Information Management and Data Engineering

CS4D2a – 4CSLL1 – CS3041
Design Optimisation
Functional Dependencies and Normalisation

Séamus Lawless
seamus.lawless@scss.tcd.ie

knowledge and data engineering group

Database Optimisation

- Need of a formal method for analysing how the relations and attributes are grouped
- A measure of appropriateness or goodness, other than the intuition of the designer
  - To assess the quality of the design
- Measures
  - Design guidelines
  - Functional Dependencies
  - Normalisation

Functional Dependencies

- Formal tool for analysis of relational schemas
  - Enables the designer to detect and describe the problems described in the previous lecture in more precise terms
- One of the most important concepts in relational schema design theory
- Main tool for measuring the appropriateness of groupings of attributes into relations

Definition

- A *functional dependency* is a constraint between two sets of attributes
- Suppose our relational database schema has n attributes
  - $A_1, A_2, \dots, A_n$
- Think of the whole database as being described by a single universal relation
  - $R = \{A_1, A_2, \dots, A_n\}$

Definition

- A functional dependency, denoted by $X \rightarrow Y$, specifies a constraint on the possible tuples that can form a relation state r of R
  - between two sets of attributes X and Y
  - X and Y are subsets of the relation R
- The constraint is
  - for any two tuples t_1 and t_2 in $r(R)$ that have $t_1[X] = t_2[X]$
  - they must also have $t_1[Y] = t_2[Y]$

Definition

- The values of the attribute set X from a tuple in r, uniquely (or *functionally*) determine the values of the attribute set Y
  - We can say that:
    - There is a *functional dependency* from X to Y

      or
    - Y is *functionally dependent* on X

Definition

- The abbreviation for functional dependency is FD or f.d.
  - The set of attributes X is called the left-hand side of the FD, Y is called the right-hand side
- Thus
  - X functionally determines Y in a relation schema R if, and only if, whenever two tuples agree on their X values, they must necessarily agree on their Y values

Things to Note

- If X is a candidate key of R, then
  - X➔Y for any subset of attributes Y of R
  - Thus, X➔R
  - In other words, if X has to be unique for every instance of R, then X uniquely determines all the other attribute values of R
- If X➔Y in R, this does not necessarily imply that Y➔X in R
  - Not commutative

Identification of FDs

- A *functional dependency* is a property of the semantics or meaning of the attributes
  - A database designer will use their understanding of the semantics of the attributes of R to specify the functional dependencies that must hold on all instances of R
    - Entity relationship modeling supports the development of this understanding

Example

- Consider again

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

- using the semantics of the attributes and relation, the following FDs should hold:
  - Ssn ➔ Ename
  - Pnumber ➔ {Pname, Plocation}
  - {Ssn, Pnumber} ➔ Hours

Disproving a FD

- You cannot use a single set of data – r(R) – to prove a FD, but you can use it to disprove a FD

TEACH

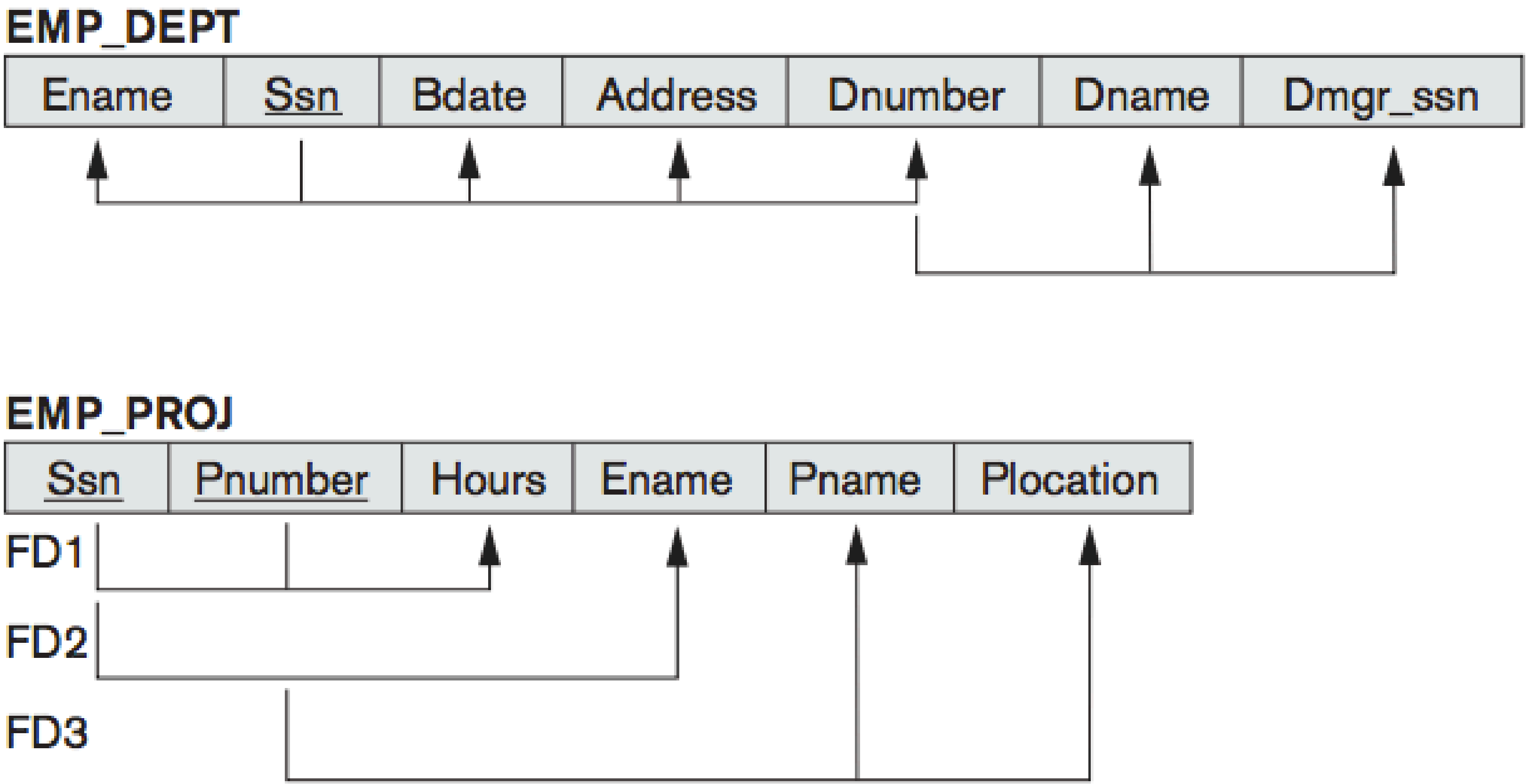
Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Bartram

