UPPSALA UNIVERSITET

Normalization and Data Quality (2)

Lecturer: Neena Thota

neena.thota@it.uu.se

Where are we?

Database design

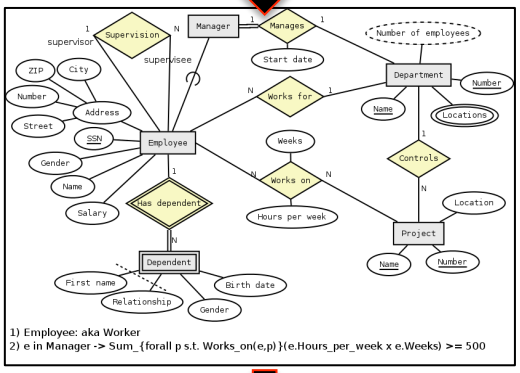
- Intro
- Database design 4 lectures
- LABS 1
- Data quality / Normalization 2 lectures
- LABS 2
- SQL

- Motivation & terminology
- The (E)ER conceptual model
- The relational model
- From ER to relational
- SQL and DBMSs (DDL)

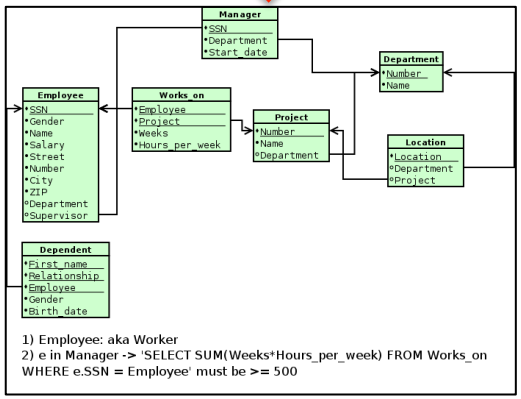
Natural language

An enterprise consists of a number of departments. Each department has a name, a number, a manager, and a number of employees. The starting date for every department manager should also be registered. A department can have several locations. Every department controls a number of projects. Each project has a unique name, a unique number (both unique only inside the project's department) and a location. For each employee, the following information is kept: name, social security number, address, salary and sex. An employee works for only one department but can work with several projects that can be related to different departments. An employee may also supervise one or more other workers. Information about the number of hours (per week) that an employee works with each project should be stored – to be a manager one must have worked at least 500 hours on projects. We also want to keep track of the dependents of each employee, for insurance purposes. We keep each dependent's first name, sex, birth date and relationship to the worker.

Entity-Relationship diagram



Relational model



SQL / DBMS


```
create relation EMP
(SSN int ,
Name varchar,
…
```

UPPSALA UNIVERSITET

Intended Learning Outcomes

- Perform normality tests for 1, 2, 3 and BCNF.
- Apply inference rules for Functional Dependencies.
- Improve the design of poorly designed schemata.

NORMALIZATION

UPPSALA UNIVERSITET

Review - Design Guidelines

- **Avoid Anomalies**
  - cause redundant work to be done during insertion / modification of a relation
  - may cause accidental loss of information during a deletion from a relation.
- **Avoid waste of storage space due to NULLs**
  - difficulty of performing selections, aggregation operations, and joins due to NULL values.
- **Avoid generation of invalid and spurious data during joins on base relations with matched attributes**
  - may not represent a proper (foreign key, primary key) relationship.

UPPSALA UNIVERSITET

Review - Normalization

- ***1st Normal Form*** **(1NF)**
  - No multivalued attributes or nested relations.
- ***2nd Normal Form (2NF)***
  - It is in 1 NF.
  - Where PK contains multiple attributes, all non-key attributes must be fully functionally dependent on the primary key. No partial dependencies are allowed.
- ***3rd Normal Form (3NF)***
  - It is in 2 NF.
  - No transitive dependencies are allowed.
    - No non-prime attribute is allowed to be FFD of any other non-prime attribute.

UPPSALA UNIVERSITET

Ex1: Normalize into 1NF

A relation is in 1NF if all attributes contain only atomic values.

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

UPPSALA UNIVERSITET

Ex1. Choices for 1NF

(1) Separate relation

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

(2) Expand key

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

(3) Replace with max (known) number of values

Dnumber	Dlocation1	Dlocation2	Dlocation3

Ex1. Three approaches to achieve 1NF

1. Remove **Dlocations** and put into separate relation (e.g. DEPT_LOCATIONS) along with PK of DEPARTMENT. PK of new relation will be **{Dnumber, Dlocation}**.
   [The best approach]

   A distinct tuple in DEPT_LOCATIONS exists for *each location* of a department.

