Lecture 1

Unit 3

Informal design guidelines for relation schemas

- Four informal measures of quality for relation schema design

  - Semantics of the attributes
  - Reducing the redundant values in tuples
  - Reducing the null values in tuples
  - Disallowing the possibility of generating spurious tuples

Semantics of the Relation Attributes

- **Semantics specifies how to interpret the attribute values stored in a tuple of the relation-i.e. how the attribute values in a tuple relate to one another.**

- GUIDELINE
  - Informally, each tuple in a relation should represent one entity or relationship instance.
  - Attributes of different entities should not be mixed in the same relation
  - Only foreign keys should be used to refer to other entities

EMPLOYEE

				f.k.
ENAME	SSN	BDATE	ADDRESS	DNUMBER
	p.k.			

DEPARTMENT

		f.k.
DNAME	DNUMBER	DMGRSSN
	p.k.	

DEPT_LOCATIONS

f.k.	
DNUMBER	DLOCATION
p.k.	

PROJECT

			f.k.
PNAME	PNUMBER	PLOCATION	DNUM
	p.k.		

WORKS_ON

f.k.	f.k.	
SSN	PNUMBER	HOURS
p.k.		

- ***Bottom Line**: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*

Redundant Information in Tuples and Update Anomalies

- **One goal of schema design is to minimize the storage space used by the base relations** (and hence the corresponding files).

- Another serious problem with using the relations as base relations is the problem of **update anomalies**

