DATABASE SYSTEMS

CS – 355/ CE – 373

Instructor: Maria N. Samad

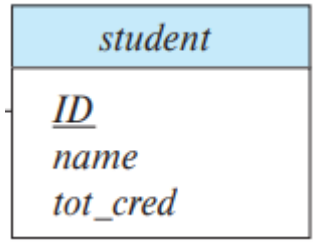
September 10th, 2024

E-R DIAGRAMS TO RELATION SCHEMAS

- Database represented by an E-R Diagram can be reduced to collection of relation schema
  - This is done to design the actual tables/relations in the database
- Each tuple in Relation Schema refers to each entity in the Entity Set
- E-R Diagram may consist of the following:
  - Strong Entity Sets
  - Weak Entity Sets
  - Relationship Sets

E-R DIAGRAMS TO RELATION SCHEMAS

- ***Strong Entity Sets:***
  - The primary key of the entity set becomes the primary key of the resultant schema
  - The rest of the attributes remain the same
  - For given entity set:



  - The resultant relation schema will simply be: *student(ID, name, tot_cred)*

E-R DIAGRAMS TO RELATION SCHEMAS

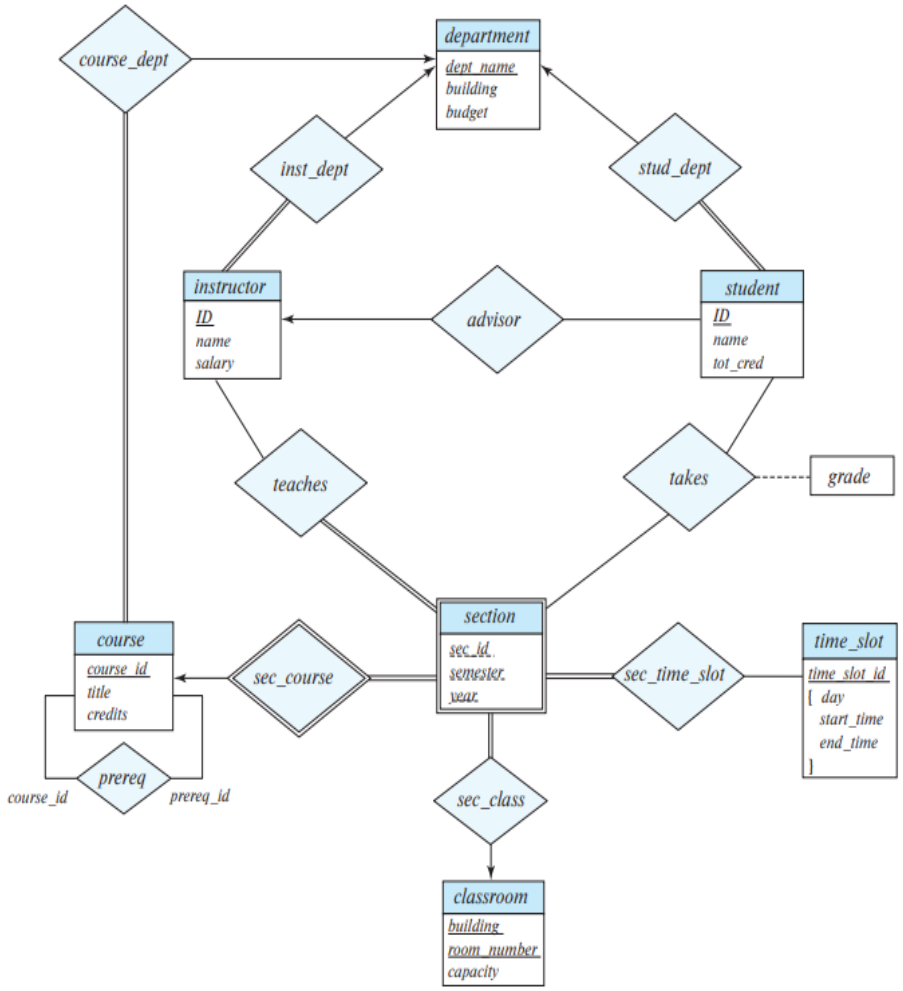


Figure 6.15 E-R diagram for a university enterprise.