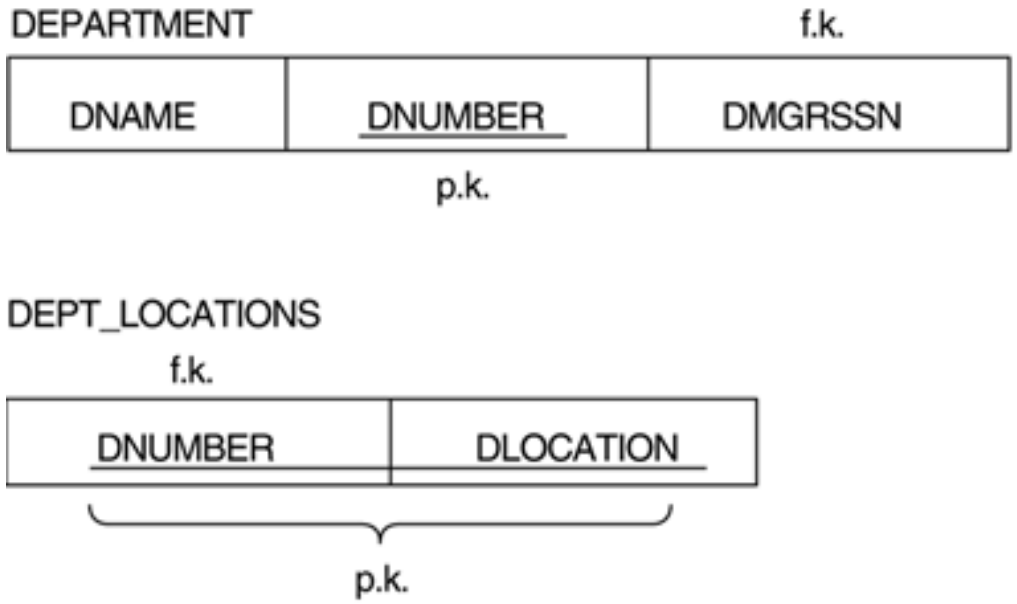
2. Expand the key so there will be a separate attribute in original DEPARTMENT relation for each location of a department. PK becomes **{Dnumber, Dlocation}**.
   [Introduces redundancy]

3. If the maximum number of values is known for **Dlocations**, replace it with atomic attributes **Dlocation1, Dlocation2, . . . , Dlocation*n***.

   [Introduces NULLs]

Ex1. The best approach

- Remove attribute *Dlocations* and place it in a separate relation **Dept_locations** along with primary key *Dnumber* of Department.
- The Primary Key of **Dept_locations** is combination (*Dnumber,Dlocation*).



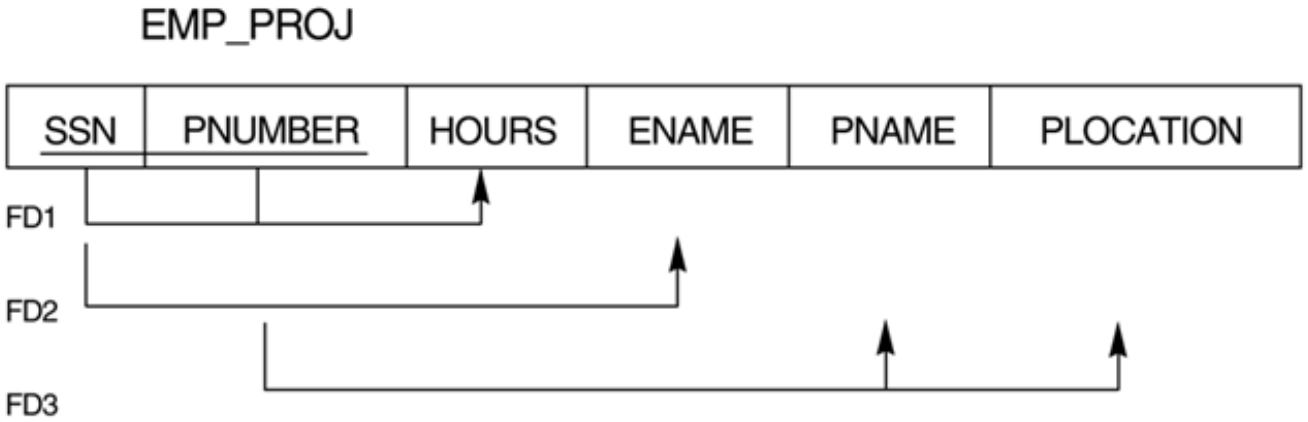
UPPSALA UNIVERSITET

Steps to Normalize a Relation into 1NF

1. Remove the **multi-valued** attribute or nested relations and create a new relation to contain it.
2. Add to the new relation a copy of the PK of the original relation.
3. Determine the PK of the new relation.

