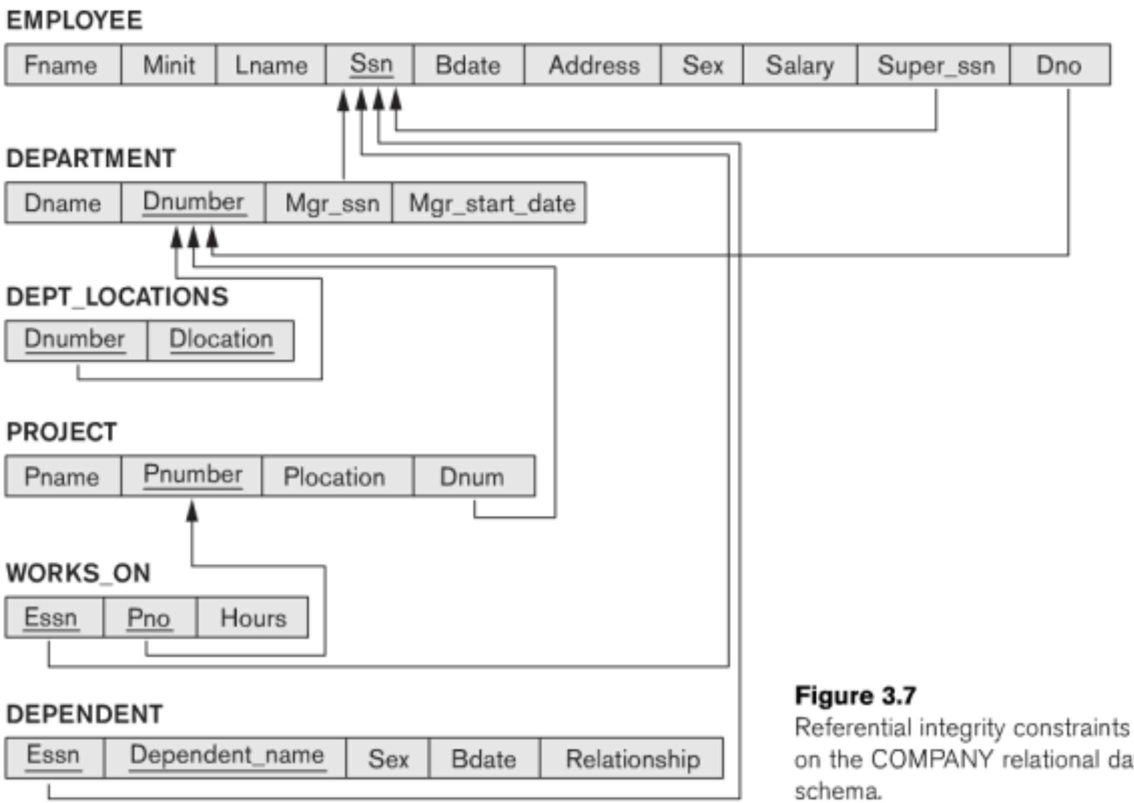
RELATION SCHEMA DESIGN

Corresponding Reading: Chapter 14.1-14.2

Relation Schemas

- Each relation schema consists of a number of attributes and the relational database schema consists of a number of relation schemas.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 3.7
Referential integrity constraints displayed on the COMPANY relational database schema.

Relation Schemas

- We've grouped attributes to form a relation schema by using common sense.
  - Employee information (name, bdate, etc.) are grouped into the EMPLOYEE relation
- We need a formal way of analyzing why one grouping of attributes into a relation schema may be better than another.
- We will discuss some DB theory that was developed with the goal of evaluating relational schemas for design quality.

The "Goodness" of Relation Schemas

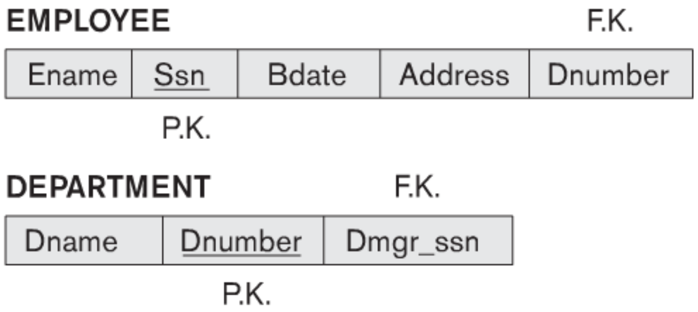
- Two levels where we can evaluate "goodness" of relation schemas:
  - Logical (conceptual) Level
    - How users interpret the relation schemas and the meaning of their attributes
    - A good schema enables user to clearly understand the meaning of data in relations and are able to formulate queries
  - Implementation (physical storage) Level
    - How are the tuples in a base relation are stored and updated
      - Base relations (not VIEWS) are physically stored as files.

