Module 7

Designing Relational Schemas

Objectives

- Learn what makes one relational schema a better design than another
- Learn what functional dependencies are
- Learn normalization techniques for decomposing relational schemas
- Reference:
  Elmasri & Navathe, Chapter 12

95.305

Topics

- Design Guide-lines for Relational Schemas
- Functional Dependencies
- Normal Forms
- 2nd and 3rd Normal Forms
- Boyce-Codd Normal Form

What we have looked at so far

- So far we have designed relational schemas by

  -common sense
  -mappings from E-R diagrams
- Is there a more formal theory of what makes one grouping of attributes a “good” relation

Simple Guide-lines for Good Relational Design

- based on:

  -meaning of attributes
  -minimizing redundancy
  -minimizing null values in tuples
  -not allowing spurious tuples

  These objective can conflict with one another

Attribute Meaning

- Attributes are generally grouped because together they convey some meaning
- For example the attributes may all convey information about an employee of a company
- If attributes are well organized with respect to their meaning, the relations are are easier to understand, and the design easier to do.

An example Company Schema

EMPLOYEE

				FK
ENAME	SSN	BDATE	ADDRESS	DNUMBER

DEPARTMENT

		FK	
DNAME	DNUMBER	DMGRSSN	DLOCATIONS

DEPT_LOCATIONS

DNUMBER	DLOCATION
FK	

PROJECT

			FK
PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

SSN	PNUMBER	HOURS
FK	FK	

Is this a good shema?

It is in the sense that each table seems easy understand

The groupings make sense: EMPLOYEE, DEPTARTMENT, PROJECT are distinct entities

DEPT_LOCATIONS, WORKS_ON are relationships

Design Guide-line

- **Guide-line #1**
- **Design schemas so that it's easy to explain its meaning.**
- **Don't combine attributes from different entities in the same relation**

What about these relations

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION

These schemas appear easy enough to understand

But, they violate guide-line #1

So why is this bad?

Redundancy

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

The DNAME, DMGRSSN will be repeated
in the table for each employee of the department

EMP_DEPT

FNAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith, John	123456789	9-Jan-55	731 Fondern	5	Research	333445555
Wong, Franklin	333445555	8-Dec-45	638 Voss	5	Research	333445555
Zelaya, Alicia	999887777	19-Jul-58	3321 Castle	4	Admin	987654321
Wallace, Jennifer	987654321	20-Jun-31	291 Berry	4	Admin	987654321
Narayan, Ramesh	666884444	15-Sep-52	975 Fire Oak	5	Research	333445555
English, Joyce	453453453	31-Jul-62	5631 Rice	5	Research	333445555
Jabber, Ahmad	987987987	29-Mar-59	980 Dallas	4	Admin	987654321
Borg, James	888665555	10-Nov-27	450 Stone	1	HeadQuarters	888665555

Update Anomalies

- Relations that violate Guide-line #1 will exhibit update anomalies
- Anomalies:

  -insertion anomalies
  -deletion anomalies
  -modification anomalies

Insertion Anomalies

EMP_DEPT

FNAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith, John	123456789	9-Jan-55	731 Fondern	5	Research	333445555
Wong, Franklin	333445555	8-Dec-45	638 Voss	5	Research	333445555
Zelaya, Alicia	999887777	19-Jul-58	3321 Castle	4	Admin	987654321
Wallace, Jennifer	987654321	20-Jun-31	291 Berry	4	Administration	987654321
Narayan, Ramesh	666884444	15-Sep-52	975 Fire Oak	5	Research	333445555
English, Joyce	453453453	31-Jul-62	5631 Rice	5	Res.	333445555
Jabber, Ahmad	987987987	29-Mar-59	980 Dallas	4	Admin	987654321
Borg, James	888665555	10-Nov-27	450 Stone	1	HeadQuarters	888665555

How do we insert a new employee who is not assigned to a department? (null-fill the dept. fields)

How do we enter a new dept. which does not have any employees? (cannot null fill a key attribute)

How do we know redundant information is consistently entered? (e.g. Admin vs. Administration)