UPPSALA UNIVERSITET

Ex2. Normalize into 2NF



Determine full/partial dependencies

FDs

Ssn, Pnumber → Hours

Ssn → Ename

Pnumber → Pname, Plocation

??

Ssn → Hours

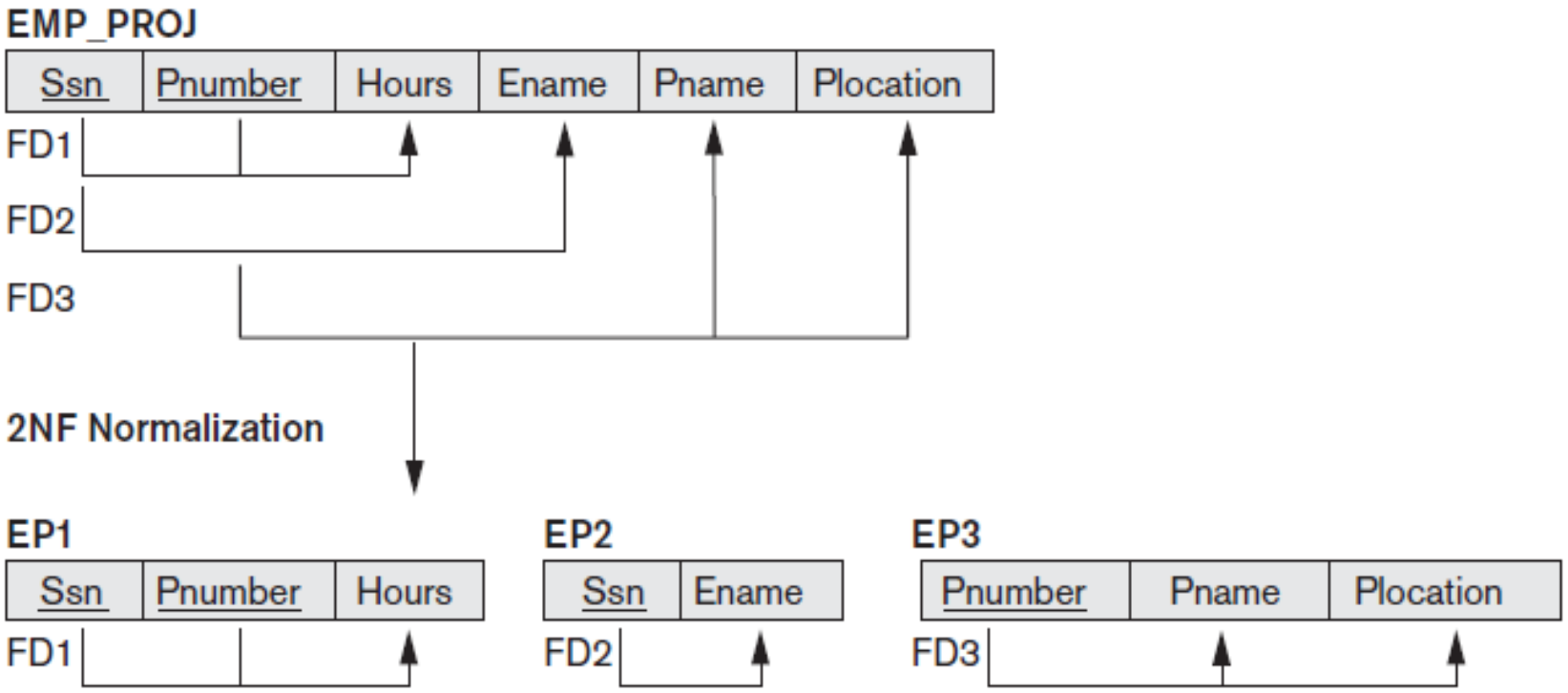
Pnumber → Hours

SSn, Pnumber → Ename

UPPSALA UNIVERSITET

Ex2. Approach - Normalize into 2NF

A relation schema R is in 2NF if every nonprime attribute in R is fully functionally dependent on the primary key of R.

