Normalization

The Normal Form Hierarchy

- **A top-down approach to relational design (as opposed to ER diagramming, which is bottom-up)**
- **1NF, 2NF, 3NF, BCNF, 4NF, 5NF**
- **Higher normal forms are more restrictive.**
- **There is a greater number of tables as you go up to a higher normal form.**
- **If a table is in a higher NF, then it is in all of the lower NFs.**
  - **(Only by convention, really, as it is possible, e.g., for a relation to be consistent w/ 3NF def and not w/ 2NF)**
- **Note that:**
  - **We strive to achieve 3NF**
  - **If you use the ER model properly and use the translation techniques properly, the relational schema are usually in 3NF.**

Update Anomalies

- Update anomalies are one of the problems we are looking to avoid by employing normalization.
- They result from uncontrolled redundancy resulting from poor design decisions.
- Note that both tables contain two separate concepts
- Updating a value of, for example, Dname, requires changing multiple records. If Dname is not changed for all employees in the department in question, an *update anomaly* occurs

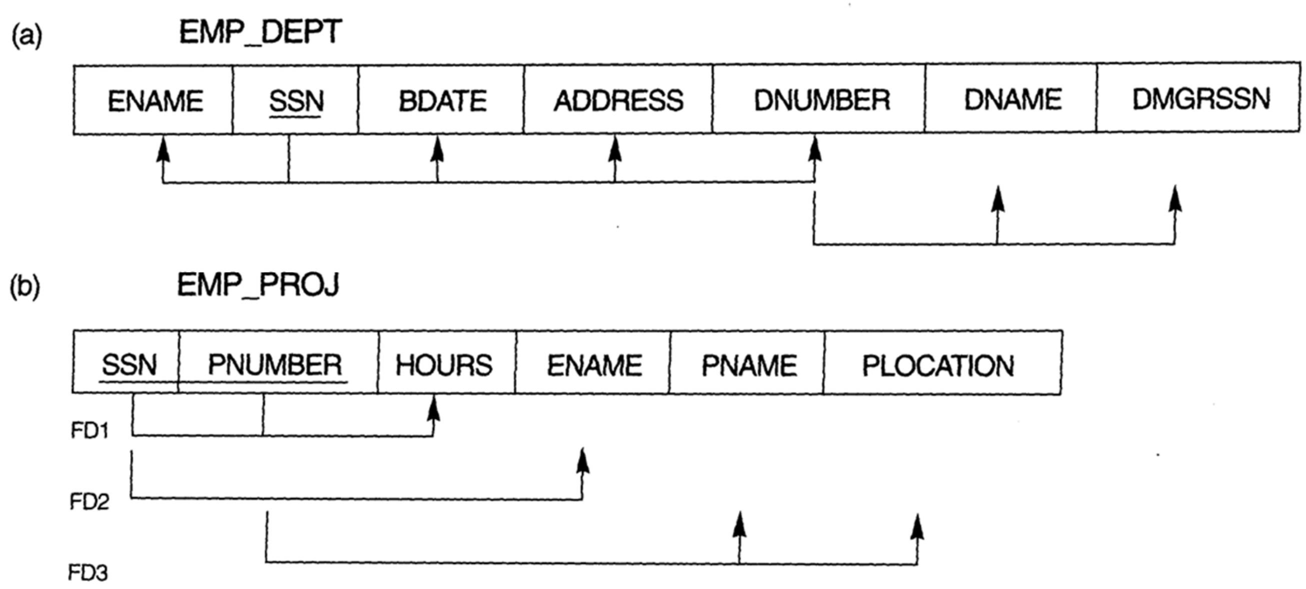


FIGURE 10.3 Two relation schemas suffering from update anomalies.

Update Anomaly Types

- Insert Anomalies
  - Can’t insert a new department into emp_dept above that has no employees yet.
  - Every new employee requires consistent re-entry of all details for that employee’s department.
- Deletion Anomalies
  - What happens when you delete the last employee from the R&D dept? All data about that department is gone(!)
- Modification Anomalies
  - If we change some info about a department, we have to change it consistently across all employees who work for that department
    - (Not as big a deal b/c a single SQL query will still do the job.)
- Related problem: null values proliferate in poorly designed relations (eg employees with no departments)

