Università
degli Studi
di Padova

Normalization for the Relational Model

Basi di Dati

Bachelor's Degree in Computer Engineering
Academic Year 2024/2025

Dipartimento
di Ingegneria
dell'Informazione

Ornella Irrera

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

Outline

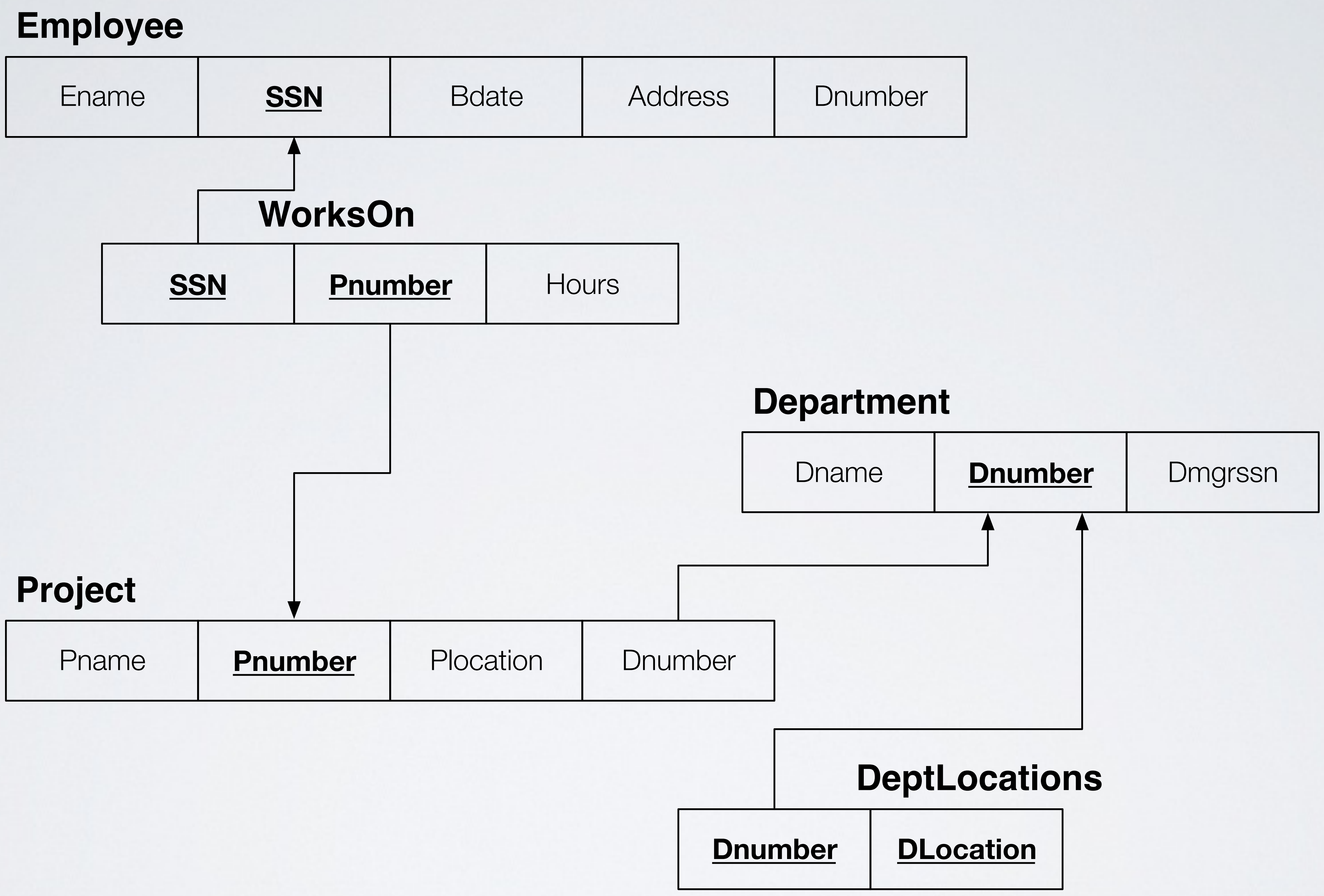
- Quality Measures for Relation Schemas
- Normalization and Functional Dependencies
- Boyce-Codd Normal Form

Informal Design Guidelines for Relational Schemas

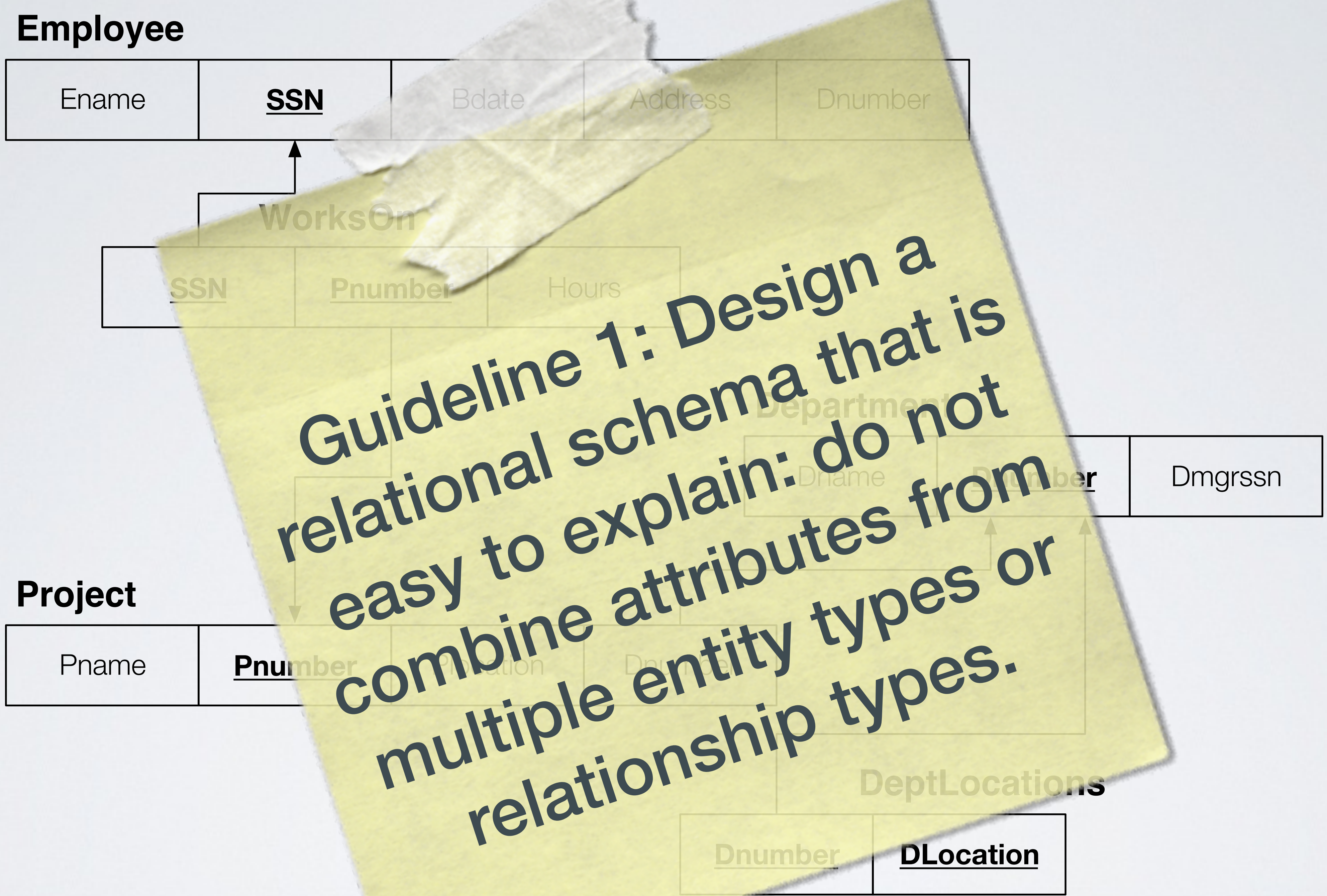
Informal Quality Measures For Relational Schemas

- Semantics of the attributes
- Reducing the redundant values in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

Semantics of the Relational Attributes



Semantics of the Relational Attributes



Guideline 1: Design a relational schema that is easy to explain: do not combine attributes from multiple entity types or relationship types.