- Can't prove Text → Course
- Can disprove Teacher → Course

Constraints

- Whenever the semantics of two sets of attributes of R indicate that a FD should hold, the dependency is specified as a *constraint*
- Hence, FDs are used to further enhance a relation schema R, by specifying constraints that must hold *at all times*

Diagrammatic Notation



Normalisation

- The normalisation process takes a relation schema through a series of tests to certify whether it satisfies a certain *normal form*
- There are a number of normal forms:
  - First Normal Form – 1NF
  - Second Normal Form – 2NF
  - Third Normal Form – 3NF
  - Boyce-Codd Normal Form – BCNF

Normalisation

- Evaluate each relation against the criteria for normal forms
  - Decompose relations where necessary
- Can be considered *relational design by analysis*
  - ER Modeling
  - Mapping to Relational Schema
  - Functional Dependencies
  - Normalisation

Normalisation

- The process of analysing relation schemas based upon their primary keys and functional dependencies in order to:
  - minimize redundancy
  - minimize insertion, deletion and modification anomalies
- Relations which do not pass the normal form tests are decomposed into smaller relation schemas

Normalisation

- Normalisation through decompositon must confirm two properties in the resulting database design
  – Non-Additive or Lossless Join Property
    - This guarantees that spurious tuple generation does not occur
  – Dependency Preservation Property
    - This ensures that each functional dependency is represented in an individual relation

Normalisation

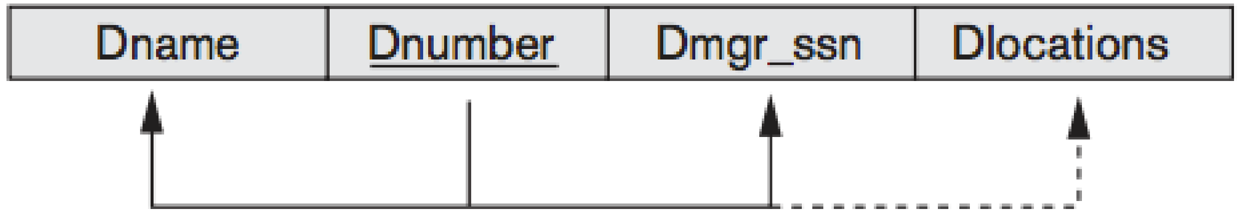
- Provides database designers with:
  - a formal framework for analysing relations based upon their primary keys and functional dependencies
  - a set of normal form tests that can be carried out on individual relation schemas so that the relational database can be normalised to the desired degree

First Normal Form

- In 1NF all attribute values must be atomic
  - The word atom comes from the Latin atomis, meaning indivisible (or literally "not to cut")
- 1NF dictates that at every row-column intersection, there exists only one value, not a list of values
- The benefits from this rule should be fairly obvious.
  - If lists of values are stored in a single column, there is no simple way to manipulate those values.