Spurious Tuples

- Tuples that result from a ‘lossy join’
  - Lossy join: joined table contains tuples that did not exist in underlying tables
- Decomposition per normalization avoids lossy joins.
- Example:
- Base Relation:
  - Emp-Proj (SSN, P#, Hours, Ename, Pname, Ploc)
    - (111-11-1111, P12, 40, Bill, ProjX, Houston)
    - (222-22-2222, P37, 20, Mary, ProjY, Houston)
- Suppose we decompose to:
  - R1(<u>Ename, Ploc</u>)
    - (Bill, Houston)
    - (Mary, Houston)
  - R2 (<u>SSN, P#,</u> Hours, Pname, Ploc)
    - (111-11-1111, P12, 40, ProjX, Houston)
    - (222-22-2222, P37, 20, ProjY, Houston)
- What happens when we do a natural join?
  - We gain nonsense (‘spurious’) tuples
  - Data integrity is lost after join → ‘lossy’ join

Functional Dependencies

- X ➔ Y
- “X functionally determines Y”
- For a given value of X there can be one and only one value of Y
- If you know X, you also know with certainty Y
- If you know a person’s SSN, you can determine for sure their first name
- If you know a person’s first name you CANNOT determine their SSN (b/c hundreds of thousand of people, all w/different SSNs, may share that name!)

Functional Dependencies

- B is a function of A if for every A there is at most one value for B
  - Sound familiar?
    - Similar to cardinality, but attribute is unit of measure
  - Representation:
    - $A \rightarrow B$
  - Interpretation:
    - “B is a function of A” or
    - “A functionally determines B”

Two Entities, A & B

- Cardinality Analogy: Functional Dependencies
  - If A:B are 1:1

    A → B & B → A
  - If A:B are 1:N

    B → A ONLY
  - If A:B are N:1

    A → B ONLY
  - If A:B are N:M

    There are NO functional dependencies
- Note, though, that FD pertains to *attributes* where cardinality pertains to *relationships*

FD Inferences

- An FD is a property of the relational schema, NOT any particular state of the relation
  - Have to have a priori knowledge of the semantics to know which FDs hold
    - (Just like you did with cardinality, btw)
  - HOWEVER, FDs can be ruled *out* by looking at the data.

FD Example

- Which FDs can be ruled *out* here?

TEACH

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

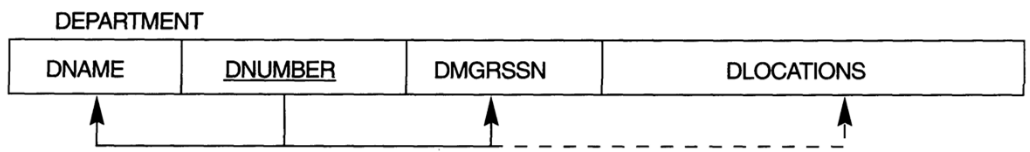
Normalization Terminology

- Superkey – set of attributes assuring uniqueness
  – (any set that contains one or more candidate keys)
- Candidate key – *minimal* set of attributes assuring uniqueness
- Primary Key – one set of attributes arbitrarily chosen from among set of candidate keys to act as PK
- Prime attribute – member of some candidate key
- Nonprime attribute – *not* a member of any candidate key

First Normal Form

- A relation *R* is in 1 NF if all attributes have atomic values

(a)



(b) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c) DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

