integrity
  + Ways to handle foreign key violation by cases
    - Insertion of a tuple in Enrolled with a studid that is not in Students
      * INSERT command is simply rejected
    - Deletion of a row of Students
      * Delete all Enrolled rows that refer to the deleted Students row
      * Disallow the deletion of the Students row if an Enrolled row refers to it
      * Set the studid column to the sid of some (existing) ‘default’ student, for every Enrolled row that refers to the deleted Students row
      * For every Enrolled row that refers to it, set the studid column to null
        + But if this option conflicts with the fact that studid is part of the primary key of Enrolled and therefore cannot be set to null
  + SQL to handle foreign key violations DELETE and UPDATE
    - 
      * NO ACTION
        + Default option
        + The action (DELETE or UPDATE) is to be rejected
      * CASCADE
        + DELETE

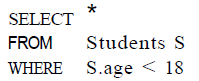
If a Students row is deleted, all Enrolled rows that refer to it are to be deleted as well

* + - * + Update

If the sid column of a Students row is updated, the update is carried out in each Enrolled row that refers to the updated Students row

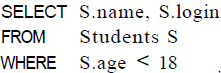
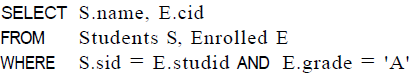
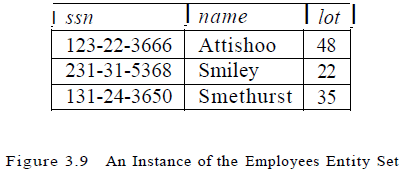
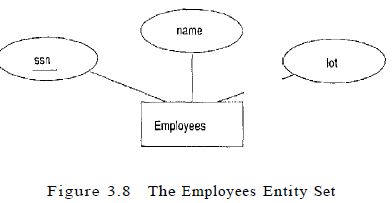
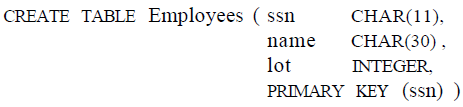
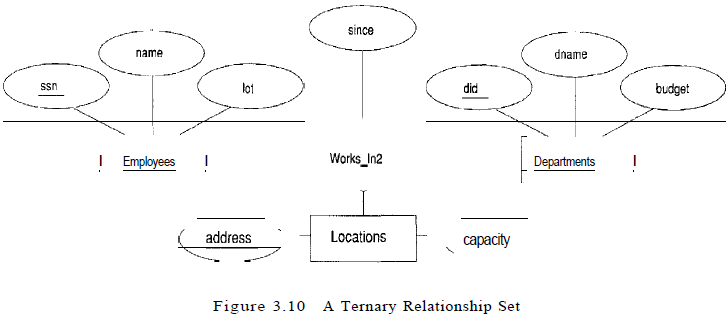
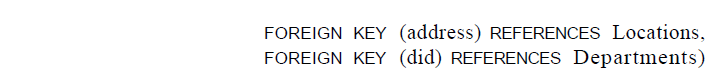
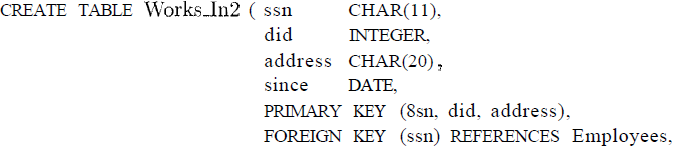
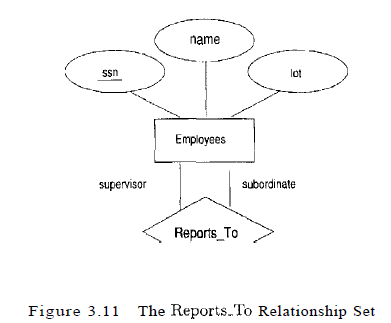
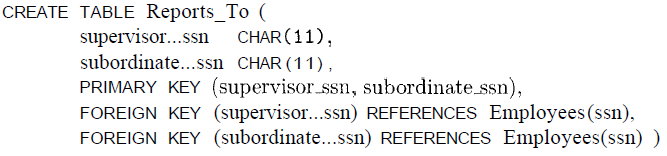
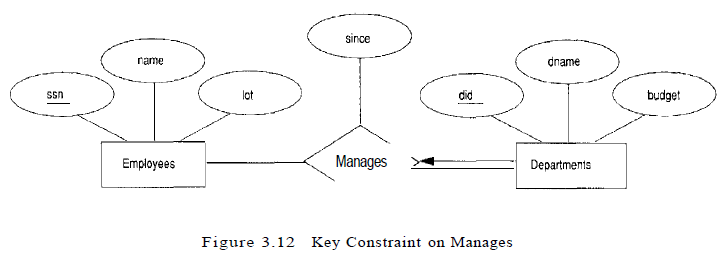
* + - * ON DELETE SET DEFAULT
        + If a Students row is deleted, we can switch the enrollment to a ‘default’ student
        + Default student is specified as part of the definition of the sid field in Enrolled