Resultant Schemas?

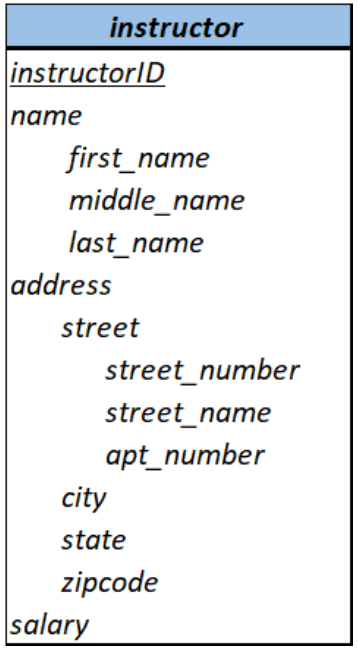
- Strong Entity Sets with simple Attributes
  - *department(dept_name, building, budget)*
  - *instructor(ID, name, salary)*
  - *student(ID, name, tot_cred)*
  - *course(course_id, title, credits)*
  - *classroom(building, room_number, capacity)*

E-R DIAGRAMS TO RELATION SCHEMAS

- ***Strong Entity Sets:***
  - What if Strong Entity Sets do not have simple attributes?
    - Composite Attributes
    - Multivalued Attributes
    - Derived Attributes

E-R DIAGRAMS TO RELATION SCHEMAS

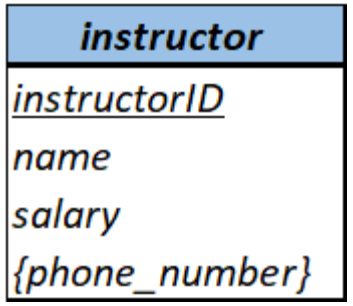
- ***Strong Entity Sets having Composite Attributes:***
  - A separate attribute is defined for each of the components and subcomponents of composite attributes
  - For example, let's say instructor entity set has the following simple and composite attributes:



  - Resultant Schema will be:
    - *instructor (instructorID, first_name, middle_name, last_name, street_number, street_name, apt_number, city, state, zipcode, salary)*

E-R DIAGRAMS TO RELATION SCHEMAS

- ***Strong Entity Sets having Multivalued Attributes:***
  - Convert the multivalued attribute to an atomic attribute and have **multiple** tuples to represent each value
  - Preferable approach is to define them as separate schema to save space and avoid redundancy and inconsistencies



- Resultant Schemas will be:
  - *instructor (instructorID, name, salary)*
  - *instructor_phone (instructorID, phone_number)*