1NF

DEPARTMENT



DEPARTMENT

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

Achieving 1NF

DEPARTMENT

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

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

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

1NF

- 1NF also disallows multi-valued attributes that are themselves composite

EMP_PROJ

Ssn	Ename	Pnumber	Hours

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0

Achieving 1NF

- Follow the same steps as with a single multi-valued attribute

EMP_PROJ

Ssn	Ename	Pnumber	Hours

EMP

Ssn	Ename

EMP_PROJ

Ssn	Pnumber	Hours

Achieving 1NF

- Remove the attribute(s) that violates 1NF and put it into a separate, new relation
- Add the primary key of the original relation to the new relation
  - This will serve as a foreign key
- The primary key of the new relation is a composite primary key
- This *decomposes* a non-1NF relation into two 1NF relations

1NF Example

- A database stores information about people
  - People can own more than one car
  - People can have more than one phone number

Person

<u>PPS_num</u>	Name	Car_Reg_No	Car_Model	Car_CC	Phone_Num

Person

<u>PPS_num</u>	Name

Car

<u>Owner_PPS</u>	<u>Car_Reg_No</u>	Car_Model	Car_CC

Phone

<u>Owner_PPS</u>	<u>Phone_Num</u>

Second Normal Form

- A table is said to be in Second Normal Form (2NF) if:
  - it is 1NF compliant
  - every non-key column is *fully functionally dependent* on the entire primary key.
- In other words:
  - tables should only store data relating to one "thing" (or entity)
  - that entity should be described by its primary key.

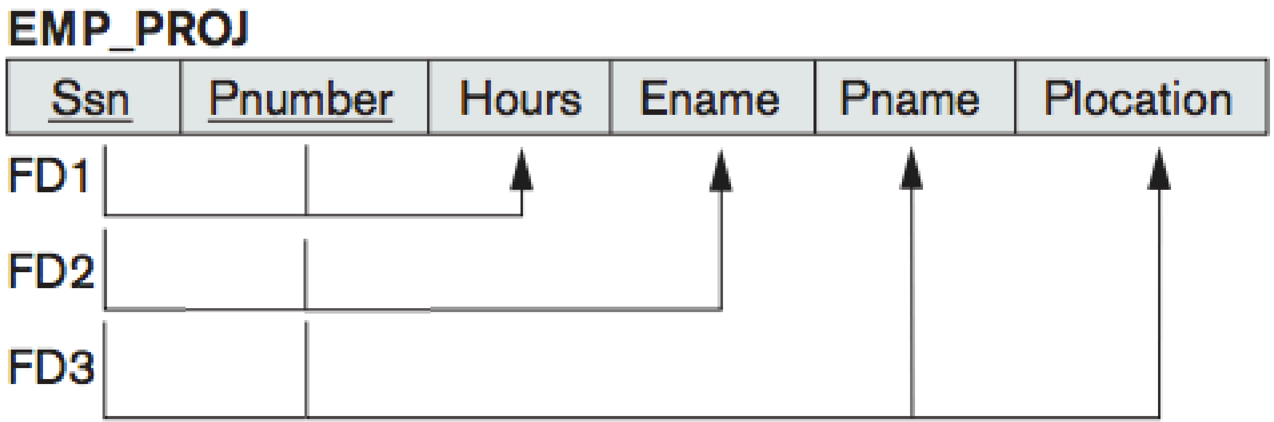
Full Functional Dependency

- A FD $X \rightarrow Y$ is said to be a *full functional dependency* if the removal of any single attribute from the set of attributes X means that the dependency no longer holds
  - Think of X as a composite primary key
  - All the other attributes must be dependent upon the full key, not just part of it
- for any attribute $A \varepsilon X$
  - $(X - \{A\})$ does not functionally determine Y

Partial Functional Dependency

- A FD $X \rightarrow Y$ is said to be a *partial functional dependency* if a single attribute can be removed from the set of attributes X yet the dependency still holds
- for any attribute A ε X
  - $(X - \{A\}) \rightarrow Y$