Deletion Anomalies

EMP_DEPT

FNAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith, John	123456789	9-Jan-55	731 Fondern	5	Research	333445555
Wong, Franklin	333445555	8-Dec-45	638 Voss	5	Research	333445555
Zelaya, Alicia	999887777	19-Jul-58	3321 Castle	4	Admin	987654321
Wallace, Jennifer	987654321	20-Jun-31	291 Berry	4	Administration	987654321
Narayan, Ramesh	666884444	15-Sep-52	975 Fire Oak	5	Research	333445555
English, Joyce	453453453	31-Jul-62	5631 Rice	5	Res.	333445555
Jabber, Ahmad	987987987	29-Mar-59	980 Dallas	4	Admin	987654321
Borg, James	888665555	10-Nov-27	450 Stone	1	HeadQuarters	888665555

If we delete the last employee of a department, we must also delete the department

Modification Anomalies

EMP_DEPT

FNAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith, John	123456789	9-Jan-55	731 Fondern	5	Research	999887777
Wong, Franklin	333445555	8-Dec-45	638 Voss	5	Research	999887777
Zelaya, Alicia	999887777	19-Jul-58	3321 Castle	4	Admin	987654321
Wallace, Jennifer	987654321	20-Jun-31	291 Berry	4	Admin	987654321
Narayan, Ramesh	666884444	15-Sep-52	975 Fire Oak	5	Research	333445555
English, Joyce	453453453	31-Jul-62	5631 Rice	5	Research	333445555
Jabber, Ahmad	987987987	29-Mar-59	980 Dallas	4	Admin	987654321
Borg, James	888665555	10-Nov-27	450 Stone	1	HeadQuarters	888665555

If we change the manager of dept. #5, we must know to change all occurrences of this information, or else the database will be inconsistent (see table)

Design Guide-line

- **Design Guide-line #2**
- **Design the relational schemas so that no insertion, deletion, or modification anomalies occur in the relations**
- **If any anomalies are present, they must be noted clearly so that updating programs can operate correctly**

Reasons for breaking Guide-lines

EMPLOYEE

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

DEPARTMENT

		FK
DNAME	<u>DNUMBER</u>	DMGRSSN

EMP_DEPT

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

- **EMPLOYEE, DEPARTMENT are better for base relations**
- **EMP_DEPT may be suitable for a view**
- **EMP_DEPT may result in better performance and warrant keeping redundancies**

NULL Values

- “Fat” relations are subject to many NULL values
- e.g. if attributes apply only to some tuples
- NULL values cause problems in understanding, joins, and aggregate functions
- NULL values can be difficult to interpret
  - -attribute does not apply
  - -attribute value is unknown
  - -attribute value is absent

Design Guide-line

- Design Guide-line #3
- Try to avoid relation attributes whose values can be NULL
- If NULLs are unavoidable, try to ensure they only apply to exceptional cases
- e.g. if only 10% of employees have an office, it may be inappropriate to include an OFFICE_NUM attribute in the EMPLOYEE table
- Instead, create an EMP_OFFICE relation

Spurious Tuples

EMP_PROJ

<u>SSN PNUMBER</u> HOURS ENAME PNAME PLOCATION

EMP_LOC

<u>ENAME PLOCATION</u>

EMP_PROJ1

<u>SSN PNUMBER</u> HOURS PNAME PLOCATION

- **Consider decomposing EMP_PROJ into two separate relations EMP_LOC and EMP_PROJ1**
- **EMP_LOC means employee name ENAME works on some project a location PLOCATION**
- **EMP_PROJ1 means employee SSN works on project PNUMBER for HOURS at PLOCATION**
- **Is this a good decomposition?**

EMP_PROJ Decomposition

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith, John	X	Bellaire
123456789	2	7.5	Smith, John	Y	Sugarland
666884444	3	40	Narayan, Ramesh	Z	Houston
453453453	1	20	English, Joyce	X	Bellaire
453453453	2	20	English, Joyce	Y	Sugarland
...	...	...	...	...	...

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION
123456789	1	32.5	X	Bellaire
123456789	2	7.5	Y	Sugarland
666884444	3	40	Z	Houston
453453453	1	20	X	Bellaire
453453453	2	20	Y	Sugarland
...	...	...	...	...

EMP_LOC

ENAME	PLOCATION
Smith, John	Bellaire
Smith, John	Sugarland
Narayan, Ramesh	Houston
English, Joyce	Bellaire
English, Joyce	Sugarland
...	...

Is this a good decomposition?

Attempt to recover EMP-PROJ info with a JOIN

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION
123456789	1	32.5	X	Bellaire
123456789	2	7.5	Y	Sugarland
666884444	3	40	Z	Houston
453453453	1	20	X	Bellaire
453453453	2	20	Y	Sugarland
...	...	...	...	...