E-R DIAGRAMS TO RELATION SCHEMAS

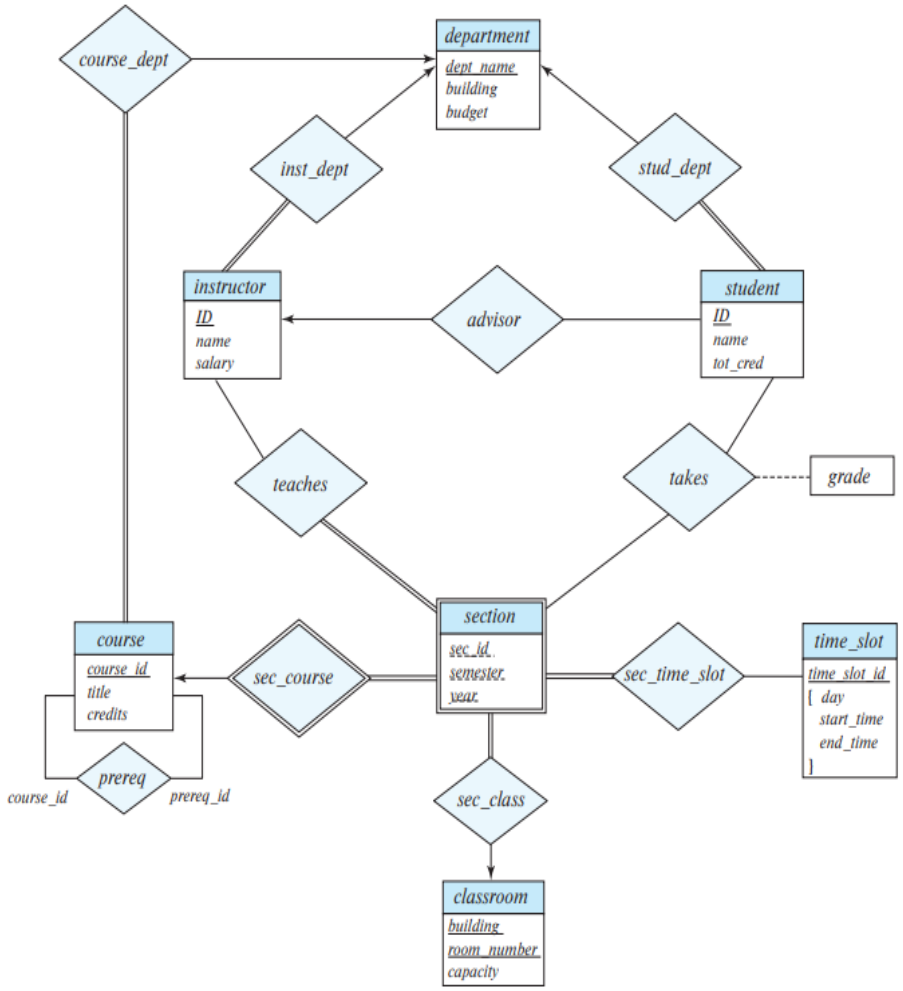


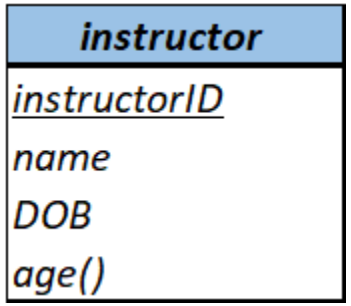
Figure 6.15 E-R diagram for a university enterprise.

Resultant Schemas?

- Strong Entity Sets with Multivalued Attributes:
  - *time_slot(time_slot_id)*
  - *time_slot_day(time_slot_id, day)*
  - *time_slot_stime(time_slot_id, start_time)*
  - *time_slot_etime(time_slot_id, end_time)*
- Here the original relation has no other meaningful attributes, hence remove that relation, and keep the rest, i.e.
  - *time_slot_day(time_slot_id, day)*
  - *time_slot_stime(time_slot_id, start_time)*
  - *time_slot_etime(time_slot_id, end_time)*

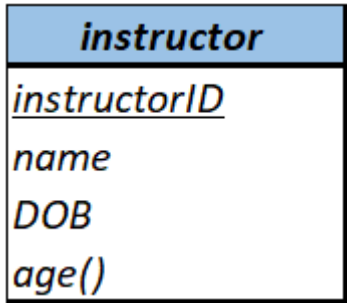
E-R DIAGRAMS TO RELATION SCHEMAS

- ***Strong Entity Sets having Derived Attributes:***
  - Not specifically represented in the relation schema
  - Will need to define them as transactions or as a part of functional requirements of any specific module in the design document
  - For example, the instructor entity set has a derived attribute, age( )



E-R DIAGRAMS TO RELATION SCHEMAS

- ***Strong Entity Sets having Derived Attributes:***
  - For example, the instructor entity set has a derived attribute, age( )