2NF Example

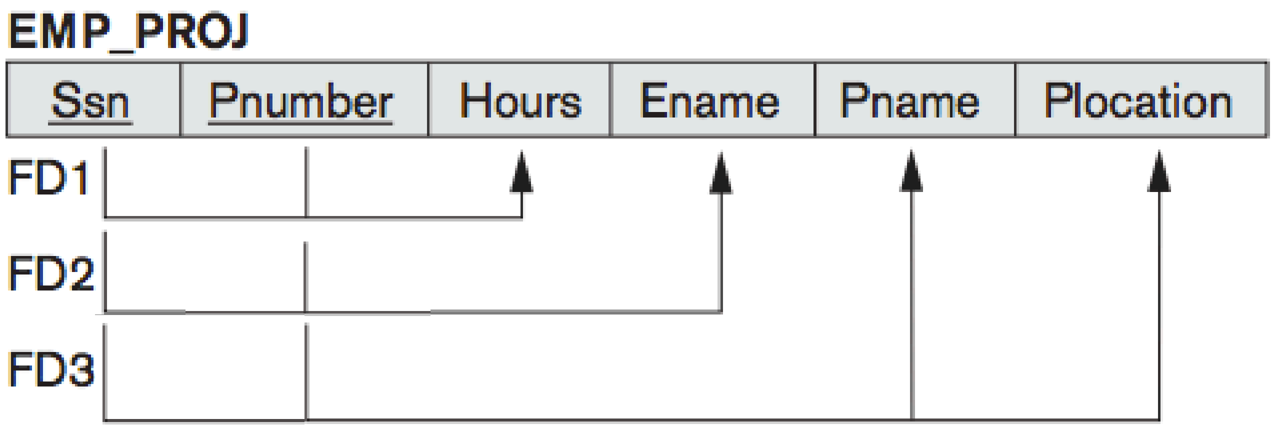


- Composite Primary Key – {Ssn, Pnumber}
- {Ssn,Pnumber} → Hours – *Full FD*
- {Ssn,Pnumber} → Ename – *Partial FD*
- {Ssn,Pnumber} → {Pname, Plocation} – *Partial FD*

Achieving 2NF

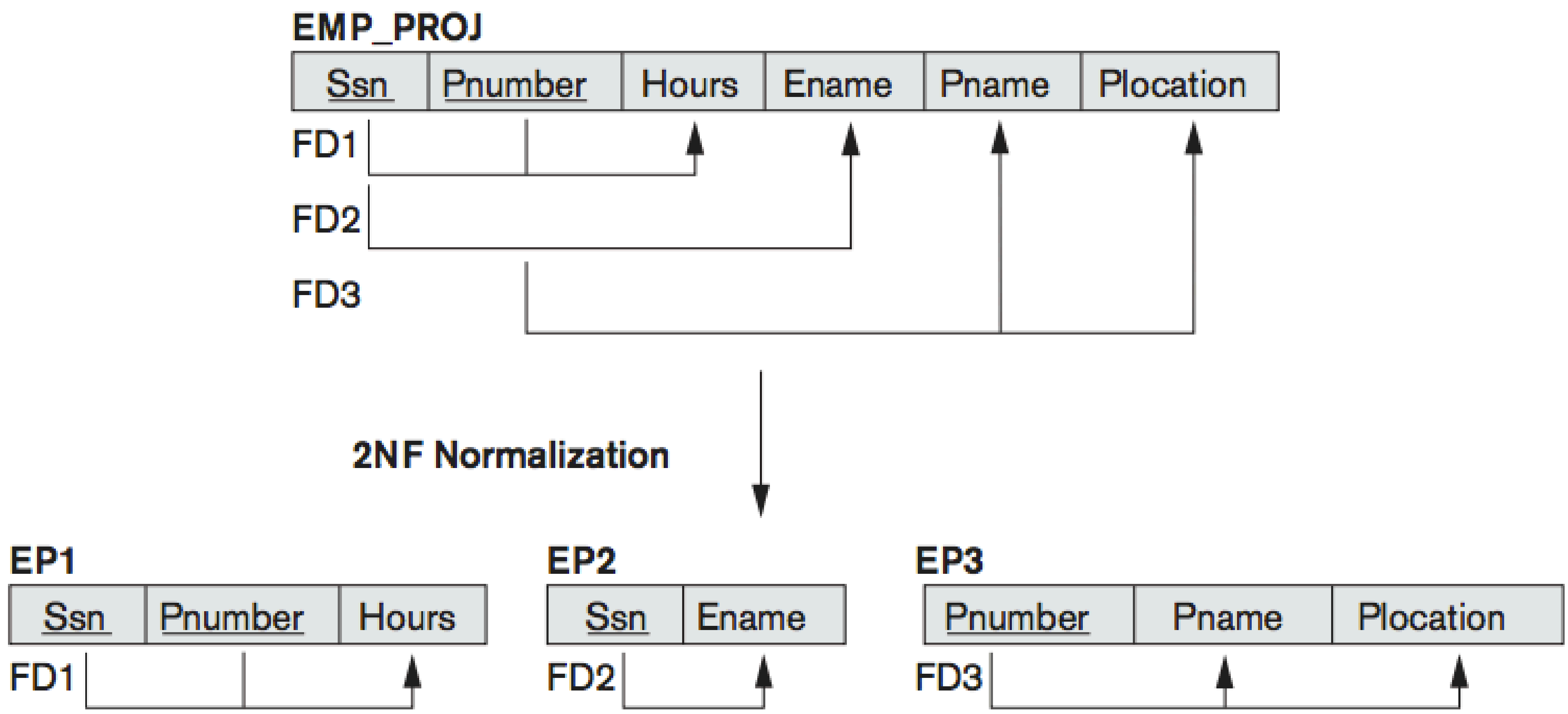
- Limit the FDs to only the parts of the key that they are dependent upon
- Decompose the relation into separate relations using these FDs
- The primary key of the new relations is the left-hand side of the FD
- This *decomposes* a non-2NF relation into a set of new 2NF relations