Semantics of the Relational Attributes

EmpDept

Ename	**SSN**	Bdate	Address	Dnumber	Dname	DmgrSSN

EmpProj

SSN	**Pnumber**	Hours	Ename	Pname	Plocation

Semantics of the Relational Attributes

EmpDept

Ename	<u>**SSN**</u>	Bdate	Address	Dnumber	Dname	DmgrSSN

EmpProj

<u>**SSN**</u>	<u>**Pnumber**</u>	Hours	Ename	Pname	Plocation

Although there is nothing wrong logically with these two relations, they violate Guideline 1 by mixing attributes from distinct real-world entities, therefore they are considered poor designs.

Redundant Information in Tuples

EmpDept

Ename	**SSN**	Bdate	Address	Dnumber	Dname	DmgrSSN

Redundant Information in Tuples

EmpDept

Ename	**SSN**	Bdate	Address	Dnumber	Dname	DmgrSSN

Problem of update anomalies:

- Insertion Anomalies:
  - to insert a new employee we must include the attribute values for the department correctly and consistently with the values of the other tuples or NULL values if the employee does not work for a department yet
  - It is difficult to insert a new department that has no employees yet

Redundant Information in Tuples

EmpDept

Ename	**SSN**	Bdate	Address	Dnumber	Dname	DmgrSSN

Problem of update anomalies:

- Insertion Anomalies:
  - to insert a new employee we must include the attribute values for the department correctly and consistently with the values of the other tuples or NULL values if the employee does not work for a department yet
  - It is difficult to insert a new department that has no employees yet
- Deletion anomalies: if we delete the employee tuple representing the last employee working for a department we lose the information related to the department

Redundant Information in Tuples

EmpDept

Ename	**SSN**	Bdate	Address	Dnumber	Dname	DmgrSSN

Problem of update anomalies:

- Insertion Anomalies:
  - to insert a new employee we must include the attribute values for the department correctly and consistently with the values of the other tuples or NULL values if the employee does not work for a department yet
  - It is difficult to insert a new department that has no employees yet
- Deletion anomalies: if we delete the employee tuple representing the last employee working for a department we lose the information related to the department
- Modification Anomalies: if we change the values of one of the attributes of a particular department we must update the tuples of all the employees who work in that department

Redundant Information in Tuples

EmpDept

Ename	<u>SSN</u>	Bdate	Address	Dnumber	Dname	DmgrSSN

Problem of update anomalies:

- Insertion Anomalies:
  - to insert a new employee we must include the attribute values for the department correctly and consistently with the values of the other tuples or NULL values if the employee does not work for a department yet
  - It is difficult to insert a new department that has no employees
- Deletion anomalies: if we delete the employee tuple representing the last employee working for a department we lose the information related to the department
- Modification Anomalies: if we change the values of one of the attributes of a particular department we must update the tuples of all the employees who work in that department

Guideline 2: Design the relational schema so that no update anomalies are present. If any anomalies are present, note them and make sure that the programs that update the database operate correctly.

NULL Values in Tuples

If many of the attributes do not apply to all tuples, we end up with many NULL values in those tuples. This will cause the following issues:

- Waste space at storage level
- Since NULL values can have multiple interpretations (Not Applicable, Unknown, Known but Absent), we may lose the different meanings

NULL Values in Tuples

If many of the attributes do not apply to all tuples, we end up with many NULL values in those tuples. This will cause the following issues:

- Waste space at storage level
- Since NULL values can have multiple interpretations (Not Applicable, Unknown, Known but Absent), we may lose the different meanings
- Specify join operations (inner and outer joins produce different results when NULLs are involved)

NULL Values in Tuples

If many of the attributes do not apply to all tuples, we end up with many NULL values in those tuples. This will cause the following issues:

- Waste space at storage level
- Since NULL values can have multiple interpretations (Not Applicable, Unknown, Known but Absent), we may lose the different meanings
- Specify join operations (inner and outer joins produce different results when NULLs are involved)
- Account for them with aggregation operations

Notice that we do not want to avoid NULL values at all, we want to avoid systematic NULL values.

NULL Values in Tuples

If many of the attributes do not apply to all tuples, we end up with many NULL values in those tuples. This will cause the following issues:

- Waste space at storage level
- Since NULL values can have multiple interpretations (Not Applicable, Unknown, Known but Absent), we may lose the different meanings
- Specify join operations (inner and outer joins produce different results when NULLs are involved)
- Account for them with aggregation operations

Notice that we do not want to avoid NULL values at all, we want to avoid systematic NULL values.

> **Guideline 3: Avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure they apply in exceptional cases only.**

Generation of Spurious Tuples: Example

EmpProj

<u>SSN</u>	**<u>Pnumber</u>**	Hours	Ename	Pname	Plocation

SSN	Pnumber	Hours	Pname	Ename	Plocation
123456	1	32.5	Product X	Smith John	Bellaire
123456	2	7.5	Product Y	Smith John	Sugarland
666884	3	40.0	Product Z	Kumar Ramesh	Houston
453453	1	20.0	Product X	English Joyce	Bellaire
453453	2	20.0	Product Y	English Joyce	Sugarland

Generation of Spurious Tuples: Example

EmpProj

SSN	**Pnumber**	Hours	Ename	Pname	Plocation

SSN	Pnumber	Hours	Pname	Ename	Plocation
123456	1	32.5	Product X	Smith John	Bellaire
123456	2	7.5	Product Y	Smith John	Sugarland
666884	3	40.0	Product Z	Kumar Ramesh	Houston
453453	1	20.0	Product X	English Joyce	Bellaire
453453	2	20.0	Product Y	English Joyce	Sugarland

EmpProj1

SSN	**Pnumber**	Hours	Pname	Plocation

SSN	Pnumber	Hours	Pname	Plocation
123456	1	32.5	Product X	Bellaire
123456	2	7.5	Product Y	Sugarland
666884	3	40.0	Product Z	Houston
453453	1	20.0	Product X	Bellaire
453453	2	20.0	Product Y	Sugarland

EmpLocs

Ename	**Plocation**

Ename	Plocation
Smith John	Bellaire
Smith John	Sugarland
Kumar Ramesh	Houston
English Joyce	Bellaire
English Joyce	Sugarland

Generation of Spurious Tuples: Example

SSN	Pnumber	Hours	Pname	Plocation
123456	1	32.5	Product X	Bellaire
123456	2	7.5	Product Y	Sugarland
666884	3	40.0	Product Z	Houston
453453	1	20.0	Product X	Bellaire
453453	2	20.0	Product Y	Sugarland

Ename	Plocation
Smith John	Bellaire
Smith John	Sugarland
Kumar Ramesh	Houston
English Joyce	Bellaire
English Joyce	Sugarland

SSN	Pnumber	Hours	Pname	Plocation	Ename
123456	1	32.5	Product X	Bellaire	Smith John
123456	1	32.5	Product X	Bellaire	English Joyce
123456	2	7.5	Product Y	Sugarland	Smith John
123456	2	7.5	Product Y	Sugarland	English Joyce
666884	3	40.0	Product Z	Houston	Kumar Ramesh
453453	1	20.0	Product X	Bellaire	Smith John
453453	1	20.0	Product X	Bellaire	English Joyce
453453	2	20.0	Product Y	Sugarland	Smith John
…	…	…	…	…	…

