

Key Points – Design Principles for Relational Schemas

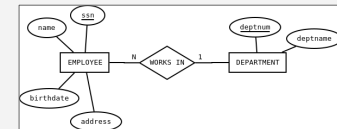
- The meaning of each relation should be easy to describe.
 - avoid combining multiple entity types and relationship types into one relation
 - tradeoff may be that the schema no longer captures the total participation constraint
- Reduce redundant information and avoid update anomalies.
- Reduce the number of NULL values.
 - avoid creating relations where attributes are often NULL
- Don't allow the possibility of generating spurious tuples.
 - occurs when a foreign key refers to something other than the primary key of the other table
 - can occur if tables are decomposed improperly

Update Anomalies

- update anomalies* are problematic conditions that can arise during/from updates

- consider a merged relation approach for WORKS_IN

WORKS_IN(name, ssn, birthdate, address, deptnum, deptname)



- an *insertion anomaly* occurs when insertion is blocked because required values are missing
 - e.g. can't add a department with no employees
- a *deletion anomaly* occurs when deletion of some information results in the loss of something else, or some deletions need to be handled specially
 - e.g. deleting the last employee from a department means department info is lost
- a *modification anomaly* occurs when information is duplicated and it is possible to update one copy without updating the others
 - e.g. changing part of a department's information requires changing every tuple where an employee works for that department

Using the merged relation approach for Manages results in a single relation for the Employee, Department, and Manages entity types. The following is one possibility:

```
Department(deptnum, deptname, ssn, empname, address, birthdate, salary, sex, startdate)
```

- Department.ssn NOT NULL

What kind(s) of update anomalies can occur?

insertion anomaly
 deletion anomaly
 modification anomaly
 none of the above

Using the foreign key approach for Manages results in a single relation for Manages and either Employee or Department. The following is one possibility:

```
Employee(ssn, firstName, lastName, address, birthdate, salary, sex, deptnum, startdate)
Department(deptnum, name)
```

- Employee.deptnum -> Department.deptnum
- Employee.deptnum is NOT NULL

What kind(s) of update anomalies can occur?

insertion anomaly
 deletion anomaly
 modification anomaly
 none of the above

- the existence of update anomalies means that these are not good mappings

Spurious Tuples

original data	ssn	projnum	hours	name	projname	projloc
	123456789	1	32.5	John B Smith	ProductX	Bellaire
	453453453	1	20.0	Joyce A English	ProductX	Houston

WORKS_ON(ssn, projnum, hours, name)
PROJECT(projnum, projname, projloc)

WORKS_ON.projnum ->
PROJECT.projnum

ssn	projnum	hours	name	projnum	projname	projloc
123456789	1	32.5	John B Smith	1	ProductX	Bellaire
453453453	1	20.0	Joyce A English	1	ProductX	Houston

combining the tables results in more tuples than we started with – and no way to reconstruct just the original

ssn	projnum	hours	name	projname	projloc
123456789	1	32.5	John B Smith	ProductX	Bellaire
123456789	1	32.5	John B Smith	ProductX	Houston
453453453	1	20.0	Joyce A English	ProductX	Bellaire
453453453	1	20.0	Joyce A English	ProductX	Houston

- getting spurious tuples means this is not a good decomposition

Spurious Tuples

- avoiding problems
 - model carefully and identify all functional dependencies
 - an employee works for a particular project in a particular place, not just a particular project
 - ensure that FKs only refer to complete PKs

original
data

ssn	projnum	hours	name	projname	projloc
123456789	1	32.5	John B Smith	ProductX	Bellaire
453453453	1	20.0	Joyce A English	ProductX	Houston

`WORKS_ON(ssn,projnum,hours,name,projloc)`
`PROJECT(projnum,projname,projloc)`

`WORKS_ON.projnum,WORKS_ON.projloc → PROJECT.projnum,PROJECT.projloc`