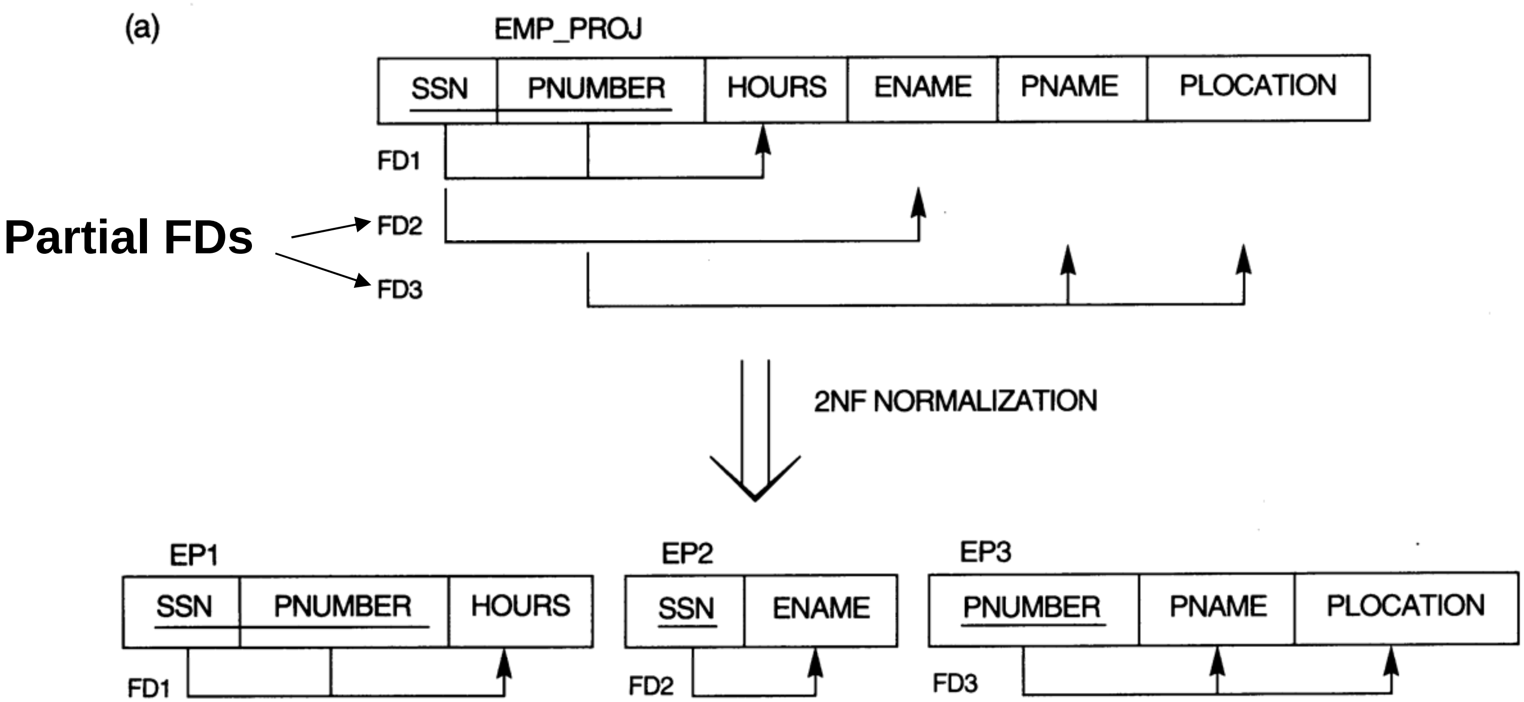
Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

- Either accept redundancy, like c) above, or (far better) convert to a multi-valued attribute weak entity situation.
- (Fairly) recent trends – object DBs, XML allows nesting that violated 1NF
- B *is NOT* in 1 NF, C *IS* in 1 NF

Second Normal Form

- A relation R is in 2 NF if every nonprime attribute A in R is *fully* functionally dependent on the primary key of R.
  - I.e., 2NF relations have no *partial dependencies*
    - Issue only with relations w/ composite PKs
    - The test for partial dependencies:
      - Take away a part of the concatenated key, if any FD still holds, then there is a *partial dependency*
        - » *Relation is NOT in 2NF*

2NF Example



(a)
EMP_PROJ
SSN
PNUMBER
HOURS
ENAME
PNAME
PLOCATION
FD1
Partial FDs
FD2
FD3
2NF NORMALIZATION
EP1
SSN
PNUMBER
HOURS
FD1
EP2
SSN
ENAME
FD2
EP3
PNUMBER
PNAME
PLOCATION
FD3

Third Normal Form

- Definition: A relation is in 3 NF if it satisfied 2NF conditions and it has no *transitive dependencies:*
- *X→Z, but only because X→Y and Y→ Z is a transitive dependency*
- *EX:*
- *Emp#, EmpName, Phone, Salary, Department#, Dname, Dlocation*
- *Not in 3NF because Emp#→Dname or Dloc only because Emp#→Dept# and Dept# in turn → Dname, Dloc*