EMP_LOC

ENAME	PLOCATION
Smith, John	Bellaire
Smith, John	Sugarland
Narayan, Ramesh	Houston
English, Joyce	Bellaire
English, Joyce	Sugarland
...	...

Natural Join

SSN	PNUMBER	HOURS	PNAME	PLOCATION	ENAME
123456789	1	32.5	X	Bellaire	Smith, John
123456789	1	32.5	X	Bellaire	English, Joyce
123456789	2	7.5	Y	Sugarland	Smith, John
123456789	2	7.5	Y	Sugarland	English, Joyce
666884444	3	40	Z	Houston	Narayan, Ramesesh
453453453	1	20	X	Bellaire	English, Joyce
453453453	1	20	X	Bellaire	Smith, John
453453453	2	20	Y	Sugarland	English, Joyce
453453453	2	20	Y	Sugarland	Smith, John
...	...	...	...	...	...

Spurious Tuples

Spurious Tuples

- **Result of the Join produced <u>spurious tuples</u>, information which does not really exist**
- **Could not recover the information that was present in EMP_PROJ relation**
- **Problem: natural join criteria (equality) was on an attribute which is neither:**
  - **-a primary key in either relation**
  - **-a foreign key in one referencing the other**

Design Guideline

- Design Guideline #4
- Design relation schemas so that they can be joined with equality conditions on attributes which are either:

  -primary keys
  -foreign keys

  in a way that ensures no spurious tuples

95.305

Topics

- Design Guide-line for Relational Schemas
- Functional Dependencies
- Normal Forms
- 2nd and 3rd Normal Forms
- Boyce-Codd Normal Form

Formalizing the Guide-lines

- Functional Dependencies are a first step to formalizing the Design Guide-lines
- They are a very important concept in the design of relational schemas
- They lead to the creation of Normal Forms which avoid the problems we have discussed thus far

Functional Dependencies

- A Functional Dependency is a <u>constraint</u> on two sets of relation attributes
- Consider relation R={A1, A2, ..., An}
- Given two attribute subsets X and Y of R

  The functional dependency X -> Y means

  given any two tuples t1 and t2 from r(R),
  if t1[X] = t2[X], then t1[Y] = t2[Y]

- We say that functionally X determines Y

e.g. Functional Dependency

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

DNUMBER -> {DNAME, DMGRSSN}

- **Interpretation: if two tuples of EMP_DEPT have the same DNUMBER value, they also have the same DNAME and DMGRSSN values**
- **Imposing a functional dependency on a relation schema limits the tuples which would be considered legal**

Functional Dependencies

- **<u>Observations</u>**
- **if X is a candidate key of R, then**

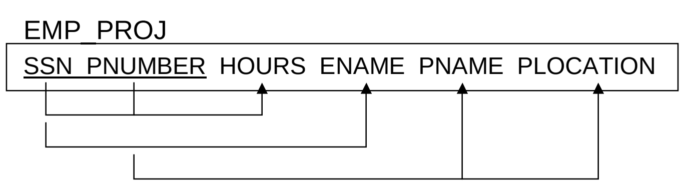
  X->Y hold for any subset Y of R

- **X -> Y in R does NOT imply Y -> X**

Functional Dependencies

- Functional Dependencies depend on the meaning of the attributes (what the designer meant them to represent)
- Functional Dependencies further describe the relation schema by placing constraints on its attributes which must hold <u>at all times</u>
- Functional Dependencies cannot be inferred from the table data, they must be explicitly stated by designer (based on meaning of relations)

example Functional Dependencies



SSN -> ENAME
PNUMBER -> {PNAME, PLOCATION}
{SSN, PNUMBER} -> HOURS

- Note diagrammatic representation on schema

Dependencies cannot be inferred

TEACH

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

- **Does TEXT -> COURSE?**
  -we cannot tell from the data it might be, but probably not

- **Does COURSE -> TEXT?**
  -No, we can tell this from the single counter example

Dependency Inference

- **Given a set of functional dependencies we can often infer others**

- **e.g. if**
  F= { SSN -> {ENAME, BDATE, ADDRESS, DNUMBER},
  DNUMBER -> {DNAME, DMGRSSN} }
- **Then the following is also true**

  SSN -> {DNAME, DMGRSSN},
  SSN -> SSN,
  DNUMBER -> DNAME

- **The designer will specify dependencies that are semantically obvious, but the database must also check for inferred dependencies**