Measuring Quality

- **Informal guidelines** for measuring the quality of relation schema design:
  - Making sure that the semantics of the attributes is clear in the schema
  - Reducing the redundant information in tuples
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples
- We *will discuss formal guidelines later* …

Semantics

- The semantics of a relation refers to its meaning resulting from the interpretation of the attribute values in a tuple.
- The easier it is to explain the semantics of the relation, the better the relation schema design will be.
- Example - EMPLOYEE schema:
  - Each tuple represents an employee with values for name, SSN, Bdate, Address and Department(Dnumber).
    - Dnumber is a **foreign key** that represents an **implicit** relationship



EMPLOYEE F.K.

Ename	Ssn	Bdate	Address	Dnumber

P.K.

DEPARTMENT F.K.

Dname	Dnumber	Dmgr_ssn

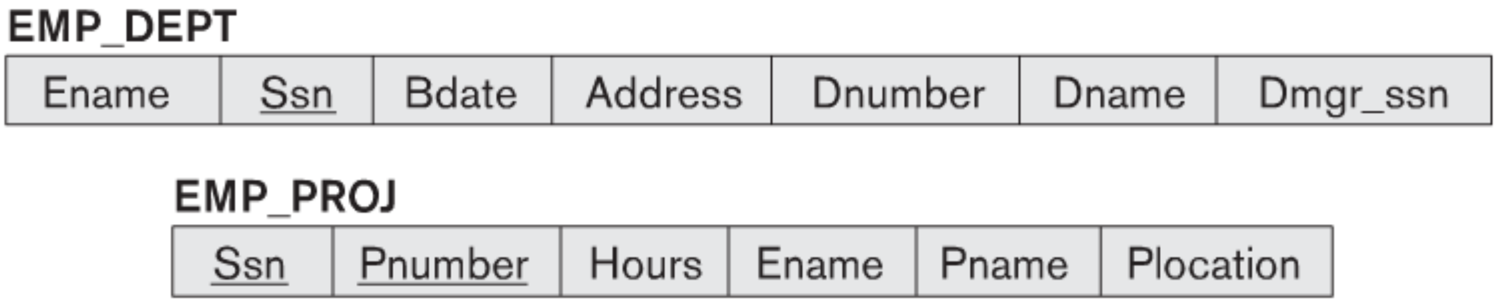
P.K.

Design Guideline #1

- Design a relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types and relationship types into a single relation.
- A relation schema should correspond to one entity type or one relationship type.
  - It will be straightforward to interpret and explain its meaning
  - If you violate this rule, then semantic ambiguities will result and the relation cannot be easily explained.

Example – Violating Guideline #1

- Two relations for storing Employee and Project data:



EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

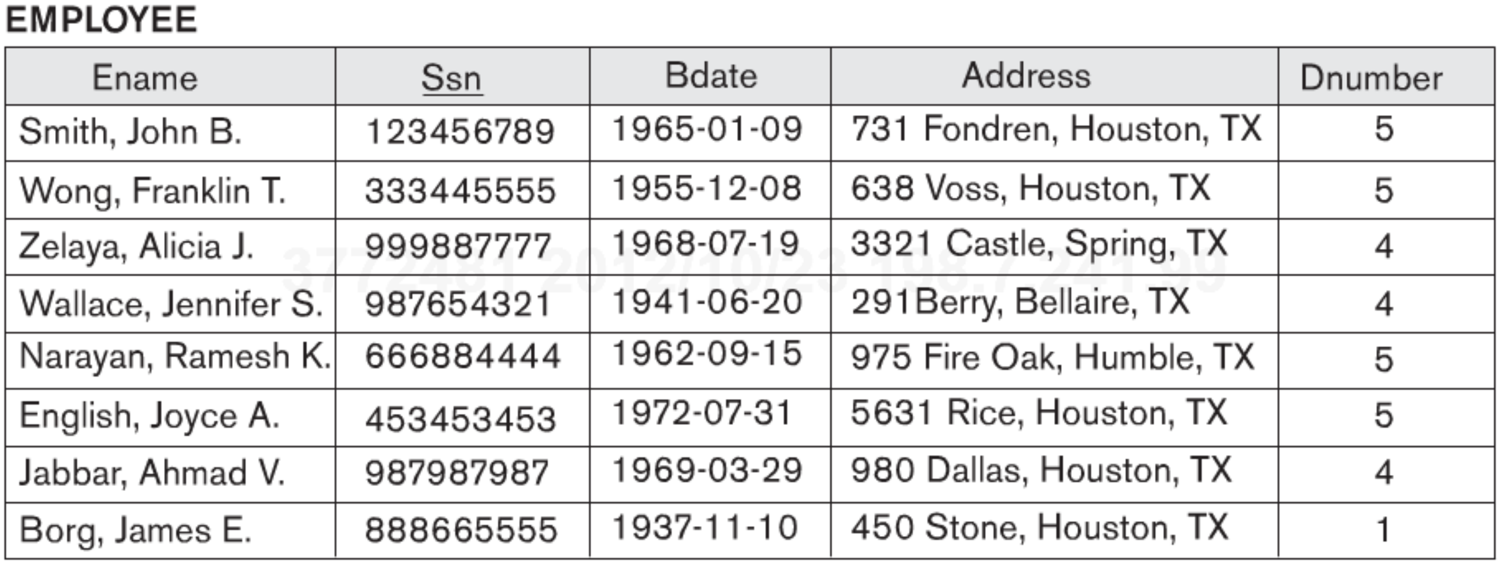
- Nothing logically wrong with the two relations, however they violate Guideline #1
  - They mix attributes from distinct real-world entities:
    - EMP_DEPT mixes attributes of employees and departments
    - EMP_PROJ mixes attributes of employees, projects, and the WORKS_ON relationship
  - Okay for VIEWS but not for BASE relations, we'll see why…