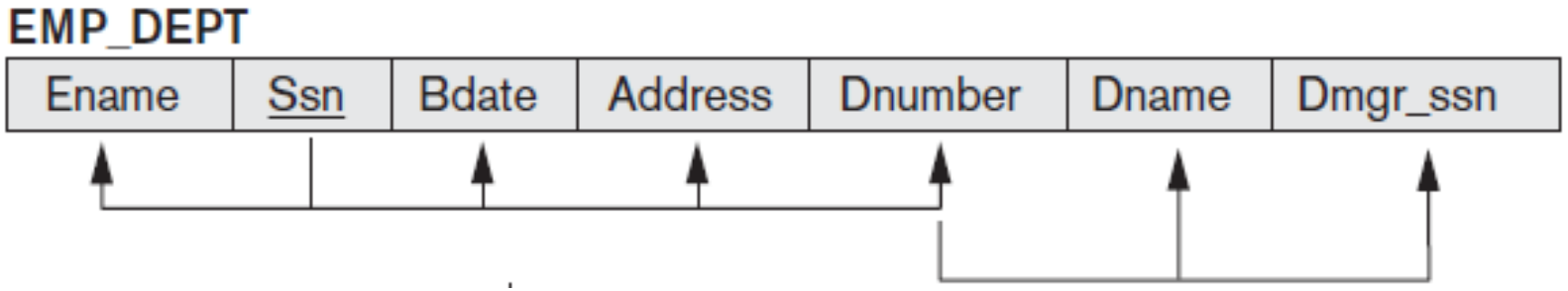


UPPSALA UNIVERSITET

Steps to Normalize a Relation into 2NF

1. Remove **nonprime attributes** that are only partially functionally dependent on PK, and place them in a new relation.
2. Add to this relation a copy of the attribute(s) which are **determinants** of these nonprime attributes. These attribute(s) will automatically become PK/s of this new relation.

Ex3. Normalize into 3NF



- FFD?
- Transitive dependencies?
- Dnumber→Dmgr_ssn is transitive through Dnumber in EMP_DEPT as

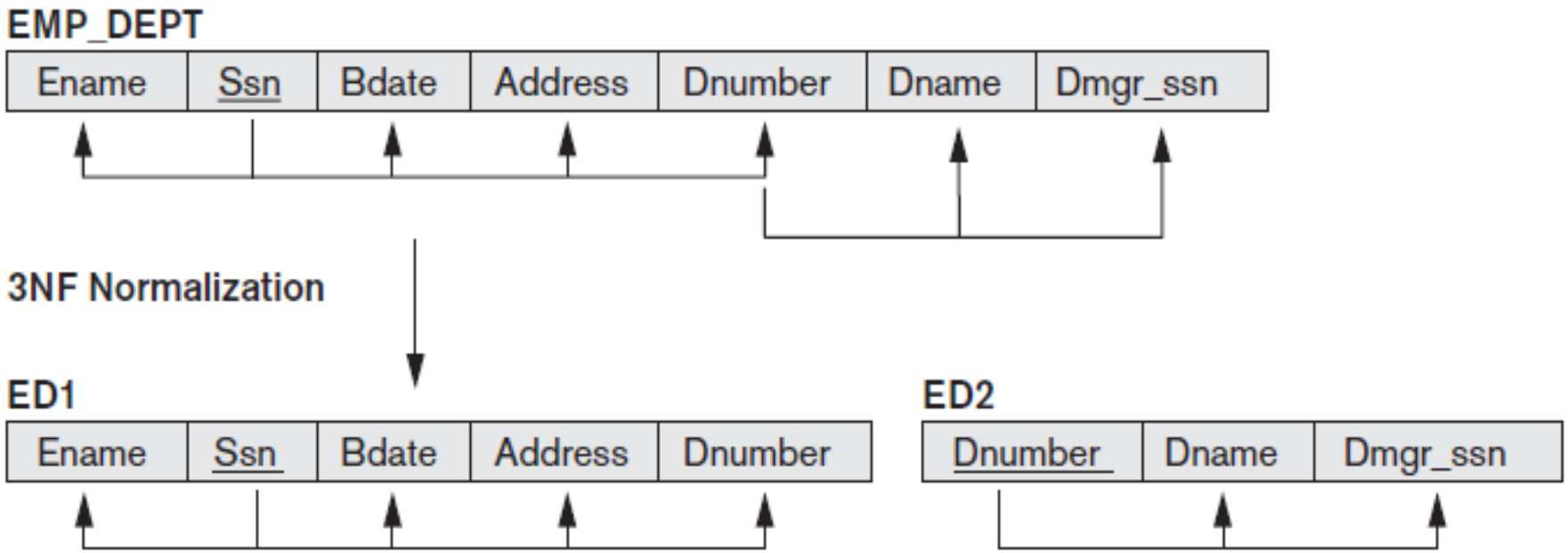
  Ssn → Dnumber

  Dnumber → Dmgr_ssn
- Note – Dnumber neither a key itself nor a subset of the key of EMP_DEPT.