Example: sid CHAR(20) DEFAULT ‘53666’

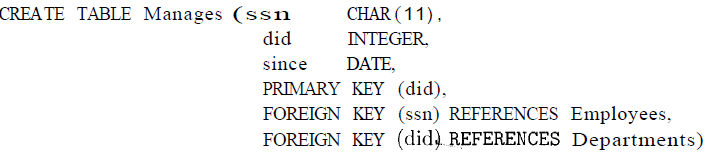
* + - * ON DELETE SET NULL
        + Use null as the default value
* Transactions and constraints
  + By default, a constraint is checked at the end of every SQL statement that could lead to a violation
    - If there is a violation, the statement is rejected
  + Modes in SQL
    - SET CONSTRAINT
      * Set constraint checking modes for the current transaction
    - DEFERRED
      * SET CONSTRAINT ConstraintFoo DEFFERRED
      * Constraint is checked at commit time
      * Example
        + Insert a boat with a non-existent sailor as the captain (Temporarily inconsistent)
        + Insert the sailor (Restore consistency)
        + Then commit and check that both constraints are satisfied
    - IMMEDIATE
      * Constraints are checked at the end of each SQL statement
* Querying relational data
  + Definition
    - Relational database query (query)
      * A question about the data, and the answer consists of a new relation containing the result
    - Query language
      * A specialized language for writing queries
  + SQL statements
    - 
      * \*
        + Retain all fields of selected tuples in the result
      * S
        + A variable that takes on the value of each tuple in Students, one tuple after the other
      * Where
        + Condition that specifies that what tuples we want to select
        + Invalid conditions

Comparison of an integer value to string value

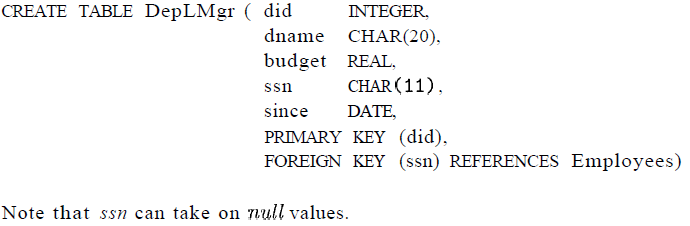
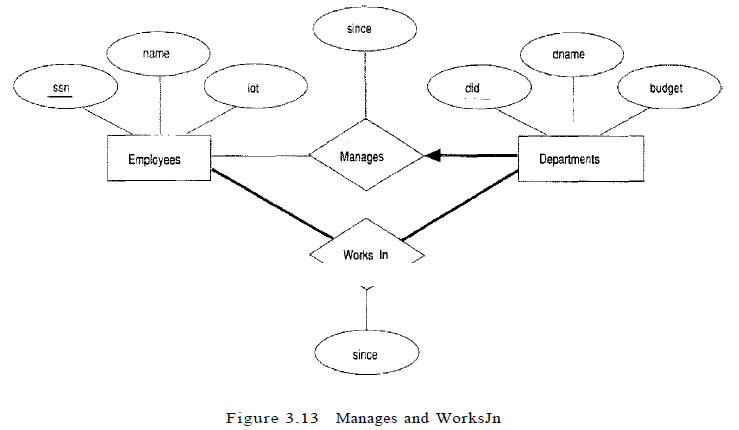
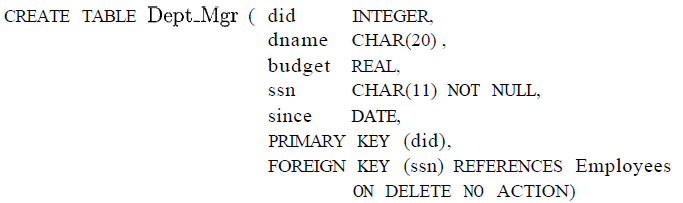
A query containing this condition produce no answer tuples