Redundant Information

- Goal of Schema design:
  - Minimize the storage space used by the base relations
  - Grouping attributes into relation schemas has a significant effect on storage space
- Let's compare
  - Our original relation schemas for EMPLOYEE and DEPARTMENT with
  - The new relation schemas EMP_DEPT and EMP_PROJ

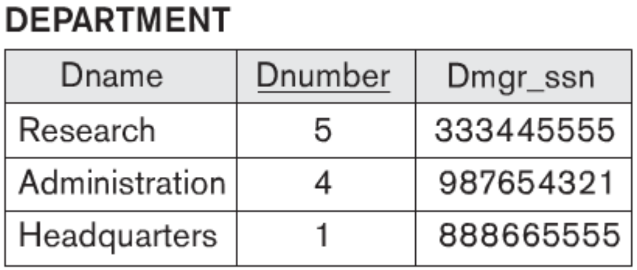
Redundant Information

- Using our original schemas:



EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

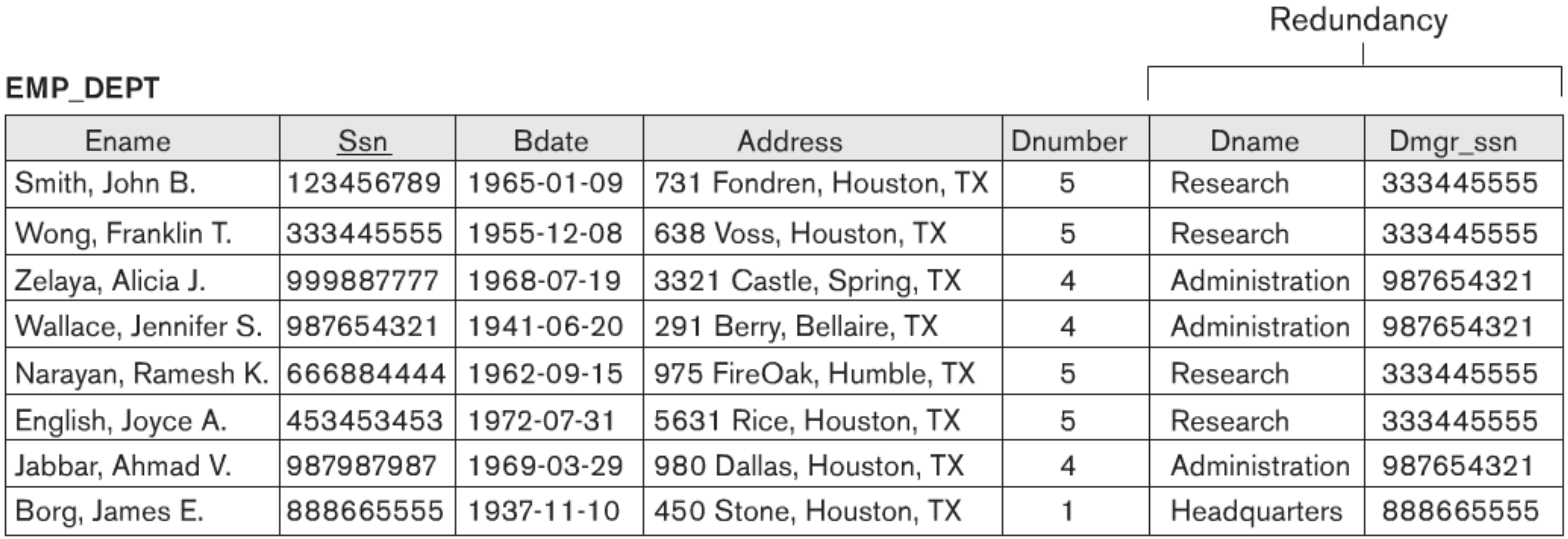


DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Redundant Information

- Using the new schemas:



Redundancy

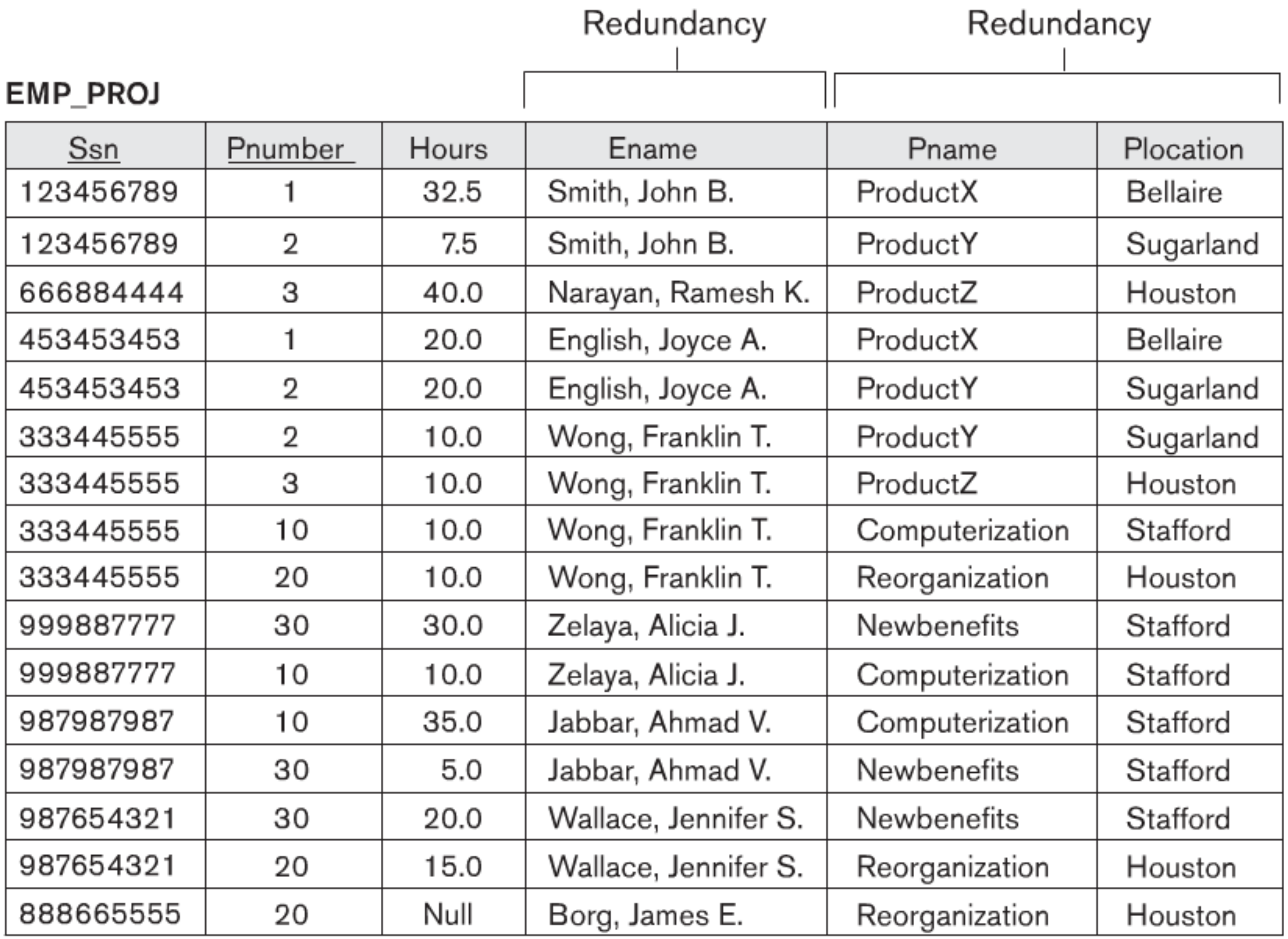
EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

  - Formed by a NATURAL JOIN on EMPLOYEE and DEPARTMENT
  - Repeated information for employees in same department
  - In the original DEPARTMENT relation, the data only appeared once.