2NF Example



- Limit the FDs to only the parts of the key that they are dependent upon
- Decompose the relation into separate relations using these FDs
- The primary key of the new relations is the left-hand side of the FD

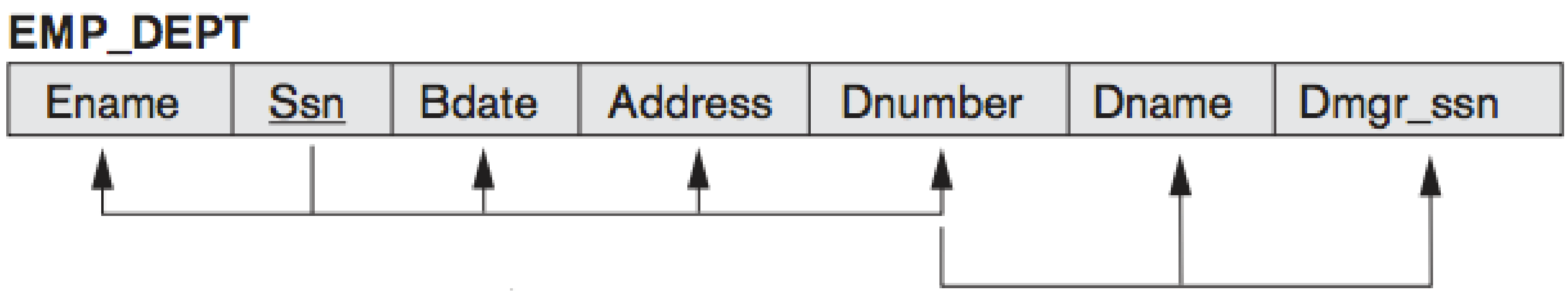
Achieving SNF



Third Normal Form

- A table is said to be in Third Normal Form (3NF) if:
  - it is 2NF compliant
  - No non-key attributes are *transitively dependent* upon the primary key.
- A functional dependency $X \rightarrow Y$ in the relation R, is said to be a *transitive dependency* if:
  - there exists a set of attributes Z in R which is neither a candidate key or a subset of any key of R
  - and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold true

Transitive Dependency

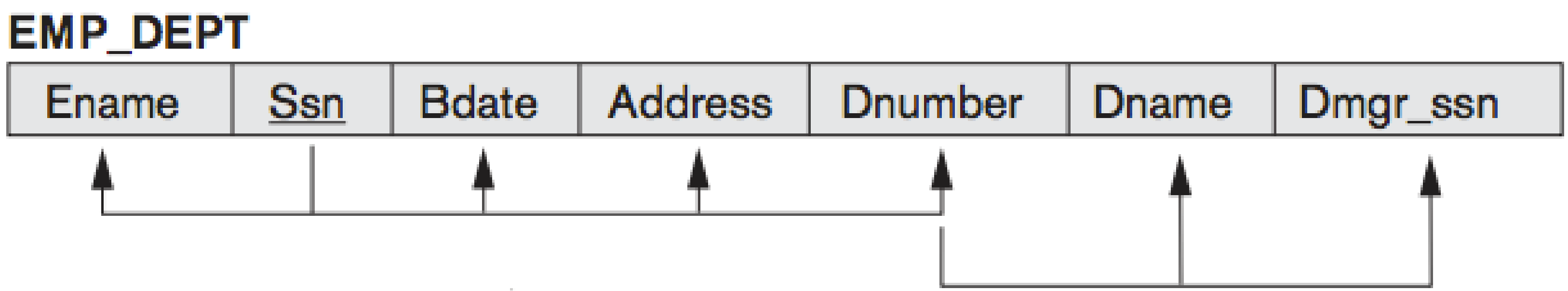


- Ssn➔Dmgr_ssn is a transitive dependency through Dnumber
  - Ssn➔Dnumber and Dnumber➔Dmgr_ssn hold
  - Dnumber is not a key itself or a subset of any key of EMP_DEPT