Generation of Spurious Tuples: Example

SSN	Pnumber	Hours	Pname	Plocation
123456	1	32.5	Product X	Bellaire
123456	2	7.5	Product Y	Sugarland
666884	3	40.0	Product Z	Houston
453453	1	20.0	Product X	Bellaire
453453	2	20.0	Product Y	Sugarland

Ename	Plocation
Smith John	Bellaire
Smith John	Sugarland
Kumar Ramesh	Houston
English Joyce	Bellaire
English Joyce	Sugarland

SSN	Pnumber	Hours	Pname	Plocation	Ename
123456	1	32.5	Product X	Bellaire	Smith John
123456	1	32.5	Product X	Bellaire	English Joyce
123456	2	7.5	Product Y	Sugarland	Smith John
123456	2	7.5	Product Y	Sugarland	English Joyce
666884	3	40.0	Product Z	Houston	Kumar Ramesh
453453	1	20.0	Product X	Bellaire	Smith John
453453	1	20.0	Product X	Bellaire	English Joyce
453453	2	20.0	Product Y	Sugarland	Smith John
…	…	…	…	…	…

Generation of Spurious Tuples: Example

SSN	Pnumber	Hours	Pname	Plocation
123456	1	32.5	Product X	Bellaire
123456	2	7.5	Product Y	Sugarland
666884	3	40.0	Product Z	Houston
453453	1	20.0	Product X	Bellaire
453453	2	20.0	Product Y	Sugarland

Ename	Plocation
Smith John	Bellaire
Smith John	Sugarland
Kumar Ramesh	Houston
English Joyce	Bellaire
English Joyce	Sugarland

SSN	Pnumber	Hours	Pname	Plocation	Ename
123456	1	32.5	Product X	Bellaire	Smith John
123456	1	32.5	Product X	Bellaire	English Joyce
123456	2	7.5	Product Y	Sugarland	Smith John
123456	2	7.5	Product Y	Sugarland	English Joyce
666884	3	40.0	Product Z	Houston	Kumar Ramesh
453453	1	20.0	Product X	Bellaire	Smith John
453453	1	20.0	Product X	Bellaire	English Joyce
453453	2	20.0	Product Y	Sugarland	Smith John
…	…	…	…	…	…

Generation of Spurious Tuples

EmpProj

SSN	**Pnumber**	Hours	Ename	Pname	Plocation

EmpProj1

SSN	**Pnumber**	Hours	Pname	Plocation

EmpLocs

Ename	**Plocation**

- Spurius tuples represent spurious or wrong information that is not valid. In this case they are represented by the additional tuples obtained by joining EmpProj1 and EmpLocs, that were not in Empproj.
- Decomposing EmpProj into EmpLocs and EmpProj1 is undesirable because, when we join them back we do not get the correct original information. This is because in this case Plocation is the attribute that relates EmpLocs and EmpProj1, and Plocation is neither a primary key nor a foreign key.

Generation of Spurious Tuples

EmpProj

<u>SSN</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation

EmpProj1

<u>SSN</u>	<u>Pnumber</u>	Hours	Pname	Plocation

EmpLocs

<u>Ename</u>	<u>Plocation</u>

> **Guideline 4: Design relations so that they can be joined with equality conditions that are either primary or foreign keys in a way that guarantees that no spurious tuples are generated.**

- Spurius tuples represent spurious or wrong information that is not valid. In this case they are represented by the additional tuples obtained by joining EmpProj1 and EmpLocs, that were not in EmpProj.
- Decomposing EmpProj into EmpLocs and EmpProj1 is undesirable because, when we join them back we do not get the correct original information. This is because in this case Plocation is the attribute that relates EmpLocs and EmpProj1, and Plocation is neither a primary key nor a foreign key.

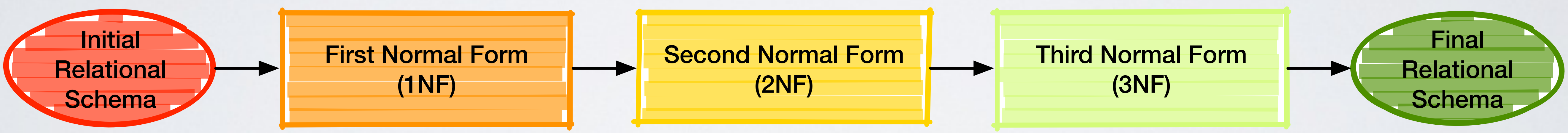
Normalization and Functional Dependencies

Normalization

Process of analyzing the relational schema based on its functional dependencies and primary keys to **minimize** the **redundancy**, the **update** anomalies and **spurious** data.

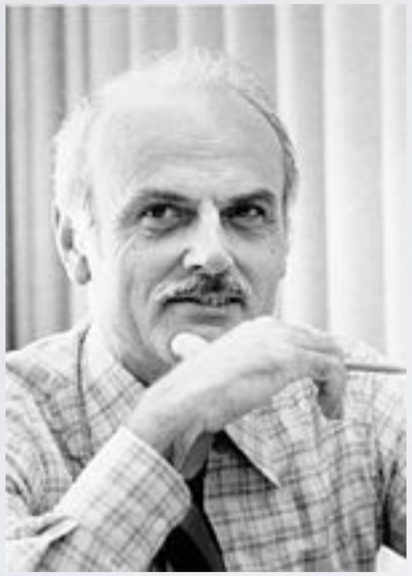
Normalization

Process of analyzing the relational schema based on its functional dependencies and primary keys to minimize the redundancy, the update anomalies and spurious data.



- The normalization process takes a relation schema through a series of tests to certify whether it satisfies a certain normal form.
- Database designers do not need to normalize to the highest possible normal form. Relations may be left in lower normalization status (for example 2NF) for performance reasons.

A Bit of History



Edgar Frank Codd

Proposed the relational model in 1969

Peter Pin-Shan Chen

Proposed the entity relationship model in 1976

Considerations

Normalization is highly used to check the quality of relational schemas when they are designed without the entity-relationship schema. When the entity-relationship schema is transformed and then mapped into the relational schema, in most of the cases the resulting schema does not need to be normalized.

Recall Superkey and Key

Let R be a relational schema and $\mathcal{S}$ a subset of attributes.
$\mathcal{S}$ is a **superkey** if for a relation instance r of R it holds:

$$\forall\, t_1, t_2 \in r \quad t_1[\mathcal{S}] \neq t_2[\mathcal{S}]$$

A **key** K is a superkey with the additional property that the removal of any attribute from K will cause K not to be a superkey anymore

A relation may have more than one key, each is called candidate key. One of the candidate key is designated to be the primary key and the others are called secondary keys.

Prime Attribute

An attribute of the relational schema R is called a prime attribute of R if it is a member of some candidate key of R.

An attribute is called nonprime if it is not a prime attribute, i.e. if it is not a member of any candidate key.

First Normal Form

First Normal Form (1NF): the domain of an attribute must include only atomic (simple, indivisible) values and the value of any attribute in a tuple must be a single value.

- Historically it was defined to disallow multivalued attributes, composite attributes, and their combination
- The removal of composite and multivalued attributes during the transformation of the ER schema guarantees the 1NF

Multivalued Attributes and Nested Relations

Dname	Dnumber	DmgrSSN	Dlocation
Research	5	333445	{Bellaire, Sugarland, Houston}
Administration	4	987987	{Stafford}
Headquarters	1	888665	{Houston}

Multivalued attributes:
more columns inside one cell

Nested Relations:
more rows inside one cell

<table>
<tr><th rowspan="2">SSN</th><th rowspan="2">Ename</th><th colspan="2">Projs</th></tr>
<tr><th>Pnumber</th><th>Hours</th></tr>
<tr><td>123456</td><td>Smith John</td><td>1
2</td><td>32.5
7.5</td></tr>
<tr><td>666884</td><td>Kumar Ramesh</td><td>3</td><td>40.0</td></tr>
<tr><td>453453</td><td>English Joyce</td><td>1
2</td><td>20.0
20.0</td></tr>
</table>

First Normal Form: Example of Multivalued Attribute

Department

Dname	<u>**Dnumber**</u>	Dmgrssn	Dlocations

Since each department can have a number of locations, this relation is not in 1NF.