Redundant Information

- Using the new schemas:


Redundancy
Redundancy

EMP_PROJ

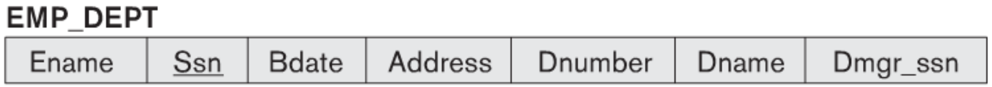
Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

Anomalies

- When we store data in relations that are formed by natural joins on base relations, we run into a series of problems called update anomalies.
- Classified into three types of anomalies:
  - Insertion Anomalies
  - Deletion Anomalies
  - Modification Anomalies

Insertion Anomalies

- Inserting a new employee into EMP_DEPT



EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

  - Must include either the attribute values for their department or NULLS (if employee is not assigned a dept. yet)
    - To associate a new employee to Dept. 5, we must enter all the attribute values for Dept. 5 correctly so that they are **consistent** with the corresponding values for Dept. 5 in other tuples.
  - Difficult to insert a new department that has no employees into the relation.
    - Would have to place NULL values in the attributes for employee
      - This VIOLATES the entity integrity [SSN is a primary key]

Deletion and Modification Anomalies

- What happens when we **DELETE** the last employee working for a department?
  - We have now lost all information regarding the department.
- What happens when we change the values of an attribute for a department (say the manager of Dept 5)?
  - We must update ALL employees that work in Dept 5
    - Otherwise the database will become inconsistent
  - If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples.

Design Guideline #2

- Design the base relation schema so that no insertion, deletion, or modification anomalies are present in the relations.
- If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.
  - Note: If you really need EMP_DEPT to exist (for user queries, etc.), create a stored relation (SQL VIEW) in addition to the base relations EMPLOYEE and DEPARTMENT.
    - The base relations can therefore remain anomaly free.

NULL Values

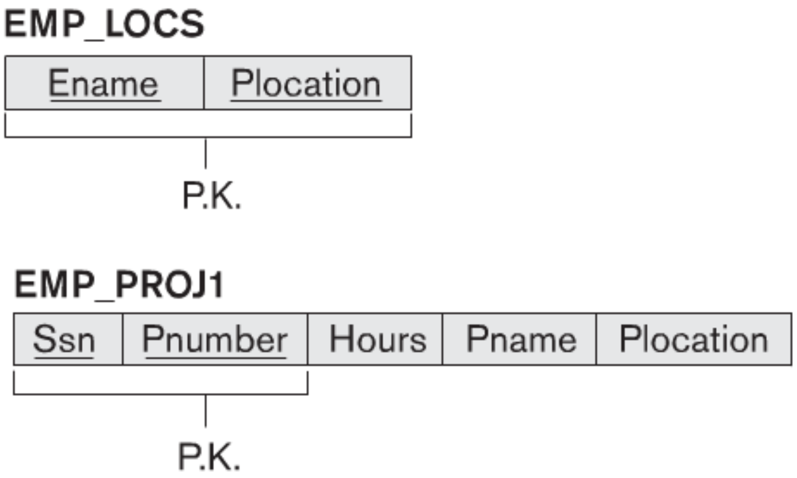
- If many attributes do not apply to all tuples in a relation, we can end up with many NULL values in those tuples.
  - Wastes space at the storage level
  - Makes JOIN and other operations difficult to understand
- What happens when we try to COUNT or SUM attributes?
- NULLs can also have multiple meaning:
  - The attribute does not apply to this tuple
  - The attribute value for this tuple is unknown
  - The value is know but is absent (not recorded yet)

Design Guideline #3

- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL.
- If NULL values are unavoidable:
  - Make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.
- Example:
  - If only 15 percent of employees have offices, there is no justification for including the Office_number attribute in EMPLOYEE.
    - Instead, create a new relation EMP_OFFICES(Essn, Office_number) to only include employees with office spaces.

Example – Decompose EMP_PROJ

- Consider two relation schemas (EMP_LOCS, EMP_PROJ1) that can be used instead of the EMP_PROJ relation.



EMP_LOCS

Ename	Plocation

P.K.