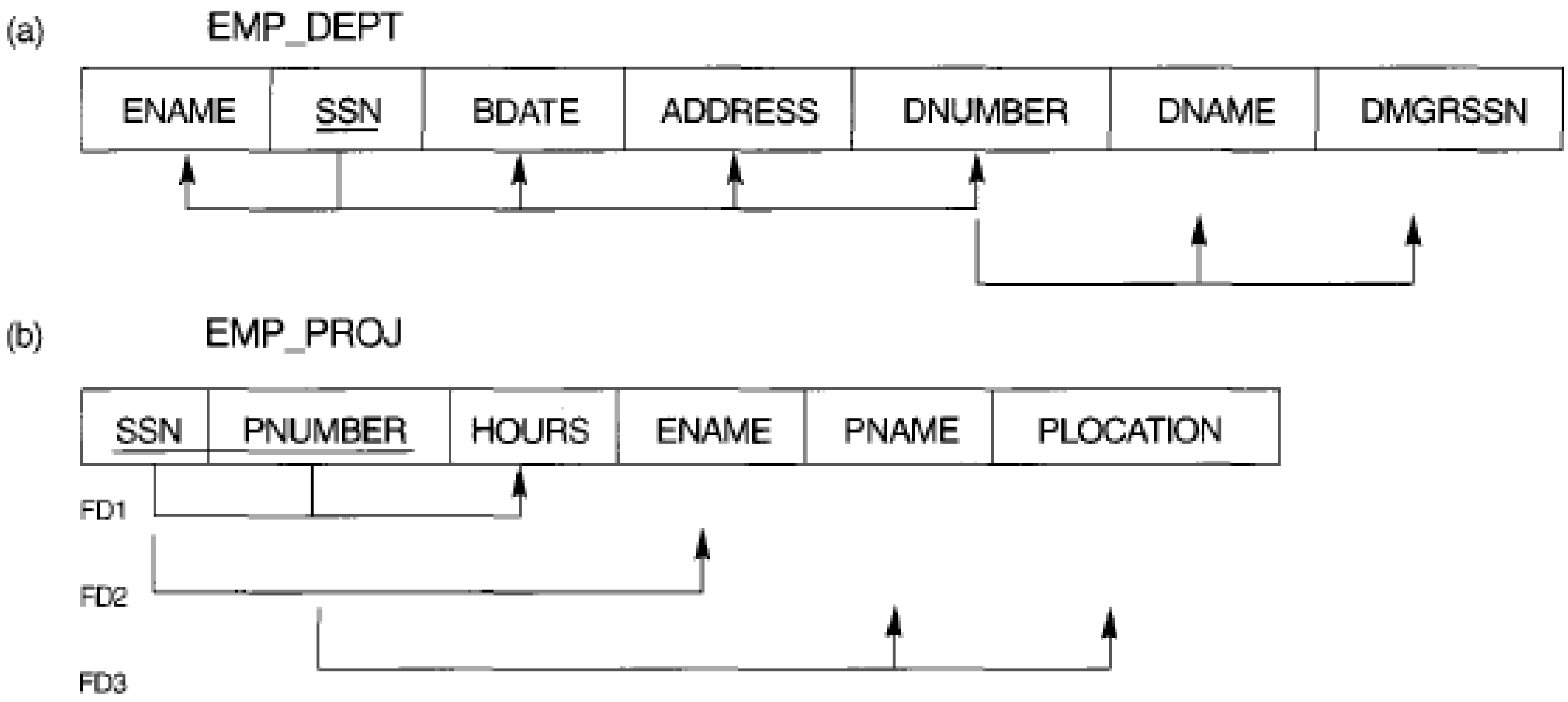


FIGURE 10.3 Two relation schemas suffering from update anomalies.

Insertion Anomalies

- It may not be possible to store some information unless some other information is stored as well.

EMP_DEPT

redundancy

ENAME	<u>SSN</u>	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

EMPLOYEE

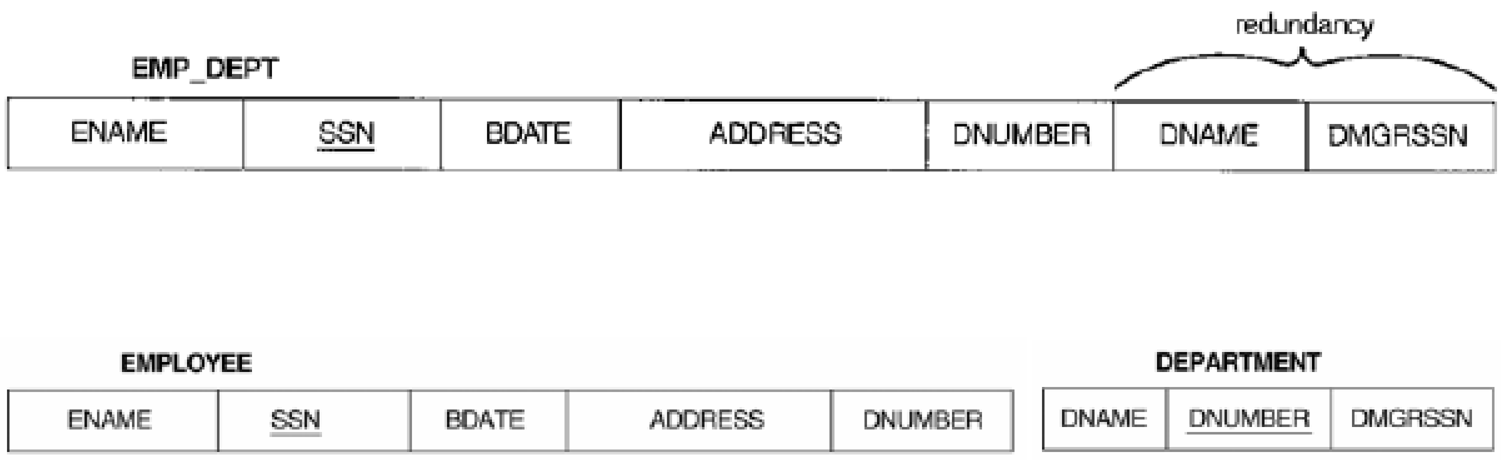
ENAME	<u>SSN</u>	BDATE	ADDRESS	DNUMBER

DEPARTMENT

DNAME	<u>DNUMBER</u>	DMGRSSN

Deletion Anomalies

- It may not be possible to delete some information without losing some other information as well



Modification Anomalies

- If one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly updated.

EMP_DEPT

redundancy

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
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

- GUIDELINE
  - Design a schema that does not suffer from update anomalies.
  - If there are any present, then note them so that applications can be made to take them into account

NULL VALUES IN TUPLES

- Problem with nulls is how to account for them when aggregate operations such as COUNT or SUM are applied.