Strategies to achieve 1NF:

1. Remove the attribute Locations and place it in a separate relation DeptLocations with Dnumber (primary key of department)
2. Expand the primary key to {Dnumber, Dlocations}, this solution will introduce redundancy
3. If a maximum number of values is known for the attribute Dlocations, replace it by the corresponding number of atomic attributes Dlocation1, Dlocation2,… This solution will introduce NULL values and querying on this attribute becomes more difficult

Achieving the First Normal Form

Approach 1

Department

Dname	**Dnumber**	Dmgrssn

DeptLocations

Dnumber	**Dlocation**

Approach 2

Department

Dname	**Dnumber**	Dmgrssn	**Dlocations**

Approach 3

Department

Dname	**Dnumber**	Dmgrssn	Dlocation1	Dlocation2	Dlocation3

Achieving the First Normal Form

Approach 1

Department

Dname	**Dnumber**	Dmgrssn

DeptLocations

Dnumber	**Dlocation**

Approach 2

Department

Dname	**Dnumber**	Dmgrssn	**Dlocations**

Approach 3

Department

Dname	**Dnumber**	[illegible]	[illegible]	[illegible]	Dlocation3

First Normal Form: Example of Nested Relation

SSN	Ename	Projs	
		Pnumber	Hours
123456	Smith John	1 2	32.5 7.5
666884	Kumar Ramesh	3	40.0
453453	English Joyce	1 2	20.0 20.0

- Nested relations: each tuples can have a relation within it
- To normalize this in 1NF, we move the nested relation attributes into a new relation and propagate the primary key into it
- This procedure can be applied recursively to a relation with multiple level nesting

Achieving the First Normal Form

EmpProj

SSN	Ename	Projs	
		Pnumber	Hours

EmpProj1

SSN	Ename

EmpProj2

SSN	**Pnumber**	Hours

Functional Dependency

Given a relation R and two sets of attributes $\mathcal{X}$ and $\mathcal{Y}$, a functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ specifies a constraint on the possible tuples that can form an instance r of a relation R. The constraint is that:

$$\forall\, t_1, t_2 \in r : t_1[\mathcal{X}] = t_2[\mathcal{X}] \implies t_1[\mathcal{Y}] = t_2[\mathcal{Y}]$$

Notice that:

- Functional dependencies are a property of the relation schema
- If $\mathcal{X}$ is a candidate key for R, then for any subset of attributes $\mathcal{Y}$ of R, $\mathcal{X} \rightarrow \mathcal{Y}$
- If $\mathcal{X} \rightarrow \mathcal{Y}$, this does not imply that $\mathcal{Y} \rightarrow \mathcal{X}$
- We denote functional dependencies with FD

Functional Dependency: Example

EmpProj

<u>**SSN**</u>	<u>**Pnumber**</u>	Hours	Ename	Pname	Plocation

The following functional dependencies should hold:

- SSN ⟶ Ename
- Pnumber ⟶ {Pname, Plocation}
- {SSN, Pnumber} ⟶ Hours

Inference Rules for Functional Dependecies

Typically the schema designer specifies the functional dependencies that are semantically obvious, however other functional dependencies hold and it is impossible to specify all of them

Inference Rules:

1. Reflexive Rule: $\mathcal{X} \supseteq \mathcal{Y} \implies \mathcal{X} \rightarrow \mathcal{Y}$
2. Augmentation Rule: $\mathcal{X} \rightarrow \mathcal{Y} \implies \{\mathcal{X}, \mathcal{Z}\} \rightarrow \{\mathcal{Y}, \mathcal{Z}\}$
3. Transitive Rule: $\mathcal{X} \rightarrow \mathcal{Y}$ and $\mathcal{Y} \rightarrow \mathcal{Z} \implies \mathcal{X} \rightarrow \mathcal{Z}$
4. Decomposition or Projective Rule: $\mathcal{X} \rightarrow \{\mathcal{Y}, \mathcal{Z}\} \implies \mathcal{X} \rightarrow \mathcal{Y}$
5. Union or additive Rule: $\mathcal{X} \rightarrow \mathcal{Y}$ and $\mathcal{X} \rightarrow \mathcal{Z} \implies \mathcal{X} \rightarrow \{\mathcal{Y}, \mathcal{Z}\}$
6. Pseudotransitive Rule: $\mathcal{X} \rightarrow \mathcal{Y}$ and $\{\mathcal{W}, \mathcal{Y}\} \rightarrow \mathcal{Z} \implies \{\mathcal{W}, \mathcal{X}\} \rightarrow \mathcal{Z}$

Full Functional Dependency

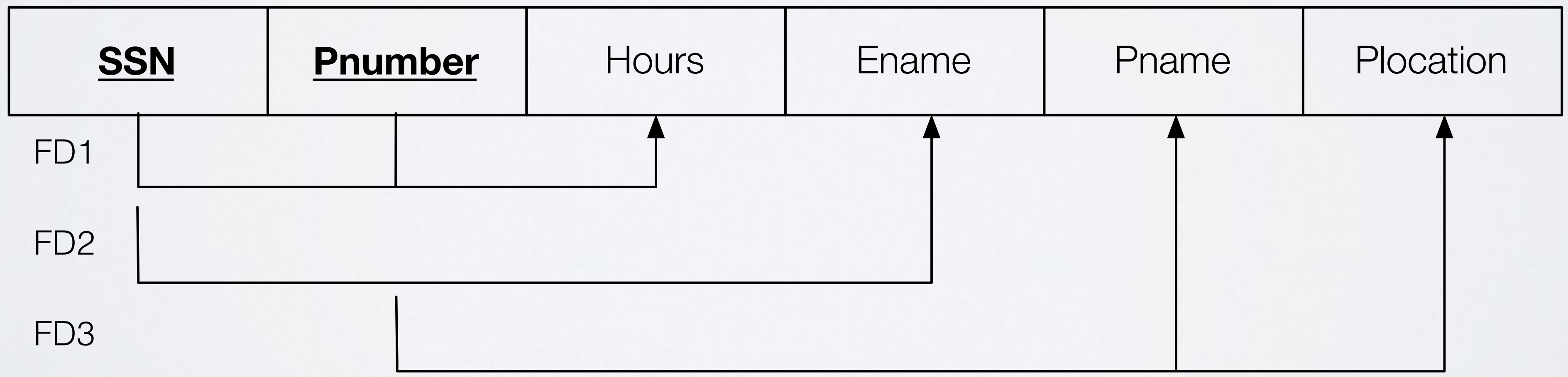
A functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ is a **full functional dependency**, if the removal of any attribute A from $\mathcal{X}$ means that the dependency does not hold any more: $\forall\ A \in \mathcal{X},\ \ (\mathcal{X} - A) \nrightarrow \mathcal{Y}$

A functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ is a partial dependency if some attributes $A \in \mathcal{X}$ can be removed from X and the dependency still holds: $\exists\ A \in \mathcal{X} : (\mathcal{X} - A) \rightarrow \mathcal{Y}$