Ex3. Approach - Normalize into 3NF

A relation schema R is in third normal form (3NF) if

- it is in 2NF
- no non-prime attribute A in R is transitively dependent on the primary key



UPPSALA UNIVERSITET

Steps to Normalize a Relation into 3NF

1. Remove the attributes with **transitive dependency** and place them in a new relation.
2. Add to this relation a copy of the attribute(s) which are the **determinants** of these nonprime attributes. These attribute(s) will automatically become the PK/s of this new relation.

UPPSALA UNIVERSITET

Note

- In X → Z and Z → Y, with X as the primary key, we consider this a problem only if Z is not a candidate key.
- If Z is a candidate key, there is no problem with the transitive dependency .
- E.g:
  - Consider EMP (SSN, Emp#, Salary).
  - Here, SSN → Emp# → Salary Emp# is a candidate key.

UPPSALA UNIVERSITET

Summary: Normal forms and Steps

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

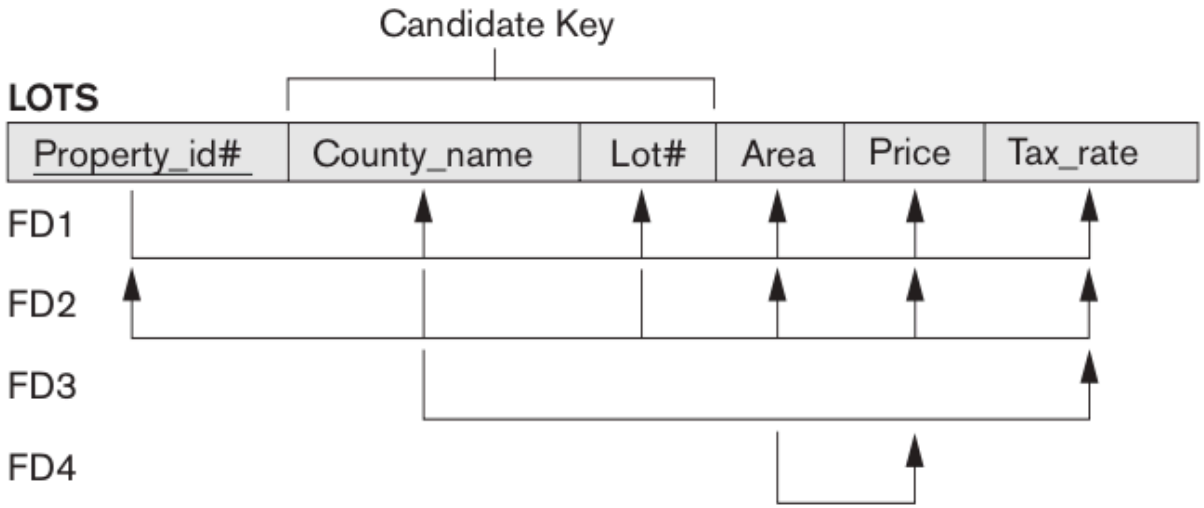
UPPSALA UNIVERSITET

BCNF

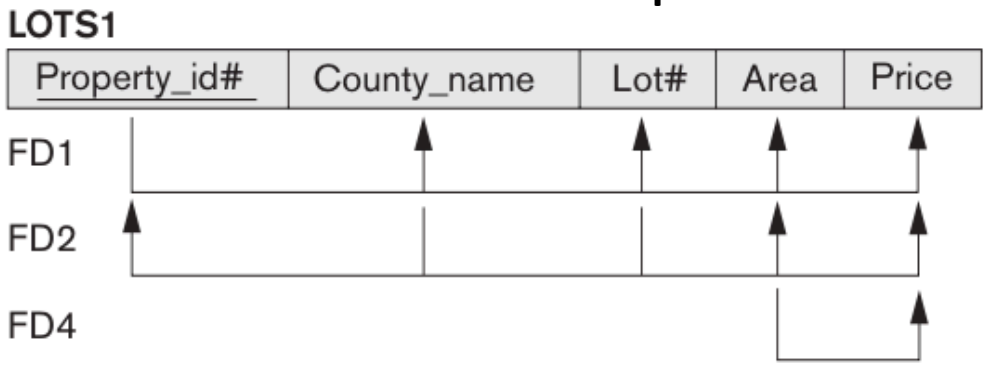
- ***Boyce-Codd Normal Form (BCNF)***
  - It is in 3NF.
  - Every determinant in a relation is a candidate key.
  - If there is only one candidate key, 3NF and BCNF are one and the same.
  - The difference between BCNF and 3NF is that in BCNF a prime attribute cannot be FFD of a non-prime attribute.
  - Therefore BCNF is a stricter condition.