  - Resultant Schema will be:
    - *instructor (instructorID, name, DOB)*
  - No attribute for age( ), but can be defined as one of the functional requirements of the module, called **SeniorBenefits**, in which all the senior employees, i.e. employees above the age of 60 will get some monetary benefits along with their salary based on their age
  - In order to calculate the benefits, one of the functional requirements of **SeniorBenefits** module is calculating the age of the employee using his/her DOB attribute, and the current date

E-R DIAGRAMS TO RELATION SCHEMAS

- ***Weak Entity Sets:***
  - The primary key of the strong entity set and the discriminator key of weak entity set is combined to form a primary key of the resultant schema
  - The rest of the attributes remain the same
  - For example, Section is weak entity to Course strong entity
  - The resultant relation schema will simply be: *section(courseID, secID, semester, year)*

E-R DIAGRAMS TO RELATION SCHEMAS

- ***Relationship Sets:***
  - Every relation forms a new schema and the keys are selected as discussed before. They are based on:
    - Mapping Cardinalities
    - Descriptive Attributes
  - For example, advisor relation is many-to-one from student to instructor, so resultant schema will have keys of both student and instructor, but primary key will be that of student
  - The resultant relation schema will simply be: *advisor(studentID, instrID)*

E-R DIAGRAMS TO RELATION SCHEMAS

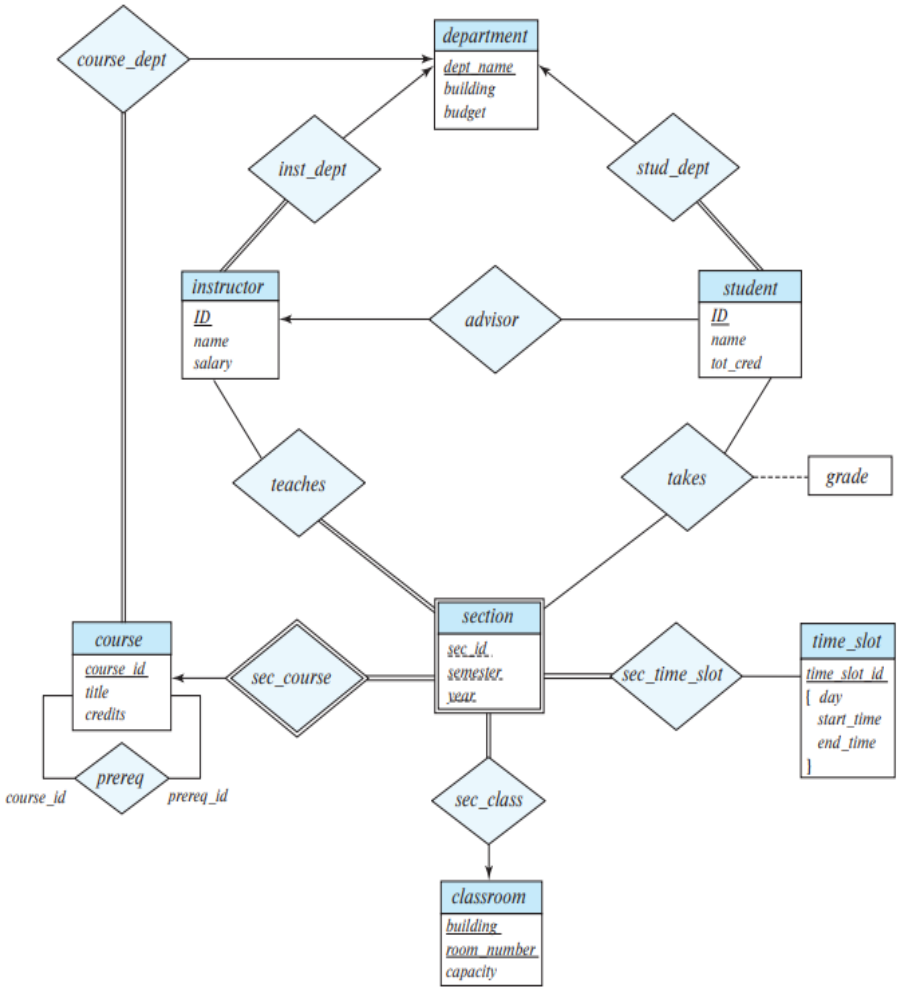


Figure 6.15 E-R diagram for a university enterprise.