Full Functional Dependency: Example

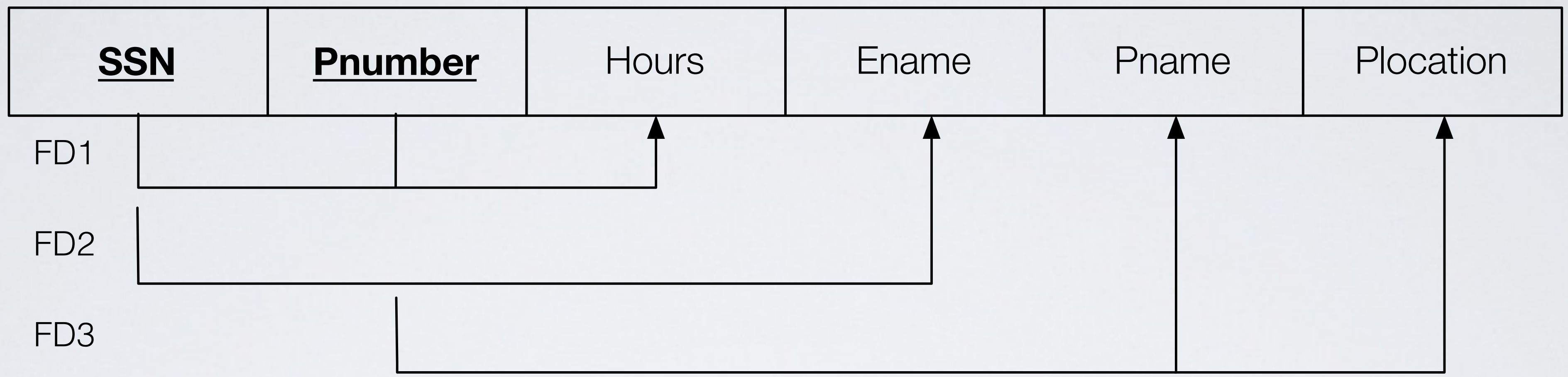
SSN	Pnumber	Hours	Pname	Ename	Plocation
123456	1	32.5	Product X	Smith John	Bellaire
123456	2	7.5	Product Y	Smith John	Sugarland
666884	3	40.0	Product Z	Kumar Ramesh	Houston
453453	1	20.0	Product X	English Joyce	Bellaire
453453	2	20.0	Product Y	English Joyce	Sugarland

EmpProj

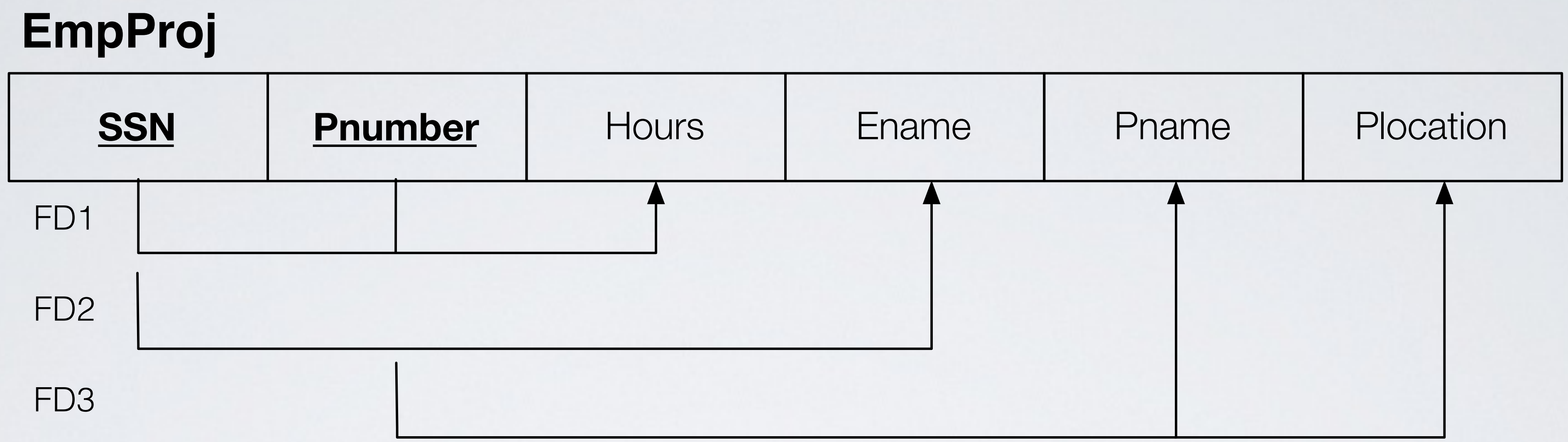


Full Functional Dependency: Example

EmpProj



Full Functional Dependency: Example



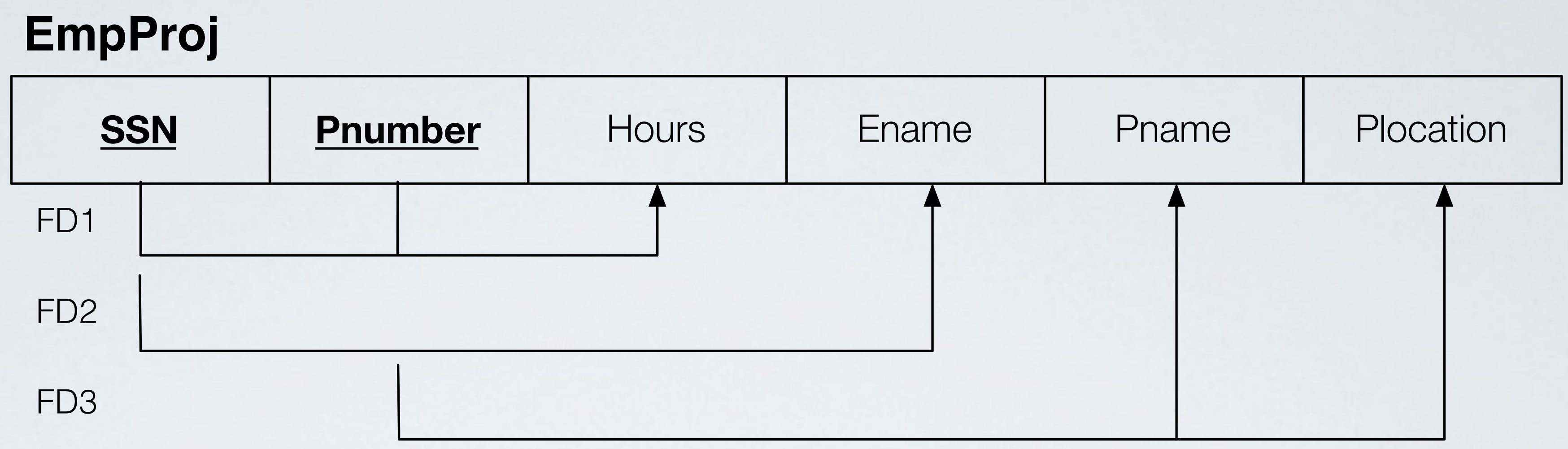
- {SSN, Pnumber} ⟶ Hours is a full dependency
- {SSN, Pnumber} ⟶ Ename is a partial dependency because SSN ⟶ Ename holds

Second Normal Form

Second Normal Form (2NF): every non prime attribute A in R is fully functional dependent on every key of R

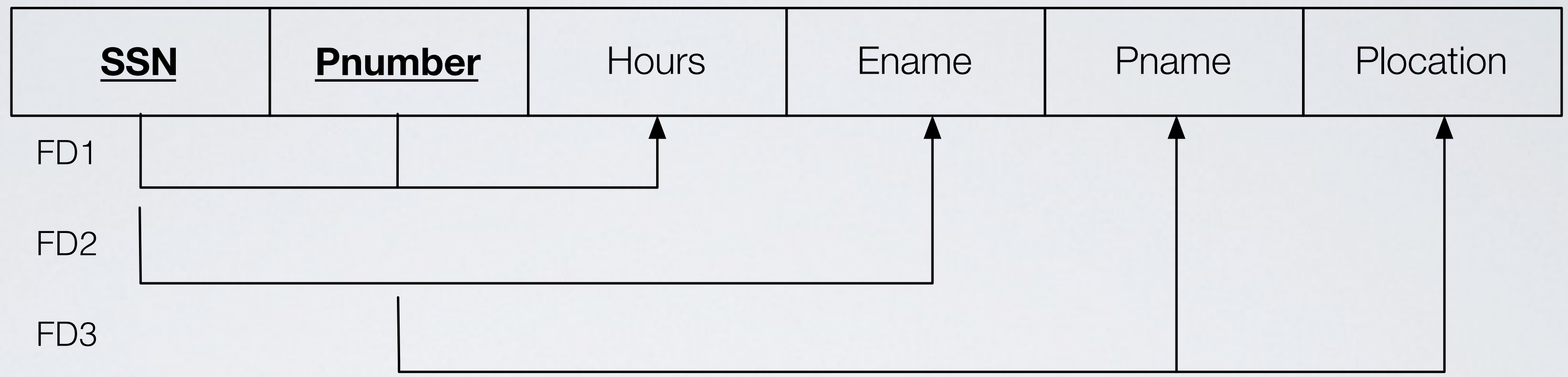
- Equivalent to: every non prime attribute A in R is not partially dependent on any key of R
- If each key contains a single attribute, then the 2NF is guaranteed