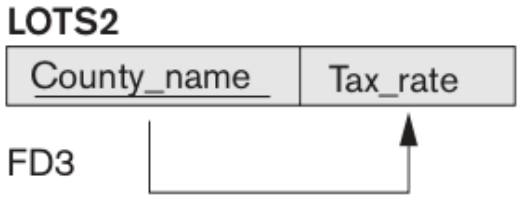
UPPSALA UNIVERSITET

Example BCNF (1)



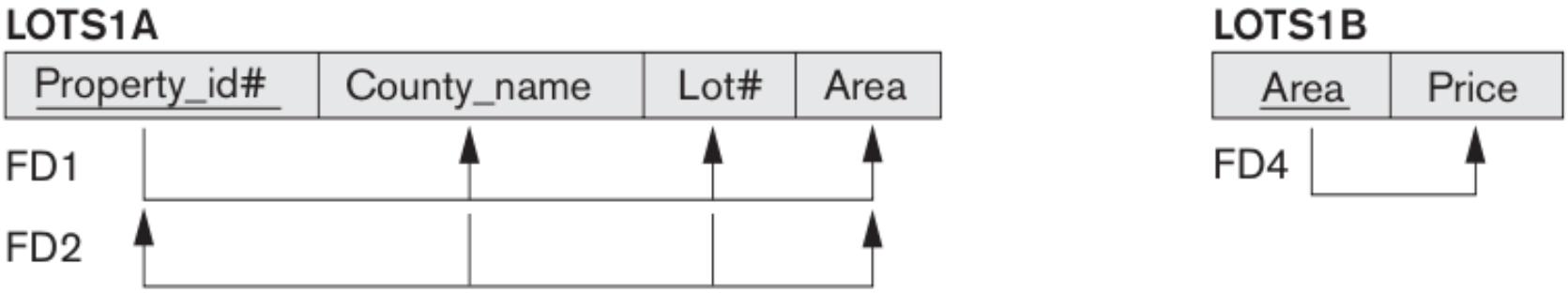
Decompose into 2NF



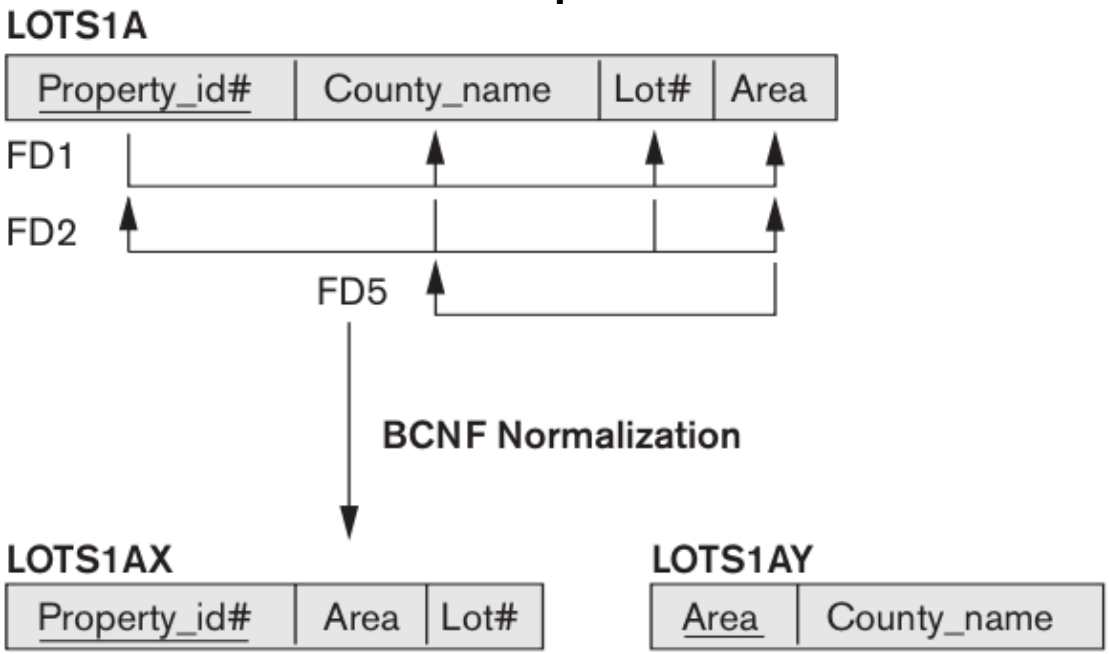


Example BCNF (2)

Decompose into 3NF



Decompose into BCNF



UPPSALA UNIVERSITET

Practical usage of normal forms

- Normal forms can be verified on each relation at a time.
- This does not guarantee that the whole schema is satisfactory.
- Two additional properties:
  - **Lossless join property**: no spurious tuples are generated after recomposing the relations.
  - **Dependency preservation property**: each functional dependency is represented in some relation.