Achieving 3NF

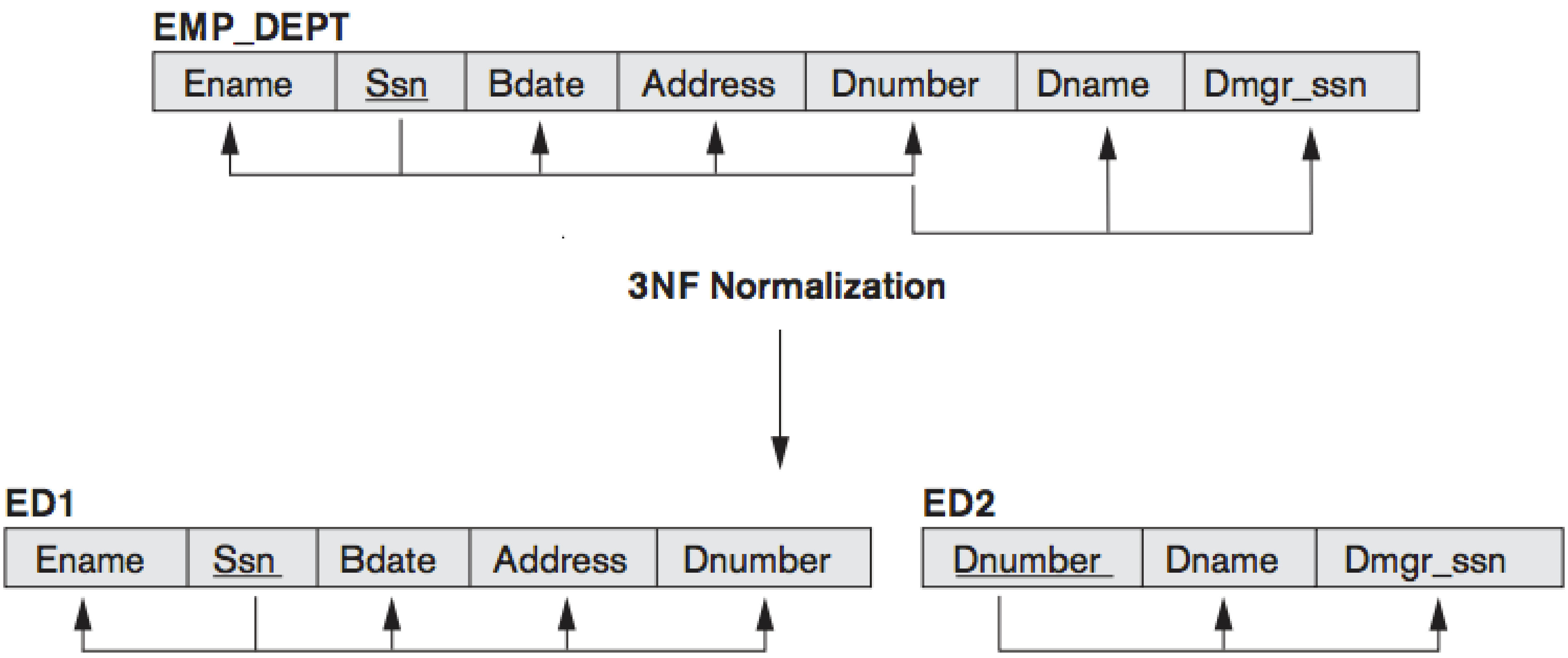
- Identify any transitive dependencies in the relation
- Decompose the relation into two separate relations using these transitive dependencies
- The primary key of the new relation is the middle attribute of the transitive dependency
- This *decomposes* a non-3NF relation into two new 3NF relations

3NF Example



- Identify any transitive dependencies in the relation
- Decompose the relation into two separate relations using these transitive dependencies
- The primary key of the new relation is the middle attribute of the transitive dependency

Achieving 3NF



kdeg
knowledge and data engineering group

2NF and 3NF

- Any functional dependency in which the left hand side is…
  - a non-key attribute or
  - a component attribute of a composite primary key

  …is a problematic FD
- 2NF and 3NF remove these problem FDs by decomposing them into new relations

Summary 1NF – 3NF

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).

Superkey

- A *superkey* SK is any set of attributes in the relation R, whose combined values will be unique for every tuple
  - $t_1[SK] \neq t_2[SK]$
- Every relation has one default superkey – the set of all its attributes
  - as, by definition, every instance of a relation must be unique