Second Normal Form: Example



Second Normal Form: Example

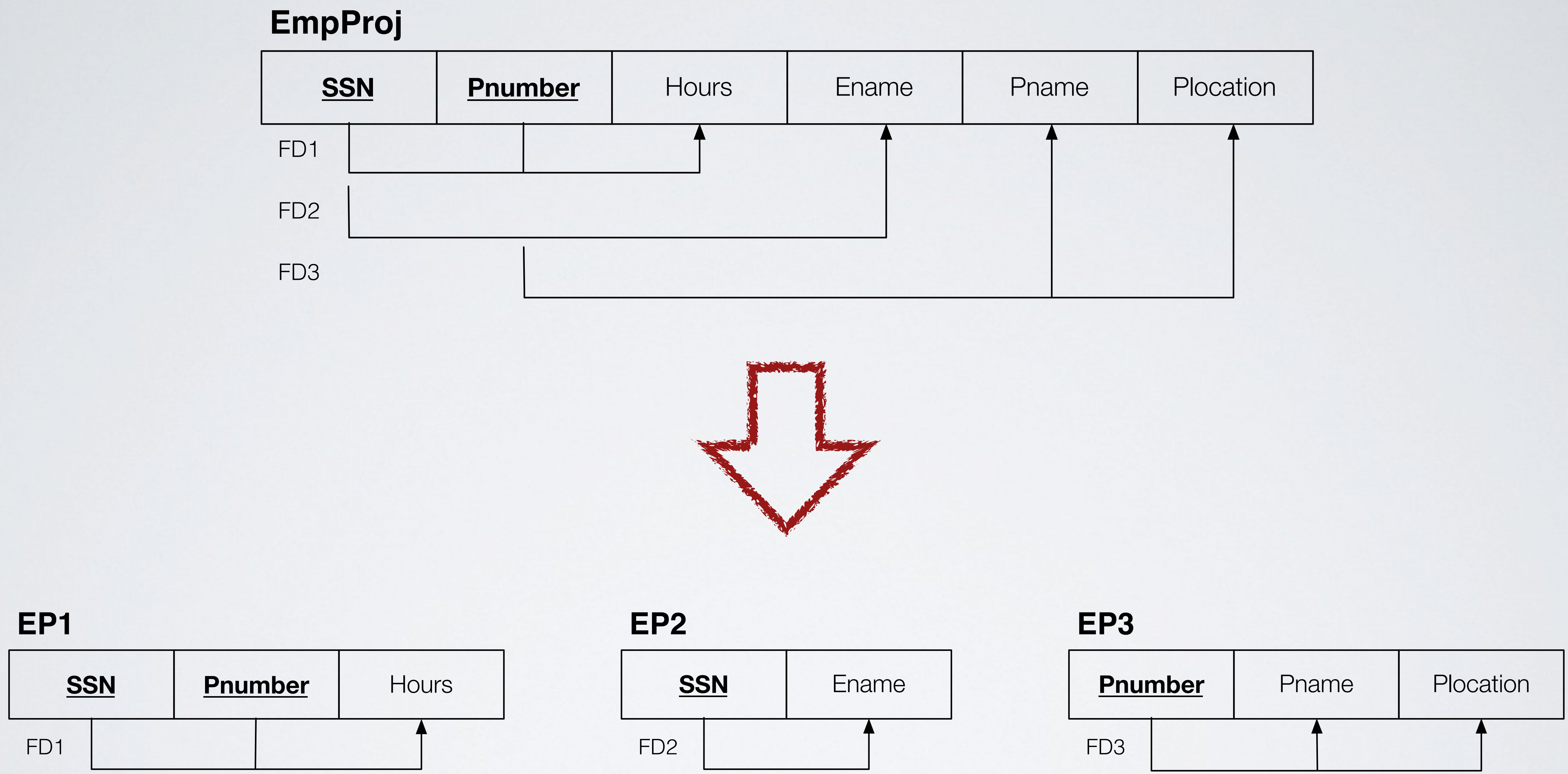
EmpProj



The EmpProj relation is in 1NF but not in 2NF:

- Ename is partially dependent on the primary key {SSN, Pnumber} through FD2
- {Pname, Plocation} is partially dependent on the primary key {SSN, Pnumber} through FD3

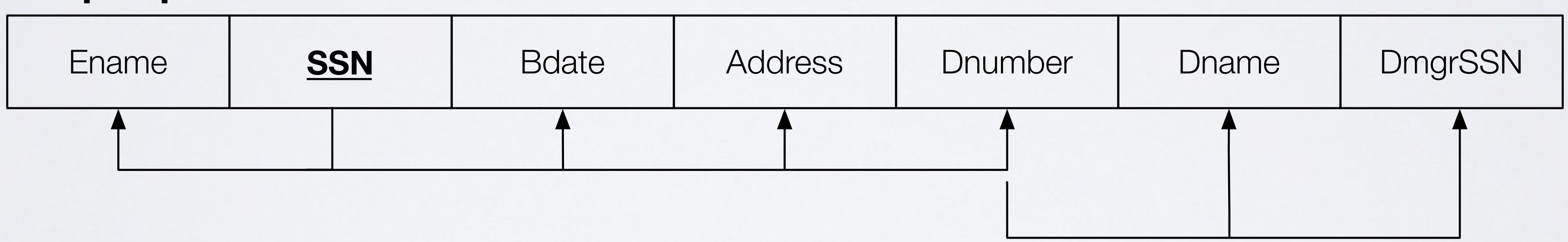
Achieving the Second Normal Form



Transitive Dependency

A functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ in a relational schema R is a **transitive dependency** if there is a set of attributes $\mathcal{Z}$ that is neither a candidate key nor a subset of any key of R, and both $\mathcal{X} \rightarrow \mathcal{Z}$ and $\mathcal{Z} \rightarrow \mathcal{Y}$ hold.

EmpDept



Since both the dependencies SSN ⟶ Dnumber and Dnumber ⟶ DmgrSSN hold, SSN ⟶ DmgrSSN is transitive through Dnumber

Third Normal Form

Third Normal Form (3NF): a relational schema R is in 3NF if it satisfies 2NF and no non prime attribute of R is transitively dependent on any key of R

- Intuitively any functional dependency in which the left hand side is part of any key or is a non key attribute is a problematic FD

Trivial and Nontrivial Functional Dependency

A functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ is **trivial** if $\mathcal{X} \supseteq \mathcal{Y}$, otherwise it is **nontrivial**

This definition is a consequence of the reflexive rule, which states that a set of attributes always determines itself or any of its subsets.