* + - 
      * A query can extract a subset of the fields of each selected tuple
    - 
      * If there a Studetns tuple S and an Enrolled tuple E such that S.sid = E.studid and E.grade = ‘A’, then print the student’s name and the course id
* Logical database design ER to Relational
  + Entity sets to Tables
    - Each attribute of the entity set becomes an attribute of the table
    - 
    - 
  + Relationship sets (without constraints) to tables
    - Must identify each participating entity and give values to the descriptive attributes of the relationship
    - Attributes of the relation include
      * Primary key attributes of each participating entity set, as foreign key fields
      * The descriptive attributes of the relationship set
    - Set of non-descriptive attributes is a superkey for the relation
    - If there are no key constraints, this set of attributes is a candidate key
    - 
    - 
      * NOT NULL constraint is implicit for each of address, did, and ssn fields
    - 
      * Role indicators supervisor and subordinate are used to create meaningful field names in the CREATE statement
      * 
        + We need to explicitly name the referenced field of Employees because the field name differs from the name of referring fields
  + Translating relationship sets with key constraints
    - 
      * First approach
        + Manages has the attributes ssn, did, and since
        + Helps to avoid inefficiency caused by space waste
        + Can be a slow operation as some important queries require us to combine information from two relations
        + Key constraint: Each department has at most one manager

Did is itself a key for Manages, set did, ssn is not a key

* + - * + 
      * Second approach
        + Include the information about the relationship set in the table corresponding to the entity set with the key, taking advantage of the key constraint
        + Helps to eliminate the need for separate Manages relation, and queries asking for a department’s manager can be answered without combining information from two relations
        + Drawback is that space could be wasted if several departments have no managers

The added fields would have to be filled with null values

* + - * + Because a department has at most one manager, we can add the key fields of the Employees tuple denoting the Manager and the since attribute to the Departments tuple
        + 
  + Translating relationship sets with participation constraints
    - 
      * By participation constraint, every department is required to have a manager
      * By key constraint, every department is required to have at most one manager
      * By second approach
        + 

Participation constraint

Because ssn cannot take on null values, each tuple of Dept\_Mgr identifies a tuple in Employees

Key constraint

By setting did as primary key, one department has at most one manager

* + - * By first approach
        + The constraint that every department must have a manager cannot be captured using the first translation approach
        + If we add NOT NULL constraints to the ssn and did fields

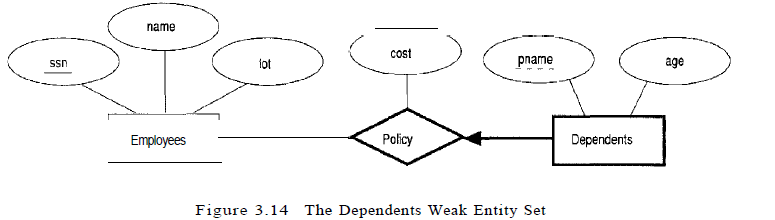
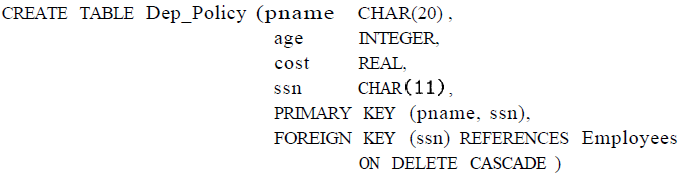
The constraint would prevent the firing of a manager

The constraint does not ensure that a manager is initially appointed for each department

* + - Need for assertion
      * To ensure total participation of Departments in Works\_In using SQL
        + Have to guarantee that every did value in Departments appears in a tuple of Works\_In
        + Tuple of Works\_In must have non-null values in the fields that are foreign keys referencing other entity sets involved in the relationship
        + Solution