Resultant Schemas?

- Relationship sets:
  - *inst_dept (instructorID, dept_name)*
  - *stud_dept (studentID, dept_name)*
  - *course_dept (courseID, dept_name)*
  - *teaches (instructorID, courseID, secID, semester, year)*
  - *takes(studentID, courseID, secID, semester, year, grade)*
  - *sec_course (courseID, secID, semester, year)*
  - *sec_time_slot (courseID, secID, semester, year, time_slot_id)*
  - *sec_class (courseID, secID, semester, year, building, room_number)*
  - *prereq (courseID, prereqID)*

E-R DIAGRAMS TO RELATION SCHEMAS

- Resultant Schemas:

Relations

- *department(dept_name, building, budget)*
- *instructor(ID, name, salary)*
- *student(ID, name, tot_cred)*
- *course(course_id, title, credits)*
- *classroom(building, room_number, capacity)*
- *time_slot_day(time_slot_id, day)*
- *time_slot_stime(time_slot_id, start_time)*
- *time_slot_etime(time_slot_id, end_time)*
- *section(courseID, secID, semester, year)*

Relationship Sets

- *sec_class (courseID, secID, semester, year, building, room_number)*
- *advisor(studentID, instructorID)*
- *inst_dept (instructorID, dept_name)*
- *stud_dept (studentID, dept_name)*
- *course_dept (courseID, dept_name)*
- *teaches (instructorID, courseID, secID, semester, year)*
- *takes(instructorID, courseID, secID, semester, year, grade)*
- *sec_time_slot (courseID, secID, semester, year, time_slot_id)*
- *prereq (courseID, prereqID)*
- *sec_course (courseID, secID, semester, year)*

E-R DIAGRAMS TO RELATION SCHEMAS

- Once all schemas have been deduced, redundant ones must be removed by keeping the entity ones
- For example:
  - *section(courseID, secID, semester, year)*
  - *sec_course(courseID, secID, semester, year)*

E-R DIAGRAMS TO RELATION SCHEMAS

- After removing redundancies and defining schemas for all entity sets and relationship sets, we get the following schemas:

- *department(dept_name, building, budget)*
- *instructor(instructorID, name, salary)*
- *student(studentID, name, tot_cred)*
- *course(courseID, title, credits)*
- *classroom(building, room_number, capacity)*
- *time_slot_day(time_slot_id, day)*
- *time_slot_stime(time_slot_id, start_time)*
- *time_slot_etime(time_slot_id, end_time)*
- *section(courseID, secID, semester, year)*
- *sec_class (courseID, secID, semester, year, building, room_number)*
- *advisor(studentID, instructorID)*
- *inst_dept (instructorID, dept_name)*
- *stud_dept (studentID, dept_name)*
- *course_dept (courseID, dept_name)*
- *teaches (instructorID, courseID, secID, semester, year)*
- *takes(instructorID, courseID, secID, semester, year, grade)*
- *sec_time_slot (courseID, secID, semester, year, time_slot_id)*
- *prereq (courseID, prereqID)*

E-R DIAGRAMS TO RELATION SCHEMAS

- Activity Sheet

E-R DIAGRAMS TO RELATION SCHEMAS

- Activity Sheet Solution:
  - ERD to Relational Schema Solution

DESIGN GUIDELINES FOR RELATION SCHEMAS

- The informal guidelines that are used as measures to determine the quality of the relation schema:
  1. Ensure that the semantics of the attributes are clear in the schema
  2. Reduce redundant information in tuples
  3. Reduce null values in tuples
  4. Disallow the possibility of generating spurious tuples