3NF Alternative Definition

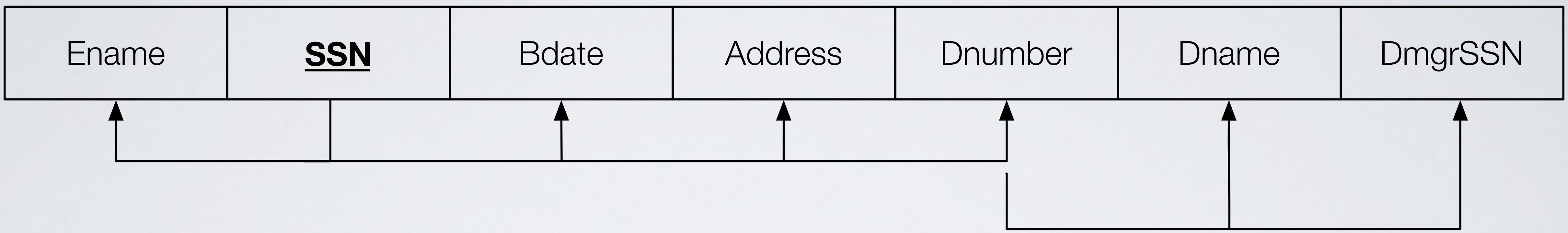
Third Normal Form (3NF): a relational schema R is in 3NF if whenever a nontrivial functional dependency $\mathcal{X} \rightarrow A$ holds in R, either:

A. $\mathcal{X}$ is a superkey of R

B. A is a prime attribute of R

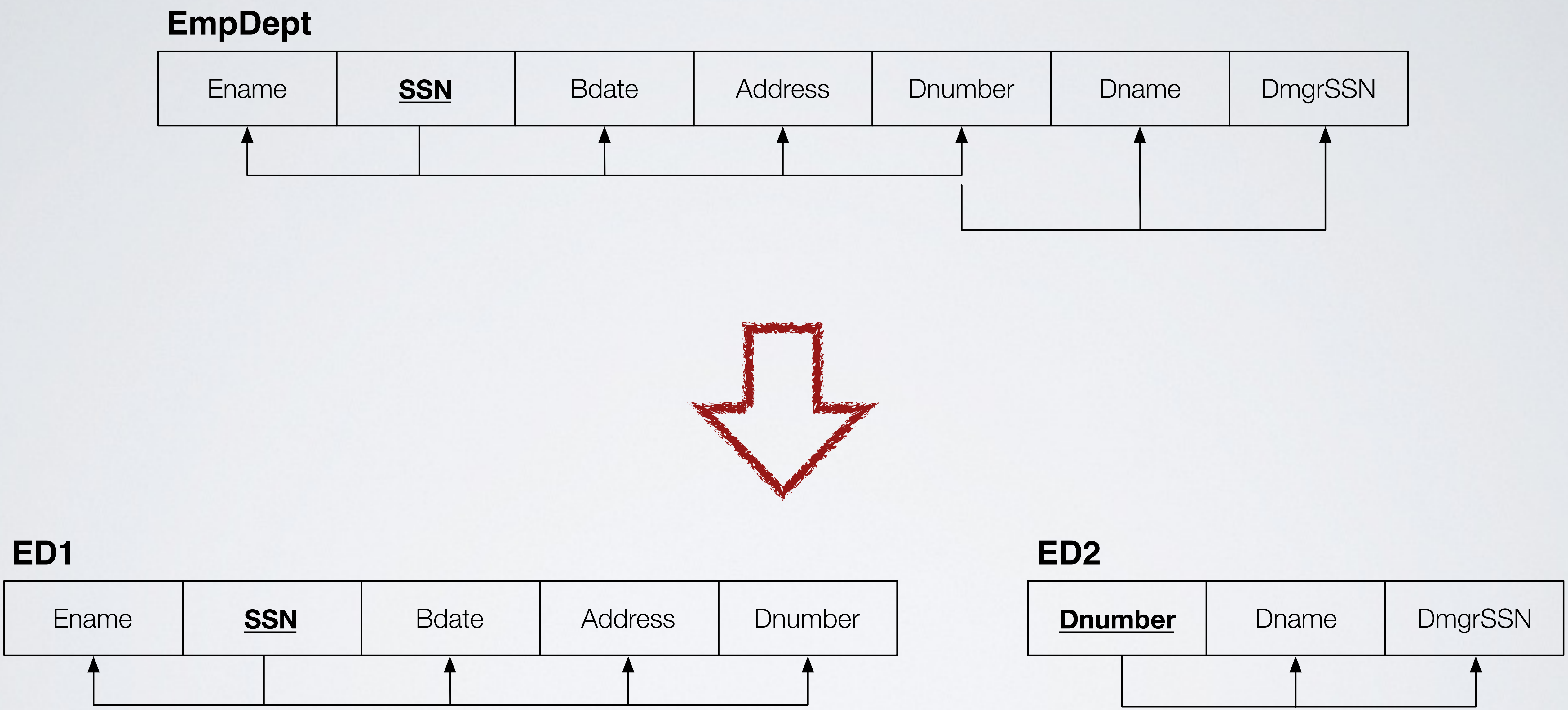
Third Normal Form: Example

EmpDept



This relation is in 2NF, however it is not in 3NF. Indeed the dependency SSN ⟶ DmgrSSN is transitive through Dnumber, and Dnumber is neither a key itself nor a subset of a key.

Achieving the Third Normal Form



Summary of Normal Forms

NF	Definition	Normalization
1NF	Relation should have no nonatomic attributes or nested relations.	Form new relations for each nonatomic attribute or nested relation.
2NF	For relations where keys contains multiple attributes, no nonkey attribute should be functionally dependent on a part of any key.	Decompose and set up a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
3NF	There should be no transitive dependency of a nonkey attribute on any key.	Decompose and set up a relation that includes the nonkey attributes that functionally determine other nonkey attributes.

Boyce-Codd Normal Form

Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF): a relational schema R is in BCNF whenever a nontrivial functional dependency $\mathcal{X} \rightarrow A$ holds in R, then $\mathcal{X}$ is a superkey of R

Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF): a relational schema R is in BCNF whenever a nontrivial functional dependency $\mathcal{X} \to A$ holds in R, then $\mathcal{X}$ is a superkey of R

Notice:

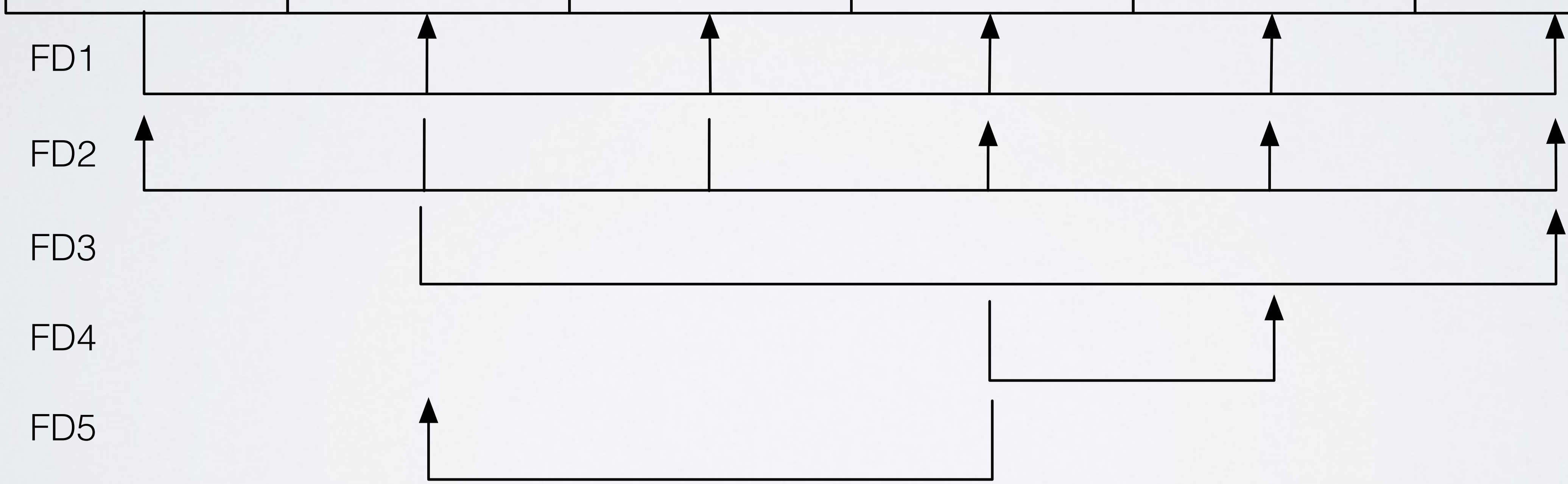
- Difference between 3NF and BCNF: condition B is missing
- BCNF is stricter than 3NF, every relation in BCNF is in 3NF but 3NF $\not\Longrightarrow$ BCNF