Superkey

Car

<u>Engine_No</u>	Reg_Year	Reg_County	Reg_Num	Model	CC

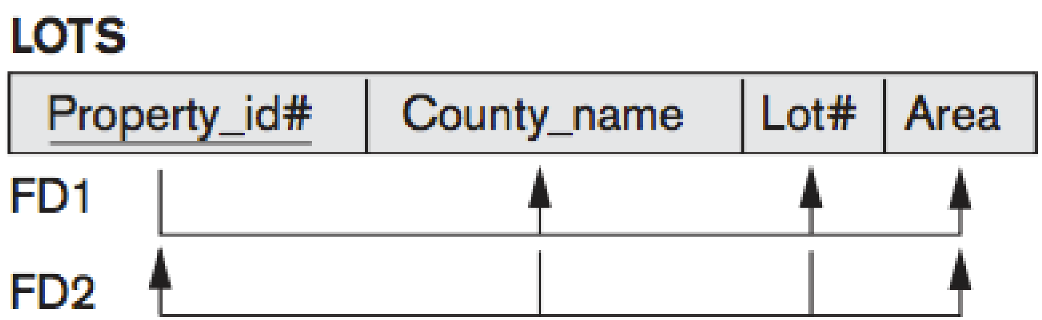
- Candidate Keys
  - Engine_No
  - {Reg_Year, Reg County, Reg Num}
- Primary Key
  - Engine_No
- Superkeys
  - {Engine_No, Reg_Year, Reg County, Reg Num}
  - {Engine_No, Reg_Year, Reg County, Reg Num, Model}
  - {Engine_No, Reg_Year, Reg County, Reg Num, Model, CC}
  - {Reg_Year, Reg County, Reg Num, Model}
  - …

Boyce-Codd Normal Form

- BCNF was created to be a simpler form of 3NF
  - Sometimes called 3.5NF
- However, it was found to be stricter than 3NF
  - Every relation in BCNF is also in 3NF
  - Every relation in 3NF is not necessarily in BCNF
- A table is said to be in Boyce-Codd Normal Form (BCNF) if:
  - whenever a functional dependency $X \rightarrow Y$ holds in the relation R, X is a superkey of R

BCNF

- Consider the following relation schema which describes parcels of land available for sale in different counties



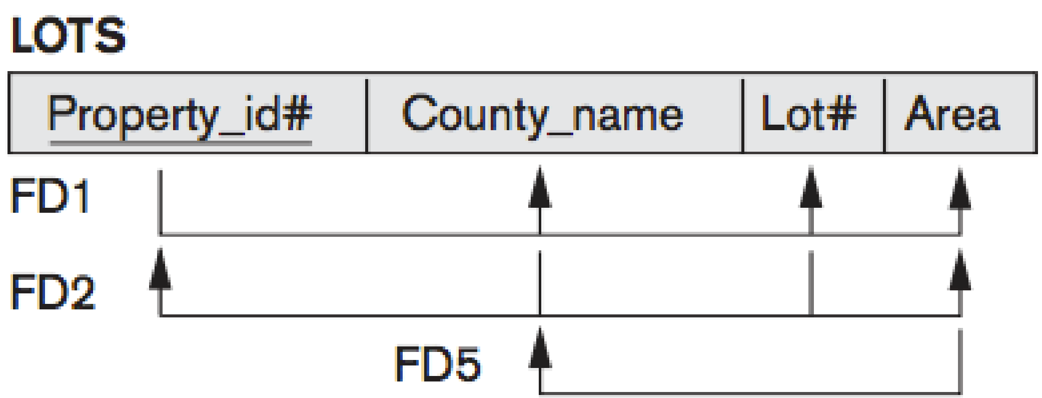
- Two candidate keys
  - Property_id#
  - {County_name, Lot#}

BCNF

- Suppose that there are thousands of lots in the LOTS relation, but:
  - the lots are from only two counties: Wicklow and Monaghan
  - Valid lot sizes in Wicklow are only 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0 acres
  - Valid lot sizes in Monaghan are 1.1, 1.2, ..., 1.9, and 2.0 acres.

BCNF

- In such a situation there would be an additional functional dependency
  - Area→County_name