3NF Example

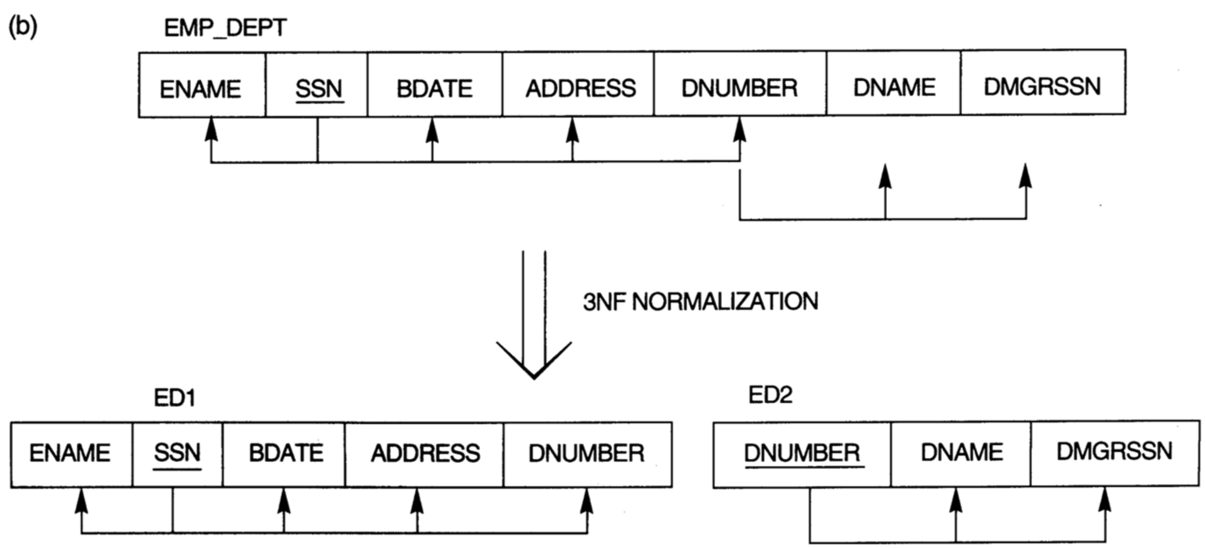


Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

- Dnumber → Dname
- Dnumber → Dmgrssn
- Dnumber is not a key
- Therefore (b) above is NOT in #NF

3NF Issues

- If relation is not in 3nf, susceptible to update anomalies
- Note that a Non 3NF table typically contains two concepts that are in a 1:N relationship as in the EMP_Dept (N:1) example
- Remedy:
  - Create a separate relation for each set of directly dependent attributes
- If you employ ER modeling properly and translate correctly, the resulting schema will be in 3NF.
- In practice, we usually stop at 3NF as higher NFs do not occur frequently.

Normalizing to 3NF

- Every non-key attribute must provide a fact about “the key, the whole key, and nothing but the key”
- Give attributes with transitive dependencies their own table

The Bottom Line

- If you apply the rules of semantic modeling that we learned in class (i.e. our process for ER diagramming and Translation), your resultant relational schema should *be in 3rd normal form anyway!*

- Higher NF = More tables
- More tables =
  - Degraded performance
  - Query complexity
  - Referential integrity challenges
- Practical Approach
  - Use semantic approach like ER model
  - Convert to schema
  - Check for 3NF
  - If absent, normalize to 3NF
  - Ignore normalization wisely, consciously, typically in the name of performance: *denormalization*

Other Normal Forms

- Normal forms up to 6th, plus Boyce-Codd Normal Form exist, but are rare in practice and beyond the scope of this class
- See http://en.wikipedia.org/wiki/Database_normalization#Normal_forms

for a good discussion of these forms if interested.

(Optional) See the following slides for BCNF, 4NF, 5NF, very informally presented

Higher Normal Forms

- BCNF, 4NF, 5NF
- Hard to recognize, unimportant in practice
- BCNF:
  - If X→Y in R, then X must be a superkey of R
- Example:
  - Property (<u>Prop#</u>, County, Lot#, Area)
  - Prop#→ County, Lot#, Area
  - Area → County
  - Area is not a superkey but it IS prime, so OK per 3NF, violate BCNF
  - (<u>Prop#</u>, Area, Lot#) & (<u>Area</u>, County)

4NF

- Involves multi-valued dependencies
- Emp (<u>Ename, Pname, Dname</u>)
  - Employee can have multiple dependents and work on multiple departments
  - Break out to
    - (Ename, Pname) and (Ename, Dname)

5 NF

- Pertains to ternary relationships
- Looks to avoid “join dependencies”
- Supply (<u>Sup#, Part#, Proj#</u>)
- IF it is the case that:
  - For a supplier S that supplies part P and
  - A project X that uses part P and
  - Supplier S supplies at least one part to project X
  - Then supplier S will also necessarily supply part P to project X
- THEN Supply is NOT in 5NF and should be
  - (<u>Sup#, Part#</u>), (<u>Sup#, Proj#</u>), (<u>Part#, Proj#</u>)
- Review from ERD perspective
- Obviously requires intuition/skill to identify JDs
  - Really, really not worried about in practice