Wrap-up

- Each normal form implies the previous one:
  - If a relation is in BCNF, it is also in 3NF, 2NF and 1NF.
  - If it is in 3NF, it is also in 2NF and 1NF.
  - If it is in 2NF, it is also in 1NF.
- BCNF is a strong condition.
- It is not always possible to transform (through decomposition) a schema to BCNF and keep the dependencies.
- 3NF has the most of BCNF's advantages and can still be fulfilled without giving up dependencies or the lossless-join property.
- 3NF admits some sort of redundancy that BCNF does not allow.

INFERENCE RULES

UPPSALA UNIVERSITET

Inferring Dependencies

Ex: Infer additional dependences from following:

$F = \{\text{Ssn} \rightarrow \{\text{Ename, Bdate, Address, Dnumber}\}, \text{Dnumber} \rightarrow \{\text{Dname, Dmgr\_ssn}\}\ \}$

- Dependencies can be **inferred or deduced** from the FDs in F, the set of functional dependencies that are specified on relation schema R.
- Set of **inference rules** can be used to define a concept called **closure** formally that includes all possible dependencies that can be inferred from the given set F.

UPPSALA UNIVERSITET

Inference Rules for FDs

1. **IR1** (reflexive rule)1:
   If $X \supseteq Y$, then $X \rightarrow Y$.
2. **IR2** (augmentation rule)2:
   $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.
3. **IR3** (transitive rule):
   $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.
4. **IR4** (decomposition, or projective, rule):
   $\{X \rightarrow YZ\} \models X \rightarrow Y$.
5. **IR5** (union, or additive, rule):
   $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$.
6. **IR6** (pseudotransitive rule):
   $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$.

UPPSALA UNIVERSITET

E.g. Computing Closure of a Set

Def: The "**closure**" of a set of attributes is the full set of attributes determined by them.

*A minimal set of attributes whose closure includes all the attributes in R is a **candidate key**.

E.g.

G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1: {A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Result:

- None of single attributes is a key
- Next calculate the closures of pairs of attributes that are possible keys.

UPPSALA UNIVERSITET

E.g. Computing Closure & Possible Keys

Given:

G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1:

{A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Step 2:

{A, B} -> {A, B, C, I},

{B, D} -> {B, D, E, F},

{A, D} -> {A, D, G, H, I, J}

Result:

None of these pairs are keys either since none of the closures includes all attributes.

UPPSALA UNIVERSITET

E.g. Computing Closure – Finding Key

Given:

G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1:

{A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Step 2:

{A, B} -> {A, B, C, I}, {B, D} -> {B, D, E, F}, {A, D} -> {A, D, G, H, I, J}

Step 3

The union of the three closures includes all the attributes:

{A, B, D} -> {A, B, C, D, E, F, G, H, I}

- Hence {**ABD**} is a key.

UPPSALA UNIVERSITET

E.g. Decomposing into Relations

Given:

G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1: {A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Step 2: {A, B} -> {A, B, C, I}, {B, D} -> {B, D, E, F}, {A, D} -> {A, D, G, H, I, J}

Step 3: The union of the three closures includes all the attributes: {A, B, D} -> {A, B, C, D, E, F, G, H, I}. Hence {ABD} is a key.

Relation R = {A, B, C, D, E, F, G, H, I}

- Check for first-level partial dependencies on key (which **violate 2NF**) from step 2: {A, B} -> {C, I}, {B, D} -> {E, F}, {A, D} -> {G, H, I, J}

UPPSALA UNIVERSITET

Normalizing to 2NF

Given: G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1: {A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Step 2: {A, B} -> {A, B, C, I}, {B, D} -> {B, D, E, F}, {A, D} -> {A, D, G, H, I, J}

Step 3: The union of the three closures includes all the attributes: {A, B, D} -> {A, B, C, D, E, F, G, H, I}. Hence {ABD} is a key.

Step 4:

R is decomposed into R1, R2, R3, R4 (keys are underlined):

R1 = {A, B, C, I}, **R2** = {B, D, E, F},

R3 = {A, D, G, H, I, J}, **R4** = {A, B, D}

- Additional partial dependencies exist in R1 and R3 because {A} -> {I}. Hence, we remove {I} into R5:

  R5 = {A, I}

R1, R2, R3, R4, R5 are the result of **2NF** decomposition.