DESIGN GUIDELINES FOR RELATION SCHEMAS

1. Ensure that the semantics of the attributes are clear in the schema
   - Semantics of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple
   - All the attributes should have certain real-world meaning and relative interpretation
   - For example:
     - *Employee (SSN, EmpName, Bdate, Address, DeptNum)*
     - *Department (DeptNum, DeptName, DeptManagerSSN)*
     - *Project (ProjectID, ProjectName, ProjectLocation, DeptNum)*
     - *Department_Locations(DeptNum, DeptLocation)*
     - *Works_On (SSN, ProjectID, Hours)*

DESIGN GUIDELINES FOR RELATION SCHEMAS

1. Ensure that the semantics of the attributes are clear in the schema
   - All the relations in the previous slides are clearly defined with only relevant information, so clear semantics
   - However, for cases like:
     - *Emp_Dept (SSN, EmpName, Bdate, Address, DeptNum, DeptName, DeptManagerSSN)*
     - *Emp_Proj (SSN, ProjectID, Hours, EmpName, ProjectName, ProjectLocation)*
   - Are these tables referring to “Employee” information or “Department”/“Project” information?
   - Nothing wrong with it but unclear semantics and can cause further problems, as discussed in the following slides
   - Better to define each one separately for “Employee” containing employee details, “Department” containing department information, and “Project” containing project attributes

EXAMPLE

- Consider the following one table database as shown on the right

- Is this Instructor information?
- Department information?
- What is the primary key here?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

DESIGN GUIDELINES FOR RELATION SCHEMAS

2. Reduce redundant information in tuples
   - Having redundant information may lead to inconsistent data and/or data anomalies
   - There are three types of data anomalies:
     - ***Insertion Anomaly***
     - ***Deletion Anomaly***
     - ***Modification Anomaly***

DESIGN GUIDELINES FOR RELATION SCHEMAS

2. Reduce redundant information in tuples
   - ***Insertion Anomaly*:**
     - If the semantics are not clear for relations, adding new tuples to the relation may cause a data anomaly what is called a ***insertion anomaly***
     - For example, in the given relation:

       Emp_Dept (SSN, EmpName, Bdate, Address, DeptNum, DeptName, DeptManagerSSN)
     - If we are adding a new department, then we must enter at least one employee information, which means an employee must exist for a new department to be created, which is not necessary.
     - Hence, NULL values are added to the Employee information, but that makes the primary key ***SSN*** NULL as well, which will violate the basic definition of relation schemas
     - This problem is known as the insertion anomaly, in which data anomaly occurs while adding/inserting to the relation schema
     - Having separate schemas will not cause this issue, as adding a new department in the Department relation won't affect the employee details in the Employee relation

DESIGN GUIDELINES FOR RELATION SCHEMAS

2. Reduce redundant information in tuples
   - ***Deletion Anomaly*:**
     - If the semantics are not clear for relations, simple removal of a tuple may also create inconsistencies in relations causing a data anomaly what is called ***deletion anomaly***
     - For example, in the given relation:

       Emp_Dept (SSN, EmpName, Bdate, Address, DeptNum, DeptName, DeptManagerSSN)
     - If we are deleting a department, an employee related to the department will automatically be deleted.
     - Does this mean that employee cannot be transferred to another department or does he/she have to leave the company by default?
     - This problem is known as the deletion anomaly, in which data anomaly occurs while removing a tuple from the relation schema
     - In separate schemas, this will never cause any issues as a department can be independently deleted from the employee information
     - Then based on company's decision, the respective employees in the Employee table can either be updated for the department accordingly

DESIGN GUIDELINES FOR RELATION SCHEMAS

2. Reduce redundant information in tuples
   - ***Modification Anomaly***:
     - If the semantics are not clear for relations, modifying an attribute value in one tuple may lead to inconsistencies in other tuple, thus causing a data anomaly what is called ***modification anomaly***
     - For example, in the given relation:

       Emp_Dept (SSN, EmpName, Bdate, Address, DeptNum, DeptName, DeptManagerSSN)
     - If a new department manager is hired, then all tuples relevant to the older manager must be updated with the new manager's information
     - There can be hundreds of employees working under that manager, hence hundreds of tuples that need to be updated
     - If even one is missed out, the entire relation will become inconsistent
     - This problem is known as the modification anomaly, in which data anomaly occurs while modifying a tuple in the relation schema
     - In separate schemas, this will never cause any issues as there will only be one tuple for each department in the Department schema, hence changing it in one place only

EXAMPLE

- Observe the **redundancy** here in the attributes *dept_name*, *building* and *budget*.

- In particular, think about the methodology if we want to update the *building* of a department and/or the *budget* of a department?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

** Content from Dr. Faisal Alvi's slides **

EXAMPLE

- Now consider the situation when we want to add a new department to the university.
- How can a department be added when currently there are no employees in it?
- Will the table have **NULL** values?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

EXAMPLE

- Can we decompose the relation *in_dep* into two relations in order to minimize and/or eliminate these anomalies?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Figure 7.2 The *in_dep* relation.

DESIGN GUIDELINES FOR RELATION SCHEMAS

3. Reduce null values in tuples
   - By basic database design principle, we must avoid having NULL values in the relation, especially for Primary Keys
   - Not just a wastage of space, but also difficult to define what NULL will mean in the current context, as it can have multiple meanings:
     - The attribute is **not applicable** to the tuple
       - For example, in dependents information of an Employee, there can be a child who does not have a CNIC number, so the attribute does not apply in this situation
     - The attribute **value is unknown**
       - For example, having a spouse attribute of an employee – if the employee is not married yet, then this will be an unknown value, until the employee is married
     - The attribute **value is known but missing**
       - For example, an employee has a phone number, but he/she has not given it yet to be recorded in the relation

DESIGN GUIDELINES FOR RELATION SCHEMAS

3. Reduce null values in tuples
   - If an attribute contains many NULL values, then it may not be relevant to the relation schema, so it is preferable to remove it
   - If it is still unavoidable and not possible to remove, then it will be better to move it to a separate schema in order to avoid problems in the original schema for rest of the transactions

DESIGN GUIDELINES FOR RELATION SCHEMAS

4. Disallow the possibility of generating spurious tuples
   - Relations can be decomposed to get efficient design, as well as to avoid anomalies and redundancies
   - However, decomposing does not mean it loses its original information
   - That is, when combining them back, we should get all the tuples that were there before and no extra ones
   - For example, the given schema is:

     Emp_Proj (SSN, ProjectID, Hours, EmpName, ProjectName, ProjectLocation)
   - And it is decomposed in two tables as follows:
     - ***Emp_Locs (EmpName, ProjectLocation)***
     - ***Emp_Proj1 (SSN, ProjectID, Hours, ProjectName, ProjectLocation)***

DESIGN GUIDELINES FOR RELATION SCHEMAS

4. Disallow the possibility of generating spurious tuples
   - This results in the following tuples in each relation:

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

DESIGN GUIDELINES FOR RELATION SCHEMAS

4. Disallow the possibility of generating spurious tuples
   - This produces a particularly bad schema design because we cannot recover the information that was originally in ***Emp_Proj*** from ***Emp_Proj1*** and ***Emp_Locs***
   - For example, getting information on one employee with SSN = “123456789” after combining will result in spurious tuples (specified by * on the table)
   - This is a very bad result because how can one SSN belong to so many different people? This means that it was never decomposed correctly in the first place

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.

DATA ANOMALIES

- Activity Sheet

DATA ANOMALIES

- Activity Sheet Solution:
  - Data Anomalies Solution