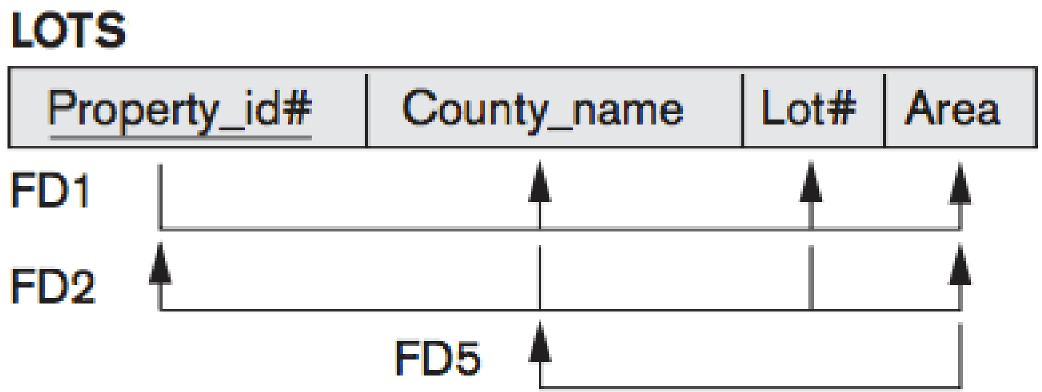


- This breaks BCNF as Area is not a superkey

Achieving BCNF

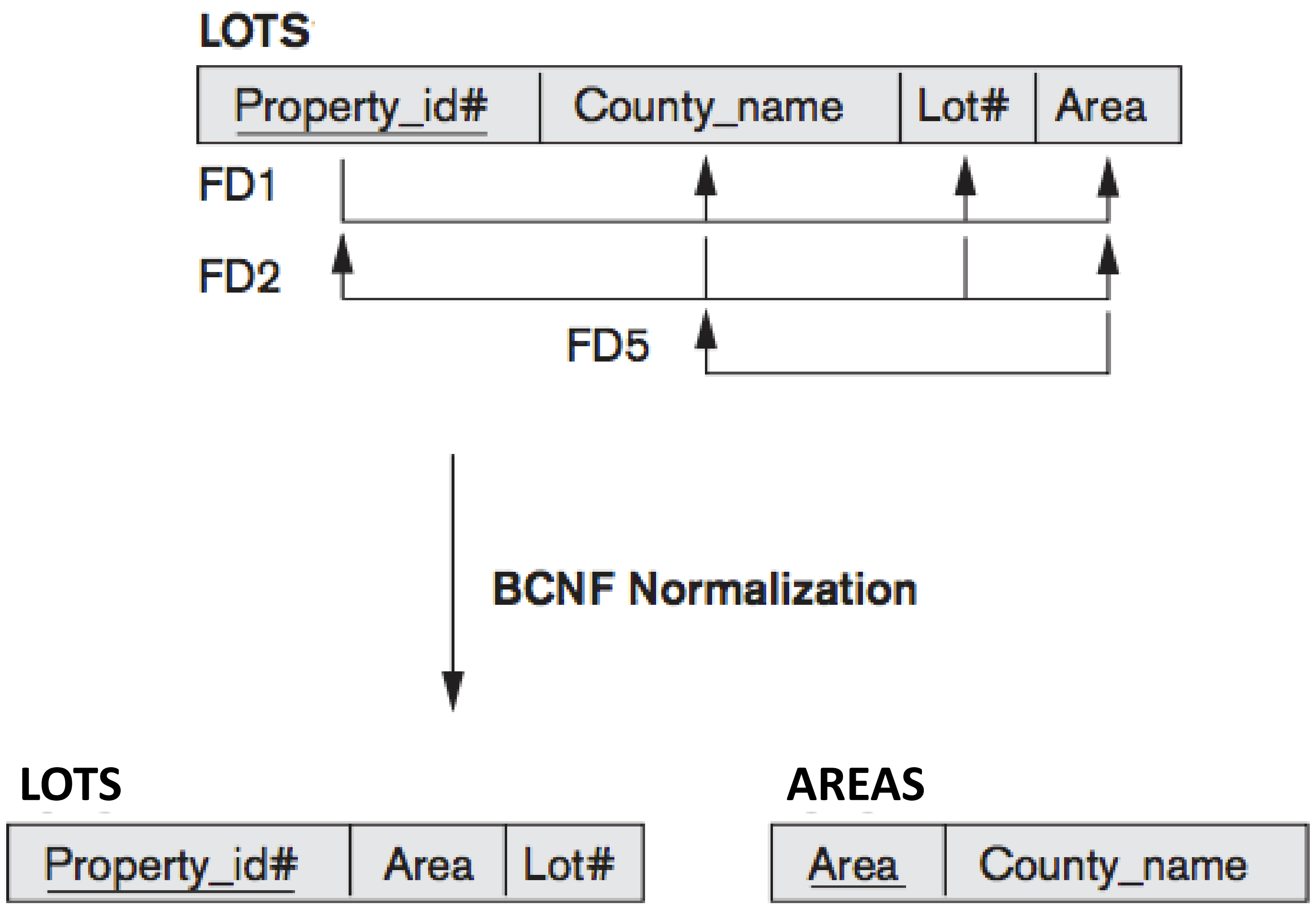
- Identify all functional dependencies in the relations
  - Identify any FDs where the left-hand side is not a superkey
- Decompose the relation into separate relations, creating a new relation for the offending FD
- The primary key of the new relation is the left hand attribute of the offending functional dependency
- This *decomposes* a non-BCNF relation into two new BCNF relations

BCNF Example



- Identify any FDs where the left-hand side is not a superkey
- Decompose the relation into separate relations, creating a new relation for the offending FD
- The primary key of the new relation is the left hand attribute of the offending functional dependency

Achieving BCNF



Normalisation

- Normalisation tests a relation schema to certify whether it satisfies a *normal form*
  - First Normal Form – 1NF
  - Second Normal Form – 2NF
  - Third Normal Form – 3NF
  - Boyce-Codd Normal Form – BCNF
- Evaluate each relation against the criteria for each normal form in turn
  - Decompose relations where necessary

Modelling a Database

- Identify and model
  - the required entities and attributes
  - the relationships between those entities
- Map from the conceptual model to a relational schema
- Identify the functional dependencies in the relation schemas
- Normalise the relation schemas