We can ensure the second part of this constraint by imposing the stronger requirement that ssn in Works\_In cannot contain null values

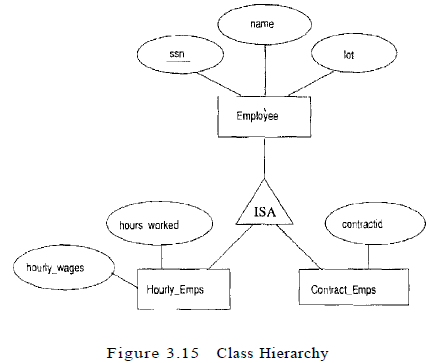
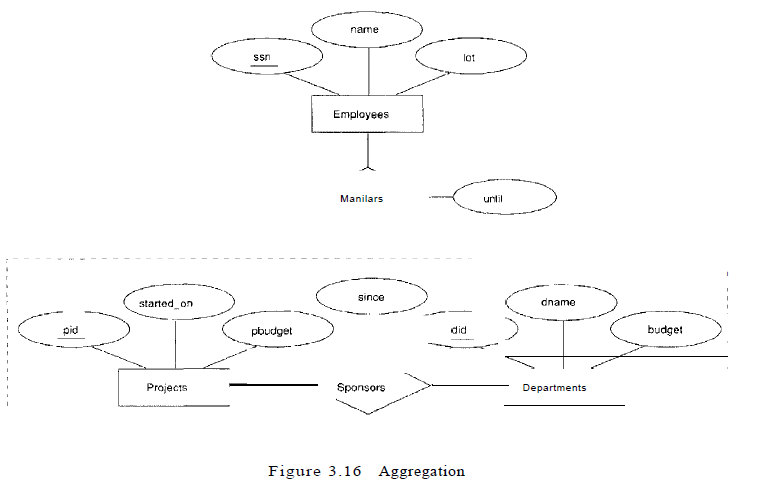
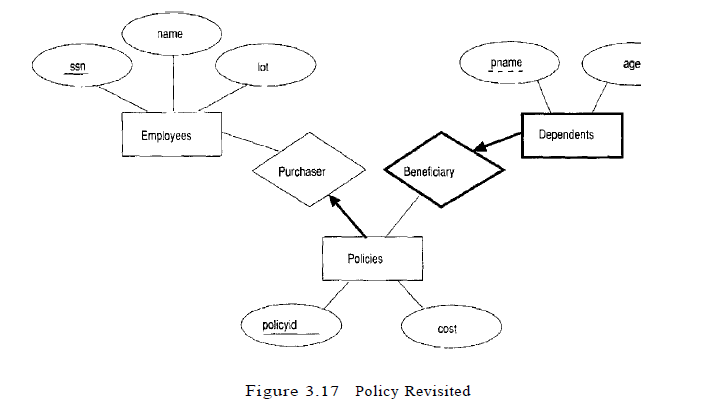
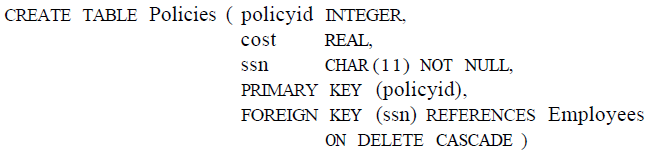
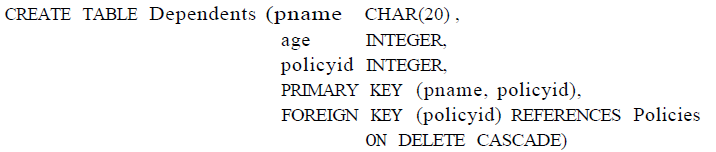
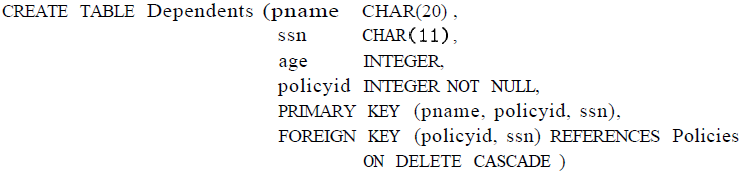
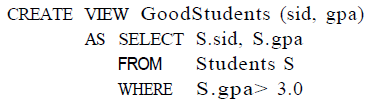
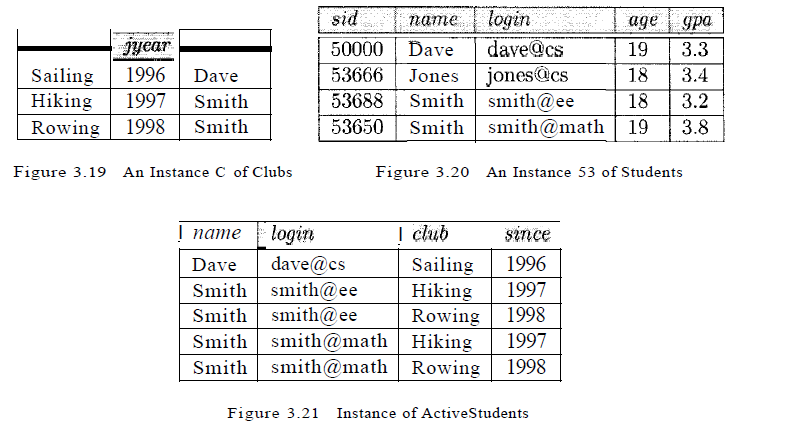
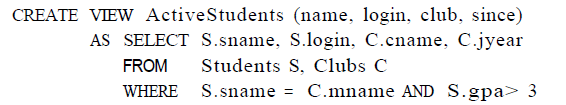
Ensure that participation of Employees in Works\_In is total is symmetric

