

Lecture 15

Principles of Relational Database Design

Week 9

Overview

- Relational database design. Redundancy and update anomalies
- Functional dependencies (FDs), superkeys, candidate keys
- Closure of a set of FDs; Closure of a set of attributes under a set of FDs
- Minimal set of FDs
- Equivalence of FD sets

Consider the running example: the company database

(DESIGN #1)

Employee (ename, ssn, bdate, address, dno)

Department(dno, dmgrssn)

Department_locations(dno, dlocation)

Project(pname, pno, plocation, dno)

Works_on(ssn, pno, hours)

- In this example each relation represents an entity, a relationship, or a multivalued attribute
- Contrast this with the following design ...

(DESIGN #2)

**this relation represents
the employee's work relation**
(obtained by employee * department)



Employee_dept
(ename, ssn, bdate, address, dno, dname, dmgrssn)

Employee_proj
(ssn, pno, hours, ename, pname, plocation)



**this relation represents
the work commitments of an employee**
(obtained by employee * works_on * project)

- In the second design employee_dept represents information about two different entities, and so does employee_proj
- This is a problem that causes redundancy and update anomalies

①

Do not combine attributes from
multiple entity types into a single
relation
(preserve a clear relation *meaning*)

Employee_dept

ename, ssn , bdate , address , dno , dname, dmgrssn

peter , 123 , 230966 , 123_HighSt , 5 , Worx , 2323
paul , 765 , 120871 , 22_LowSt , 5 , Worx , 2323
mary , 332 , 010461 , 13_SpiderSt , 4 , Mgmt , 3445

.....

Problems:

(info ref to Dept is repeated
for each Employee working
for that Dept)

- This schema is redundant!
- Updates can easily bring the database into an inconsistent state
- Potentially expensive checks must be made at each update operation every time e.g. when we insert a new / modify an existing tuple
- There is a danger of a number of update anomalies

Problems analysed...

- Insertion anomalies:
How to insert a new department with **no** employees in it? (only by NULLs in the employee)
- Insertion / Modification anomalies:
How to make sure all redundant facts about a department remain consistent?
- Deletion anomalies:
how to delete the last employee of a department? (while preserving the dept info)

②

We should design databases to
avoid insertion and deletion
anomalies

The problem with NULL values...

- NULL values in database tables cause problems:
 - eg. what is the result of *average salary* if one salary value is NULL ?
- Ambiguous: NULL may mean a) the attribute is not applicable or b) that the value is not known / or known but not recorded yet

③

Avoid schemas which make it
necessary to fill many attributes
with **NULL** values

Spurious tuples...

- Suppose we use as a conceptual schema (the *good* company database design) and suppose also that...
 - Application Program #1 needs an external VIEW that lists employee names together with project locations (where the employee works)
 - Application Program #2 needs an external VIEW that lists employee ssn together with project number, the number of hours worked on the project, project name and location.

Spurious tuples...

This is no problem, since we can define the following views:

EMP_LOCS (ename, plocation) =

$\Pi_{\text{ename, plocation}} (\text{employee} \underset{\text{ssn=ssn}}{*} \text{works_on} \underset{\text{pno=pno}}{*} \text{project})$
(employee works on some project located in plocation)

EMP_PROJ1 (ssn, pno, hours, pname, plocation) =

$\Pi_{\text{ssn, pno, hours, pname, plocation}} (\text{employee} \underset{\text{ssn=ssn}}{*} \text{works_on} \underset{\text{pno=pno}}{*} \text{project})$
(employee with ssn works hours on the project pname, pnumber, plocation)

Spurious tuples...

Would it be a good idea to actually store these relations instead of *employee*, *works_on*, and *project* instead of deriving these tables as views?

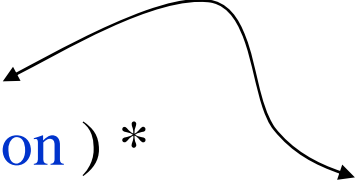
If we did that, then of course we would expect that the resulting database would be able to re-produce the same information as the one based on the original company schema.

Spurious tuples...

Eg. to list the names of employees together with the project numbers and hours worked on that project we should be able to reconstruct the original works_on relation, but with employee *names*

Works_on =
 Π ename, pno, hours
EMP_LOCS (ename, **plocation**) *
EMP_PROJ1(ssn, pno, hours, pname, **plocation**)

Join through 'plocation'



Unfortunately this will not work, because a join through **plocation** will produce unintended tuples!

Spurious tuples...

The reason for the unintended (spurious) tuples:

- an employee will join with all projects that are **at the same location**, no matter whether the employee actually works on the project or not!!!
- the reason lies in the problem that the two relations *are not related via a primary key / foreign key link* and unintended joins occur...

... hence this is a **bad** design

Emp_proj1

SSN	PNO	HOURS	PNAME	PLOCATION
123456789	1	33	ProductX	Bellaire
123456789	2	8	ProductY	Sugarland
666884444	3	40	ProductZ	Houston
453453453	1	20	ProductX	Bellaire
453453453	2	20	ProductY	Sugarland
333445555	2	10	ProductY	Sugarland
333445555	3	10	ProductZ	Houston
333445555	10	10	Computerization	Stafford
333445555	20	10	Reorganization	Houston
999887777	30	30	Newbenefits	Stafford
999887777	10	10	Computerization	Stafford
987987987	10	35	Computerization	Stafford
987987987	30	5	Newbenefits	Stafford
987654321	30	20	Newbenefits	Stafford
987654321	20	15	Reorganization	Houston
888665555	20		Reorganization	Houston

Emp_locs

ENAME	PLOCATION
Ahmad Jabbar	Stafford
Alicia Zelaya	Stafford
Franklin Wong	Houston
Franklin Wong	Stafford
Franklin Wong	Sugarland
James Borg	Houston
Jennifer Wallace	Houston
Jennifer Wallace	Stafford
John Smith	Bellaire
John Smith	Sugarland
Joyce English	Bellaire
Joyce English	Sugarland
Ramesh Narayan	Houston

Original Works_on

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
999887777	30	30
999887777	10	10
987987987	10	35
987987987	30	5
987654321	30	20
987654321	20	15
888665555	20	

Reconstructed Works_on with employee names

• ENAME	PNO	HOURS
• John Smith	1	33
• Joyce English	1	33
• John Smith	1	33
• Joyce English	1	33
• John Smith	1	20
• Joyce English	1	20
•		
• Joyce English	2	10
• John Smith	2	10
• Joyce English	2	10
• Franklin Wong	2	10
•		

148 rows selected !

④

Design database schemas so that
no spurious tuples could occur
through joins

To sum up:

Database Design Guidelines:

1. Do not combine attributes from multiple entity types into a single relation - preserve a clear meaning
2. Design the DB so that no insertion, update, deletion anomalies may occur.

Database Design Guidelines:

3. Avoid placing attributes in a base relation whose values may be frequently NULL.

(In the ER-Relational mapping - if possible, map 1:1 relationship types on an attribute of the relation that represents the **totally** participating entity type)

4. Do not have relations that contain matching attributes other than fk-pk combinations.

To be continued...