UPPSALA UNIVERSITET

Normalizing to 3NF

Given:

Given: G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Step1: {A} -> {A, I}, {B} -> {B}, {C} -> {C}, {D} -> {D}, {E} -> {E}, {F} -> {F}, {G} -> {G}, {H} -> {H, J}, {I} -> {I}, {J} -> {J}

Step 2: {A, B} -> {A, B, C, I}, {B, D} -> {B, D, E, F}, {A, D} -> {A, D, G, H, I, J}

Step 3: The union of the three closures includes all the attributes: {A, B, D} -> {A, B, C, D, E, F, G, H, I}. Hence {ABD} is a key.

Step 4: R1 = {A, B, C}, R2 = {B, D, E, F}, R3 = {A, D, G, H, J}, R4 = {A, B, D}, R5 = {A, I}

Step 5:

Check for dependencies between **non-prime attributes** in each of the relations (which violate 3NF).

- Only **R3** has transitive dependency {H} -> {J}, so decompose into R31 and R32:

  R31 = {H, J}, **R32** = {A, D, G, H}

- The final set of 3NF relations is {**R1, R2, R31, R32, R4, R5**}

UPPSALA UNIVERSITET

Wrap up

- Normalizing the data model to improve its quality.
- Based on theoretical concepts:
  - FD, FFD, prime attributes.
- Guarantees normal forms without losing dependencies.

DATA QUALITY

UPPSALA UNIVERSITET

Criteria

- Accuracy - information system represents a real world state different from the one that should be represented.
- Completeness – missing attributes and values.
- Data consistency – missing domain and key constraints and functional dependencies.
- Data interpretability – obscure meaning due to absence of standards codes.
- Schema interpretability – ambiguous attribute names.
- Schema normalization – not reaching at least 3NF.

UPPSALA UNIVERSITET

Summary Data Quality

- Data quality is one of the main problems in real systems.
- Do a good design when you can.
- Check the quality of the database if you have not designed it.
  - Normalization.
  - Guidelines.
- If you do not have any specific reason not to, use constraints in the database: data lasts longer than application programs.

UPPSALA UNIVERSITET

Exercise

Consider the following relation:

CAR_SALE(Car#, Date_sold, Salesman#, Commision%, Discount_amt

Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key.

Additional dependencies are:

Date_sold ->Discount_amt

Salesman# ->commission%

Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?

UPPSALA UNIVERSITET

Answer

CAR_SALE(Car#, Date_sold, Salesman#, Commision%, Discount_amt

Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key.

Additional dependencies are:

Date_sold -> Discount_amt

Salesman# -> Commission%

This relation satisfies 1NF but not 2NF (Car# -> Date_sold and Car# -> Discount_amt so these two attributes are not FFD on the primary key) and not 3NF. To normalize

2NF:

Car_Sale1 (Car#, Date_sold, Discount_amt)

Car_Sale2 (Car#, Salesman#)

Car_Sale3 (Salesman#, Commission%)

3NF:

Car_Sale1-1 (Car#, Date_sold)

Car_Sale1-2 (Date_sold, Discount_amt)

Car_Sale2 (Car#, Salesman#)

Car_Sale3 (Salesman#, Commission%)

BONUS MATERIAL

UPPSALA UNIVERSITET

Multi-valued dependencies

- Consider the relation *classes(course, teacher, book)*

course	*teacher*	*book*
database	Tore	Elmasri
database	Tore	Ullman
database	Martin	Elmasri
database	Martin	Ullman
operating system	Martin	Silberschatz
operating system	Martin	Shaw

- The *book* attribute specifies required literature independent of teacher.
- The relation fulfills BCNF, but still every teacher must be supplied two times.

UPPSALA UNIVERSITET

Lossless join

- We decompose a schema *R* with functional dependencies *D* into schemas *R1* and *R2*
- *R1* and *R2* is called a *lossless-join* decomposition (in relation to *D*) if $R1 \bowtie R2$ contains the same information as *R*.
  - no spurious tuples are generated when we perform the join between the decomposed relations.
  - no information is lost when the relation is decomposed.

UPPSALA UNIVERSITET

How to avoid these problems?

- Split the relation (relation) *SUPP_INFO(SNAME, INAME, SADDR, PRICE)* into:

  SUPPLIERS(SNAME, SADDR)
  SUPPLIES(SNAME, INAME, PRICE)

- Disadvantages:
  - To retrieve addresses for suppliers that provide ***Car 1***, we must perform a **JOIN operation**, *SUPPLIERS * SUPPLIES,* that we could get away with if we instead had the relation:
    SUPP_INFO(SNAME, INAME, SADDR, PRICE)
  - **SELECT** and **PROJECT** that is much cheaper will do the job in the latter case