- Reasons for nulls:
  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable

- GUIDELINE :
  - Relations should be designed such that their tuples will have as few NULL values as possible.
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)

Generation of Spurious Tuples

- Spurious tuples represent spurious or *wrong* information that is not valid

(a)

EMP_LOCS

ENAME	PLOCATION

p.k.

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION

p.k.

(b)

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	Product X	Bellaire
123456789	2	7.5	Product Y	Sugarland
666884444	3	40.0	Product Z	Houston
453453453	1	20.0	Product X	Bellaire
453453453	2	20.0	Product Y	Sugarland
333445555	2	10.0	Product Y	Sugarland
333445555	3	10.0	Product Z	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

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.

⋮

FIGURE 10.6 Result of applying NATURAL JOIN to the tuples above the dotted lines in EMP_PROJ1 EMP_LOCS of Figure 10.5. Generated spurious tuples are marked by asterisks.

- GUIDELINE:
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.
  - Bad designs for a relational database may result in erroneous results for certain JOIN operations

The Evils of Redundancy

- Redundant storage
- Update anomalies

Functional Dependency

- **A functional dependency is a constraint between two sets of attributes from the database.**

- *FD*s are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes

FUNCTIONAL DEPENDENCY (CONT..)

- Suppose that our relational database schema has n attributes A1, A2,…, An; let us think of the whole database as being described by a single universal relation schema R = {A1, A2,..An}

- A **functional dependency, denoted by X → *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 *t1* and *t2* in r that have t1[X] = t2[X], they must also have t1[Y] = t2[y]**
- This means that 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.

FUNCTIONAL DEPENDENCY (CONT..)

- We also say that there is a functional dependency from X to *Y*, or that *Y* is functionally dependent on X.

- ***FD* is a property of the attributes** in the schema
  - The constraint must hold on *every relation instance*
  - If *K* is a key of *R*, then *K* functionally determines all attributes in *R*

FUNCTIONAL DEPENDENCY (CONT..)

- **If X → *Y* in R, this does not say whether or not *Y* → X in R.**

- Certain FDs can be specified without referring to a specific relation, but as a property of those attributes.

FUNCTIONAL DEPENDENCY (CONT..)

{STATE, DRIVER_LICENSE_NO} → SSN

should hold for any adult in the United States

Functional Dependency (cont..)

- It is also possible that certain functional dependencies may cease to exist in the real world if the relationship changes.

FD ZIP_CODE → AREA_CODE

used to exist as a relationship between postal codes and telephone number codes in the United States, but with the proliferation of telephone area codes it is no longer true.

FUNCTIONAL DEPENDENCY (CONT..)

- Consider the relation schema EMP_PROJ, from the semantics of the attributes, we know that the following functional dependencies should hold:

  SSN → ENAME

  PNUMBER → {PNAME, PLOCATION}

  {SSN, PNUMBER} → HOURS

- We say that ENAME **is functionally determined** by (or functionally dependent on) SSN, or "given a value of SSN, we know the value of ENAME," and so on..

FUNCTIONAL DEPENDENCY (CONT..)

- A functional dependency is a *property of the relation schema* R, not of a particular legal relation state r of R.

- Hence, **an FD *cannot* be inferred automatically from a given relation extension r but must be defined explicitly by someone who knows the semantics of the attributes of R**

- At first glance we may think that TEXT → COURSE, we cannot confirm this unless we know that it is true *for all possible legal states* of TEACH

TEACH

TEACHER	COURSE	TEXT
Smith	Data Structures	Bartram
Smith	Data Management	Al-Nour
Hall	Compilers	Hoffman
Brown	Data Structures	Augenthaler

Inference Rules for Functional Dependencies

- We denote F to be the set of functional dependencies that are specified on relation schema R.
- The schema designer specifies the functional dependencies that are semantically *obvious*
- Other dependencies can be *inferred* or *deduced* from the FDs in F.

Inference Rules for Functional Dependencies (cont..)

- For example,
  - If each department has one manager, so that DEPT_NO uniquely determines MGR_SSN

    DEPT_NO → MGR_SSN

  - A Manager has a unique phone number called MGR_PHONE

    MGR_SSN → MGR_PHONE

  - Then these two dependencies together imply that

    DEPT_NO → MGR_PHONE

Inference Rules for Functional Dependencies (cont..)