* + - * To meet the requirement that each Employees entity must manage at least one department
        + Capture the constraint by using key and foreign key constraints
      * To capture a relationship set in which all participating entity sets have key constraints and total participation
        + Map all the entities as well as the relationship into a single table
        + Details are straightforward
  + Translating weak entity sets
    - A weak entity set always participates in a one to many binary relationship and has a key constraint and total participation
    - Use second approach
      * Must take into account that the weak entity has only a partial key
      * Must consider that when an owner entity is deleted, all owned weak entities should be deleted
    - 
      * Conditions
        + A Dependents entity can be identified uniquely only if we take the key of owning Employees entity and the pname of the Dependents entity
        + Dependents entity must be deleted if the owning Employees entity is deleted
      * 
        + Since Dependents is a weak entity, the primary key is (pname, ssn)

To ensure ssn cannot be null, ssn is part of primary key

* + - * + Use CASCADE for DELETE

Ensures that information about an employee’s policy and dependents is deleted if the corresponding Employees tuple is deleted

* + Translating class hierarchies
    - 
    - First approach
      * Map each of the entity sets Employees, Hourly\_Emps, and Contract\_Emps to a distinct relation
      * Make sure if the superclass tuple is deleted, the delete must be cascaded to Hourly\_Emps
      * Good
        + Queries in which we want to examine all employees and do not care about the attributes specific to the subclasses are handled easily using the Employees relation
      * Bad’
        + Queries in which we want to examinee specific employees may require us to combine Hourly\_Emps or Contract\_Emps with Employees to retrieve name and lot
    - Second approach
      * Create just two relations, corresponding to Hourly\_Emps and Contract\_Emps
      * The relation for Hourly\_Emps includes all the attributes of Hourly\_Emps as well as the attributes of Employees
      * Good
        + Query that needs to examine only hourly employees can do so by examining just one relation
      * Bad
        + Not applicable if we have employees who are neither hourly employees nor contract employees
        + If an employee is both an Hourly\_Emps and a Contract\_Emps entity, then the name and lot values are stored twice
        + Query that needs to examine all employees must examine two relations
  + Translating ER diagrams with Aggregation
    - 
      * Monitors relationship set
        + Key attributes of employees (ssn)
        + Key attributes of Sponsors (did, pid)
        + Descriptive attributes of Monitors (until)
      * Projects entity
        + Pid, started\_on, pbudget
      * Departments
        + Did, dname, budget
      * Sponsors
        + Key attributes of Projects (pid)
        + Key attribute of Departments (did)
        + Descriptive attributes of Sponsors (since)
      * Conditions to drop Sponsors relation
        + Sponsors has no descriptive attributes
        + Sponsors has total participation in Monitors
  + ER to Relational
    - 
      * Use key constraints to combine Purchaser information with Policies and Beneficiary information with Dependents
      * 
      * 
      * This model reflects the participation constraints in the ER diagram and intended actions when an employee entity is deleted
      * If policyid distinguishes only the policies owned by a given employee
        + Then policyid is only a partial key and Policies should be modeled as a weak entity set
        + 
* Introduction to views
  + Definition
    - View
      * A table whose rows are not explicitly stored in the database but are computed as needed from a view definition
      * View can be used just like a base table in defining new queries or views
  + VIEW SQL
    - CREATE VIEW B-Students (name, sid, course)  
       AS SELECT S.sname, S.sid, E.cid  
       FROM Students S, Enrolled E  
       WHERE S.sid = E.studid AND E.grade = ‘B’