Closure of a dependency set

- **e.g. if**
  F= { SSN -> {ENAME, BDATE, ADDRESS, DNUMBER},
  DNUMBER -> {DNAME, DMGRSSN} }
- **Then the following are also true**

  SSN -> {DNAME, DMGRSSN},
  SSN -> SSN,
  DNUMBER -> DNAME

- **the closure, F+, of a dependency set F is the set of all dependencies which can be inferred from F**

Inference Rules

(1)	**if Y is a subset of X, then X -> Y**	(reflexive)
(2)	**if X -> Y, then XZ -> YZ**	(augmentation)
(3)	**if X->Y, Y->Z, then X->Z**	(transitive)
(4)	**if X->YZ, then X->Y**	(decomposition)
(5)	**if X->Y, X-> Z, then X->YZ**	(union)
(6)	**if X->Y, WY->Z, then WX->Z**	(pseudotransitive)

Inference Rules

(1) if Y is a subset of X, then X -> Y (reflexive)

(2) if X -> Y, then XZ -> YZ (augmentation)

(3) if X->Y, Y->Z, then X->Z (transitive)

Rules 1-3 are sound and complete, known as Armstrong's inference rules, or Armstrong's Axioms

Proof of Decomposition Rule

(4) if X->YZ, then X->Y (decomposition)

Proof:

(a) X->YZ (given)

(b) YZ->Y (reflexive)

(c) X->Y (transitive)

Proof of Union Rule

(5) **if X->Y, X-> Z, then X->YZ** (union)

Proof:

(a) **X->Y, X->Z** (given)

(b) **X->XY** (augmentation, note xx->y == x->y)

(c) **XY->YZ** (augmentation)

(c) **X->YZ** (transitive)

Proof of Pseudo-Transative Rule

(6) **if X->Y, WY->Z, then WX->Z** (pseudotransitive))

Proof:

(a) **X->Y, WY->Z** (given)

(b) **WX->WY** (augmentation)

(c) **WX->Z** (transitive)

Closure of an Attribute

- **The closure, A+, of an attribute A, is the set of all attributes functionally determined by A**
- **e.g. given**
  F= { SSN -> ENAME,
  PNUMBER -> {PNAME, PLOCATION},
  {SSN, PNUMBER} -> HOURS }
- **We can calculate the following attribute closures**

  {SSN}+ = {SSN, ENAME}
  {PNUMBER}+ = {PNUMBER, PNAME, PLOCATION}
  {SSN, PNUMBER}+ = {SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS}

EXERCISE: You should be able to do this using the inference rules

Computing Attribute Closure

- **Let X be an attribute set such that X->W is an functional dependency in F**
- **You can compute X+ as follows**

```
X+ := X;
repeat
      oldX+ := X+;
      for each Y->Z in F do
            {if Y is a subset of X+
             then X+ := X+ union Z }
until (oldX+  =  X+)
```

Computing an Attributes Closure (example)

- F= { SSN -> ENAME,
  PNUMBER -> {PNAME, PLOCATION},
  {SSN, PNUMBER} -> HOURS }
- **compute** {SSN,PNUMBER}+

  x+ := {SSN, PNUMBER}
  x+ := {SSN, PNUMBER, HOURS} since {SSN,PNUMBER}->HOURS
  x+ := {SSN, PNUMBER, HOURS, ENAME} since SSN->ENAME
  x+ := {SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION}
  since PNUMBER->{PNAME, PLOCATION}
  return x+

Example Exercise

- **Given**
  R = (A,B,C,G,H,I)
  F = { A->B, A->C, CG->H, CG->I, B->H}
- **Is A->H logically implied by F?**

Example Exercise

- Given
  R = (A,B,C,G,H,I)
  F = { A->B, A->C, CG->H, CG->I, B->H}

- Is A->H logically implied by F?

- YES
  A -> B (given)
  B -> H (given)
  A -> H (transitive)

Example Exercise

- Given
  R = (A,B,C,G,H,I)
  F = { A->B, A->C, CG->H, CG->I, B->H}

- What is A+ ?

Example Exercise

- Given
  R = (A,B,C,G,H,I)
  F = { A->B, A->C, CG->H, CG->I, B->H}

- What is A+ ?

- A+ := A
  A+ := AB (since A->B)
  A+ := ABC (since A->C)
  A+ := ABCH (since B->H)
- so A+ = ABCH