Normalization Exercise

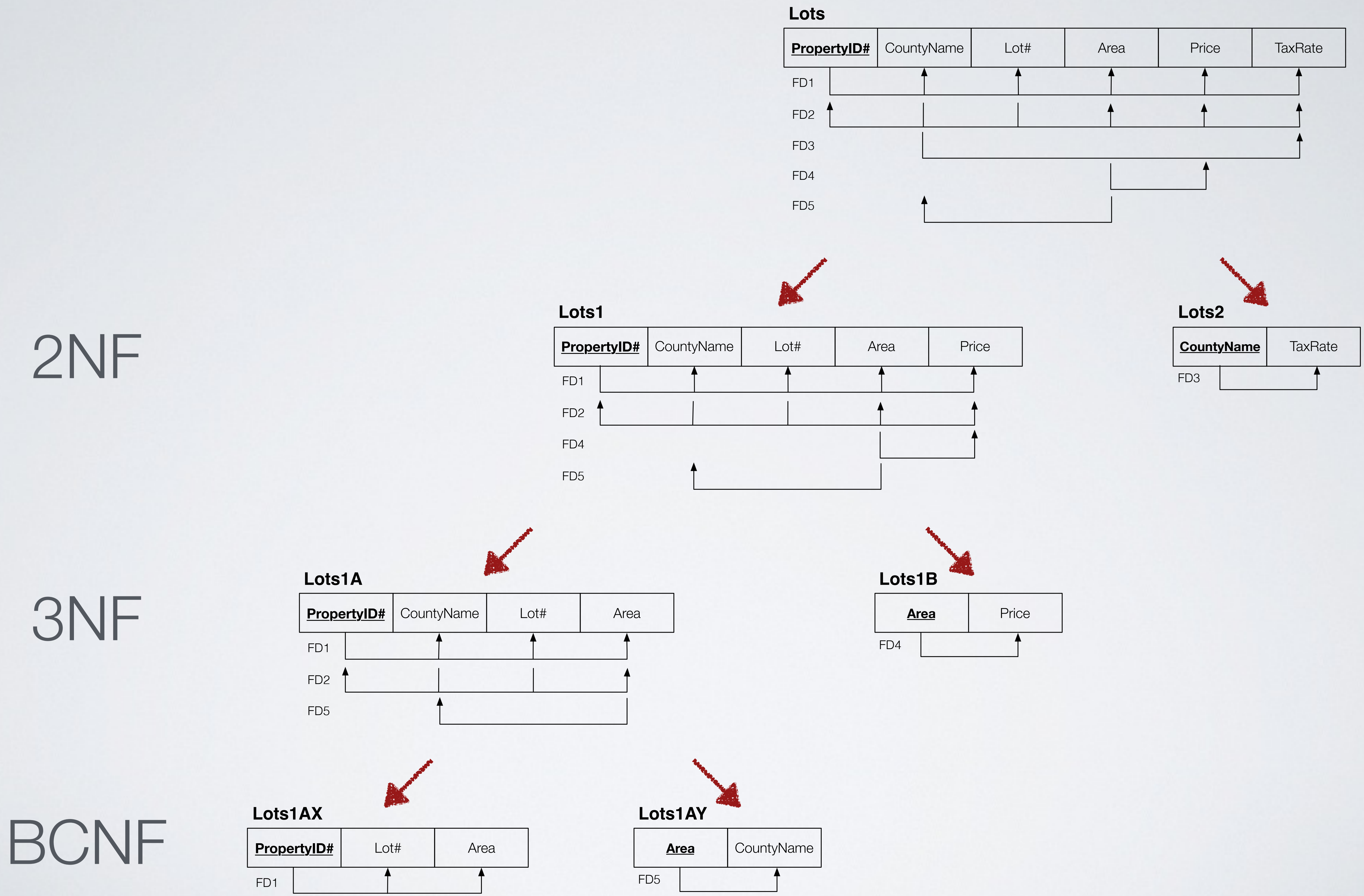
Normalize the following relation up to BCNF

Lots

PropertyID#	CountyName	Lot#	Area	Price	TaxRate



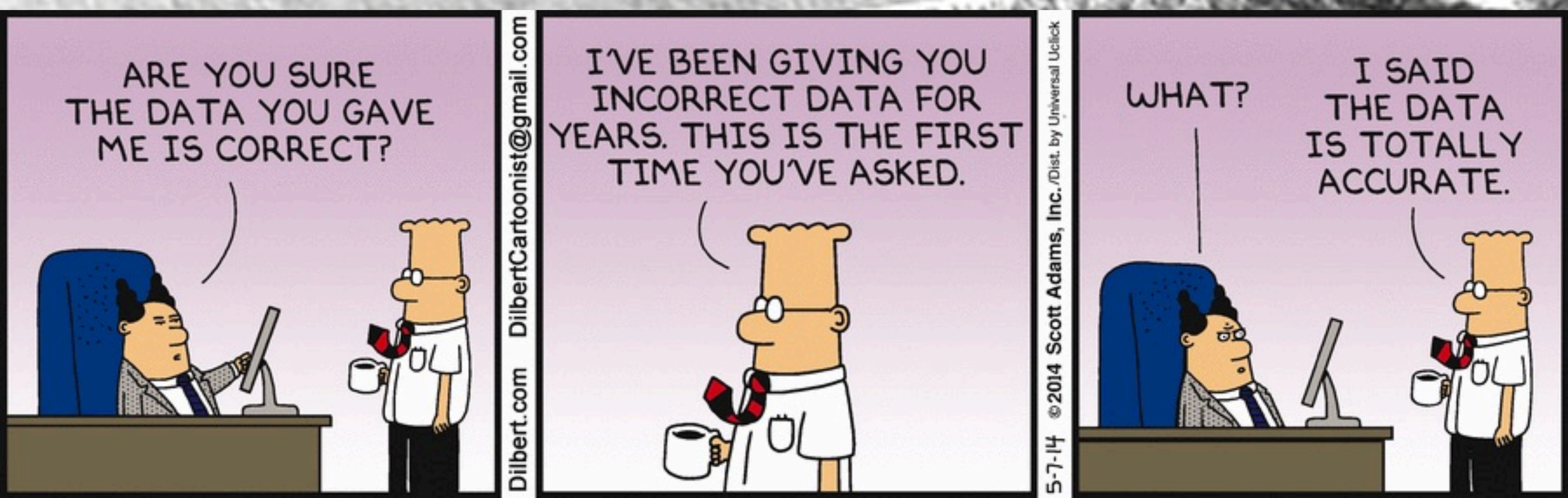
Normalization Exercise: Solution



Reference

Elmasri, R., and Navathe, S. B. (2004). Fundamental of Database Systems, 4-th edition. Pearson Addison Wesley, Boston (MA), USA, pages 293–331.


Questions?


ARE YOU SURE THE DATA YOU GAVE ME IS CORRECT?
Dilbert.com DilbertCartoonist@gmail.com
I'VE BEEN GIVING YOU INCORRECT DATA FOR YEARS. THIS IS THE FIRST TIME YOU'VE ASKED.
5-7-14 ©2014 Scott Adams, Inc./Dist. by Universal Uclick
WHAT?
I SAID THE DATA IS TOTALLY ACCURATE.