EMP_PROJ1

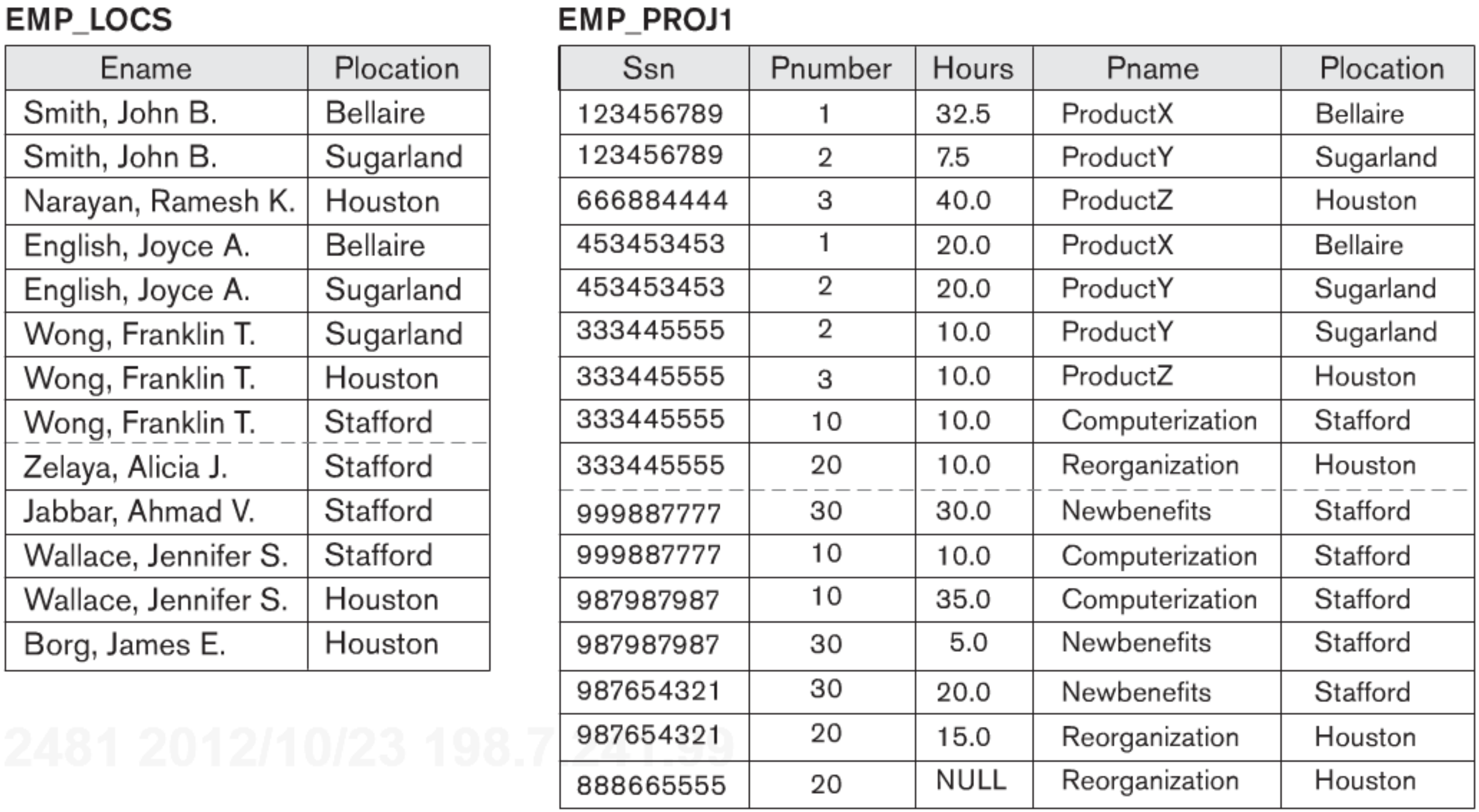
Ssn	Pnumber	Hours	Pname	Plocation

P.K.

  - A tuple in EMP_LOCS means that an employee works on some project at Plocation.
  - A tuple in EMP_PROJ1 means that an employee works X hours per work on a project at a specific location.

Example – New Relations

- Relation states of EMP_LOCS and EMP_PROJ1:



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

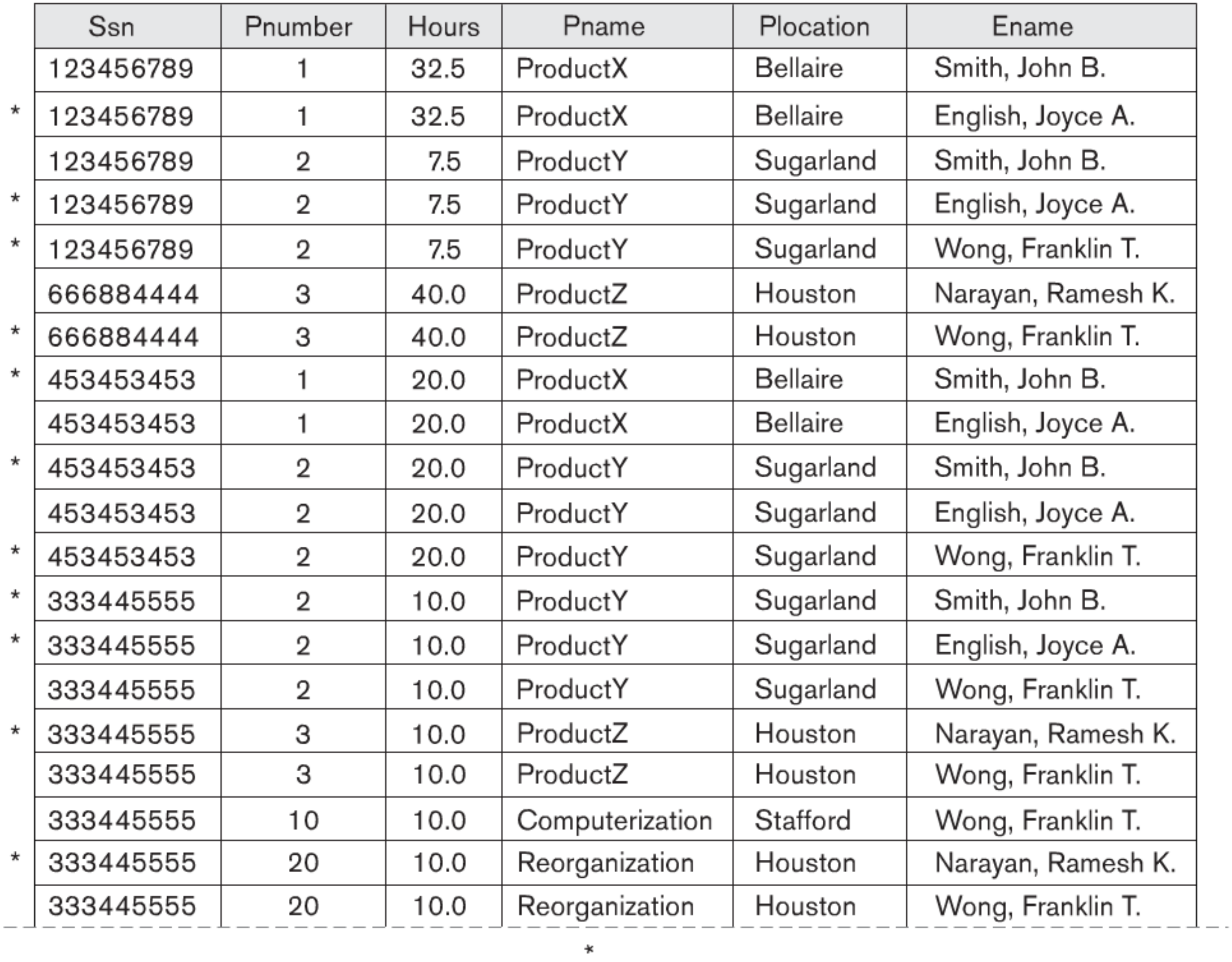
EMP_PROJ1

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

Example – How's this new design?

- What if we used these new relations instead of EMP_PROJ?
  - BAD SCHEMA DESIGN!
  - First: We cannot recover the original EMP_PROJ relation from the two new relations.
    - EMP_LOCS NATURAL JOIN EMP_PROJ1 results in many more tuples than the original set of tuples in EMP_PROJ

Example – Natural Join Result



	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.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

*

Spurious Tuples

- These additional tuples (*) that were not in the EMP_PROJ are called **spurious tuples.**
  - They represent spurious information that is not valid.
- Our choice to decompose EMP_PROJ into EMP_LOCS and EMP_PROJ1 is undesirable.
  - When we NATURAL JOIN the two relations, we do not get the correct original information.
    - Due to the fact that Plocation is the attribute that relates EMP_LOCS to EMP_PROJ1, and Plocation is neither a primary key nor a foreign key in either realtion.

Design Guideline #4

- Design relation schemas so that:
  - They can be Joined with equality conditions on attributes that are appropriately related (primary key, foreign key) in a way that guarantees that no spurious tuples are generated.
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

Summary of Informal Design Guidelines

- Watch out for these problems:
  - Anomalies that cause redundant work to be done
    - Insertion or modification of a relation
    - Deletions that may cause accidental loss of information
  - Waste of storage space due to NULLs and the difficulties of performing operations on NULL values
  - Generation of invalid and spurious data during joins on base relations
- How do we define these ideas formally?
  - We will see concepts/theory to define the "goodness" and "badness" of relations.

Formal Evaluation of Relations – 1st Method

- Functional Dependencies:
  - A functional dependency is a constraint between two sets of attributes from the database.
  - A functional dependency is a property of the **semantics** or meaning of the attributes
  - The main use of these dependencies is to specify additional constraints on the attributes of a relation that must hold at all times.
- Example: {State, Driver_license_number} --> Ssn
  - We could have a functional dependency that says for all licensed drivers in the USA, we can determine a SSN based on the state of the license and the license number.