- **The set of all dependencies that include F as well as all dependencies that can be inferred from F is called the closure of F**

- **Closure of F is denoted by F+**

- For example, suppose that we specify the following set F of obvious functional dependencies on the relation schema

  F= {SSN → {ENAME, BDATE, ADDRESS, DNUMBER},
  DNUMBER → {DNAME, DMGRSSN}}

- Some of the additional functional dependencies that we can *infer* from F are the following:

  SSN → {DNAME, DMGRSSN}
  SSN → SSN
  DNUMBER → DNAME

Inference Rules for Functional Dependencies (cont..)

- **An FD X → *Y* is inferred from a set of dependencies F specified on R if X → *Y* holds in *every* legal relation state r of R**; that is, whenever r satisfies all the dependencies in F, X → *Y* also holds in r.

- Use the notation **F |= X → *Y* to denote that the functional dependency X → *Y* is inferred from the set of functional dependencies F**.

Inference Rules for Functional Dependencies (cont..)

- Inference rules for functional dependencies
  - IRI (reflexive rule}: If X $\supseteq$ *Y*, then X → Y.
  - IR2 (augmentation rule): {X → Y} |= XZ → *YZ*.
  - IR3 (transitive rule): {X → *Y*, *Y* → Z} |= X → Z.
  - IR4 (decomposition, or projective, rule):{X → *YZ*} |=X → Y.
  - IRS (union, or additive, rule): {X → Y, X → Z} |=X → YZ
  - IR6 (pseudotransitive rule): {X → Y, WY → Z} |=WX → Z

Inference Rules for Functional Dependencies (cont..)

- Reflexive rule (IR1)
  - **States that a set of attributes always determines itself or any of its subsets**
  - It generates dependencies that are always true, such dependencies are called *trivial*
  - Formally, a functional dependency **X → Y is *trivial* if X ⊇ Y**; otherwise, it is nontrivial

Inference Rules for Functional Dependencies (cont..)

- Augmentation rule (IR2)
  - **Adding the same set of attributes to both the left- and right-hand sides of a dependency results in another valid dependency**

- Transitive rule (IR3)
  - Functional dependencies are transitive

Inference Rules for Functional Dependencies (cont..)

- Decomposition rule (IR4)
  - We can remove attributes from the right-hand side of a dependency
  - Applying this rule repeatedly can decompose the FD $X \rightarrow \{A1, A2, \ldots, An\}$ into the set of dependencies $\{X \rightarrow A1, X \rightarrow A2, \ldots, X \rightarrow An\}$

Inference Rules for Functional Dependencies (cont..)

- Union rule (IR5)
  - Allows us to do the opposite
  - We can combine a set of dependencies {X→ *A1*, X → A2, …. ,X → *An*} into the single FD X→*{A1*, A2, …. ,*An}*

- Caution
  - Although X → A and X → B implies X → AB by the union rule stated above, X →A, and Y → B does not imply that XY →AB.
  - Also, XY→A does not necessarily imply either X → A or Y → A

Inference Rules for Functional Dependencies (cont..)

- Proof of the Inference Rules – go thro by yourself
  - Proof of IR1
  - Proof of IR2 – by contradiction
  - Proof of IR3
  - Proof of IR4 – use IR1 through IR3
  - Proof of IR5 – use IR1 through IR3
  - Proof of IR6 – use IR1 through IR3

- Inference rules IR1through IR3 are sound and complete

Inference rules (cont..)

- **Sound** - Given a set of functional dependencies F specified on a relation schema R, any dependency that we can infer from F by using IR1 through IR3 holds in every relation state r of R that *satisfies the dependencies* in F, i.e. all dependencies are correct

- **Complete** - using IR1 through IR3 repeatedly to infer dependencies until no more dependencies can be inferred results in the complete set of *all possible dependencies* that can be inferred from F, i.e. all dependencies can be inferred from F only

Armstrong's rules

- The set of dependencies F+, which we called the closure of F, can be determined from F by using only inference rules IR1 through IR3.

- **Inference rules IR1 through IR3 are known as Armstrong's inference rules**