      * View has B-Students has three fields (name, sid, course) with the same domains as the fields sname and sid in Students and cid in Enrolled
      * If optional arguments name, sid, and course are omitted from the CREATE View statement, the column names sname, sid, and cid are inherited
* View, Data Independence, Security
  + Definition
    - Physical schema
      * Describes how the relations in the conceptual schema are stored, in terms of the file organizations and indexes used
    - Conceptual schema
      * Collection of schemas of the relations stored in the database
  + View mechanism
    - Provides the support for logical data independence in the relational model
    - Can be used to define relations in the external schema that mask changes in the conceptual schema of the database from applications
  + Views are important in context of security
    - Define views that give a group of users access to just the information they are allowed to see
* Updates on views
  + Distinction between a view and a base table must be kept in mind
  + Update views
    - Views that are defined on a single base table using just selection and projection, with no use of aggregate operations
      * SQL-92 standard
        + Allows updates to be specified only on updatable views
  + 
    - Can implement a command to modify the gpa of a GoodStudents row by modifying the corresponding row in Students
    - Can delete a GoodStudents row by deleting the corresponding row from Students
    - If the view did not include a key for the underlying table
      * Several rows in the table could correspond to a single row in the view
      * A command that affects a row in the view then affects all corresponding rows in the underlying table
    - Can insert a GoodStudents row by inserting a row into Students, using null values in columns of Students that do not appear in GoodStudents
      * Primary key columns are not allowed to contain null values
        + If we attempt to insert rows through a view that does not contain the primary key of the underlying table, the insertions will be rejected
  + Important observation
    - INSERT or UPDATE may change the underlying base table so that the resulting row is not in the view
      * If we try to insert a row into the view, this row can be added to the underlying Students table, but it will not appear in the GoodStudents view because it does not satisfy the view condition
    - WITH CHECK OPTION
      * SQL default action is to allow insertion but this clause can disallow the insertion
      * Only rows that will actually appear in the view are permissible insertion
* Need to restrict view updates
  + 
  + Suppose that we are often interested in finding the names and logins of students with a gpa greater than 3 who belong to at least one club, along with the club name and the date they join the club
    - 
    - To delete the row (Smith, smith@ee, Hiking, 1997) from Active students
      * Delete the row from Students <53688, Smith, smmith@ee, 18, 3.2>
        + Has the effect of also deleting the row (Smith, smith@ee, Rowing, 1998) from the view ActiveStudents
      * Delete the row from Clubs <Hiking, 1997, Smith>
        + Has effect of deleting the row (Smith, smith@math, Hiking, 1997) from the view ActiveStudents
      * Reasonable solution is to disallow such updates on views
* Destroying/Altering tables and views
  + SQL
    - DROP TABLE
      * TO delete all the rows and remove the table definition information
      * DROP TABE Students RESTRICT
        + Destroys the Students table unless some view or integrity constraint refers to Students
        + Otherwise the command fails
      * DROP TABLE Students CASCADE
        + Students is dropped an any referencing views or integrity constraints are dropped as well
      * RESTRICT or CASCADE must be always specified
    - DROP VIEW
      * A view is dropped like DROP TABLE
    - ALTER TABLE
      * Modifies the structure of an existing table
      * Add a new column
        + 

Add a column maiden-name to Students

All existing rows are padded with null values in this column

* + - * Delete a column and add or drop integrity constraints on a table
        + Not discussed in this chapter
        + Dropping columns is treated very similarly to dropping tables or views