Functional Dependency – Formal Definition

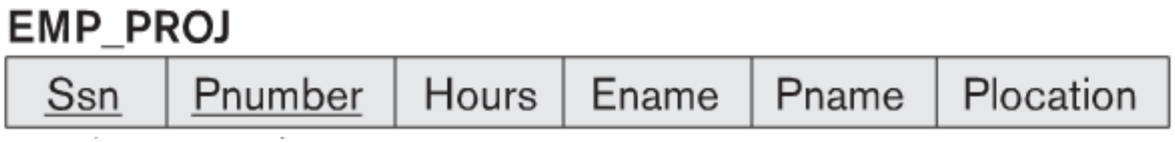
- Suppose that our relational database schema has n attributes: $A_1, A_2, \ldots, A_n$ and the entire database is described by a single, universal relation $R=\{A_1,A_2,\ldots,A_n\}$
  - A functional dependency, denoted by $X\text{-->}Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R.
  - The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X]= t_2[X]$, they must also have $t_1[Y]= t_2[Y]$.

Functional Dependency

- What does that really mean?
  - The values of the Y component of a tuple in r depend on, or are determined by, the values of the X component.
  - Alternatively, the values of the X component of a tuple uniquely (or **functionally**) determine the values of the Y component.
- We say that there is a functional dependency from X to Y or that Y is functionally dependent on X.

Functional Dependency Examples

- Consider EMP_PROJ relation



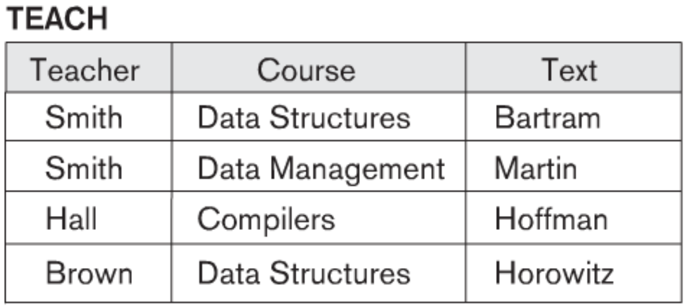
EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

- From the semantics of the attributes and relation, we identify:
  - Ssn --> Ename
    - The value of an employee's SSN uniquely determines the employee name
    - We can say: "Ename is functionally dependent on Ssn"
  - Pnumber --> {Pname, Plocation}
    - The value of a project number uniquely determines the project name and project location.
  - {Ssn, Pnumber} --> Hours
    - A combination of SSN and Pnumber values uniquely determines the number of hours an employee works on a project per week.

Creating Functional Dependencies

- Functional dependencies (FDs) are properties of a relation schema.
  - They must be defined by someone who knows the semantics of the attributes in the relation.
- Example:
  - It appears as though Text --> Course
    - We cannot confirm this unless we know that it is true for all possible legal states of TEACH.
    - However, it is sufficient to demonstrate a single counterexample to disprove a functional dependency.
      - Teacher does not functionally determine Course.



TEACH

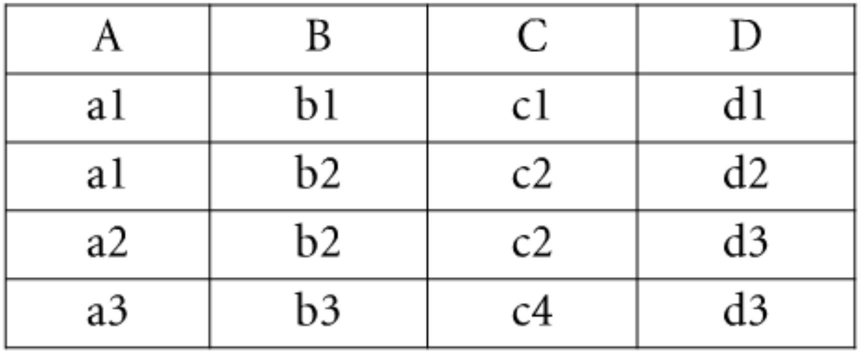
Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Creating Functional Dependencies

- Given a populated relation, one cannot determine which FDs hold and which do not unless the meaning of and the relationships among the attributes are known.
  - We can say is that a certain FD **may** hold.
  - We can also state that a certain FD does not hold if there are tuples that prove a violation of the FD.

Example – Functional Dependencies

- The following FDs may hold, because the four tuples in the current extension have no violation of these constraints.



A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

B --> C {A,B} --> C {A,B} --> D {C,D} --> B

- The following FDs do not hold because we already have violations of them in the given extension:

A --> B [Tuples 1 and 2 violate this constraint]

B --> A [Tuples 2 and 3 violate this constraint]

D --> C [Tuples 3 and 4 violate this constraint]