Chapter 15
Normal Forms (N.F.)

References/Resources:
R. Elmasri, S.B. Navathe, “Fundamentals of Database Systems”, 6th Edition, Pearson Education

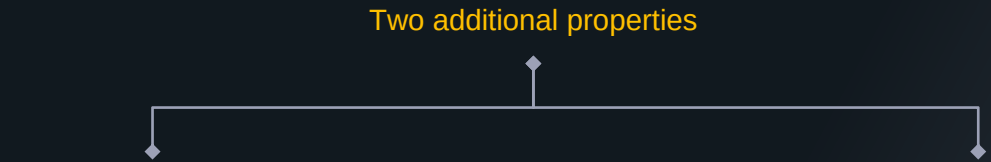
Normalization of data

It is a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of
(1) minimizing redundancy and (2) minimizing the insertion, deletion, and modification anomalies (Update Anomalies).
Unsatisfactory relation schemas that do not meet certain conditions (the **normal form tests)** are decomposed (break the relation into relation) into smaller relation schemas that meet the tests.

Normal Forms

The **normal form** of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

A Good database design = Normal forms + two additional properties



It guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.

It ensures that each f.d is represented in some individual relation resulting after decomposition.

Note: Nonadditive join property is extremely critical and must be achieved whereas the dependency preservation property is sometimes sacrificed.

Denormalization: Reverse of normalization process.

It is the process of storing the join of higher normal form as a base relation, which is in lower normal form.

To find normal forms, we need to understand the concept of superkey, key, candidate key, primary key, secondary key, prime attribute and nonprime attribute.
Please refer chapter 3 (already discussed) for superkey, key, candidate key, primary key and secondary key.

Prime attribute

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

Nonprime attribute

An attribute of relation schema R is called a nonprime attribute of R if it is not a member of any candidate key of R.

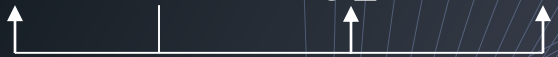
Example: WORKS_ON relation with attributes SSn, Pnumber and Hrs.
Prime attribute = {Ssn, Punumber}
Nonprime attribute = {Hrs}

Normal Forms: 1NF, 2NF and 3NF proposed by Codd and BCNF proposed by Boyce-Codd (order matters)

First Normal Form (1NF)

- It states that the domain of an attribute must include only atomic (simple, indivisible) values.
- The value of any attribute in a tuple must be a single value from its the domain.
- It disallow multivalued attributes, composite attributes, and their combinations.
- Therefore, 1NF disallows relations within relations or relations as attribute values within tuples.
- The only attribute values permitted by 1NF are single atomic (or indivisible) values.

Example: We a have relation with attributes, DEPARTMENT(Dname, Dnumber, Dmgr_ssn, Dlocation) with f.d
Dnumber→Dname, Dmgr_ssn, Dlocation



Let's populate some data in relation, the relation state will be:

DEPARTMENT

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

From relation state it is clearly visible that the relation (DEPARTMENT) is not in 1NF because Dlocation is not atomic attribute. (one cell must contains only one value)

There are 3 techniques to convert DEPARTMENT relation into 1NF:

1. Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT. This decomposes the non-1NF relation into two 1NF relations. DEPARTMENT(Dname, Dnumber, Dmgr_ssn) and DEPT_LOCATIONS(Dnumber, Dlocation)

DEPT_LOCATIONS relation state will be something like this;

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

2. Expand the key of DEPARTMENT relation with a new PK{Dnumber, Dlocation}.
DEPARTMENT relation state:

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

It has a disadvantage of redundancy. (Just to represent Dlocation in a single cell, we introduce redundancy in DEPARTMENT relation…..see Dname and Dmgr_ssn col tuple no. 1,2,3)

3. If a maximum number of values is known for the attribute—for example, if it is known that at most three locations can exist for a department—replace the Dlocations attribute by three atomic attributes: Dlocation1, Dlocation2, and Dlocation3.
Disadvantage: Introduction of NULL values if most departments have fewer than three locations.

NOTE: First technique is considered best because it does not suffer from redundancy.

Refer Fig. 15.10 for another example that is not in 1NF.

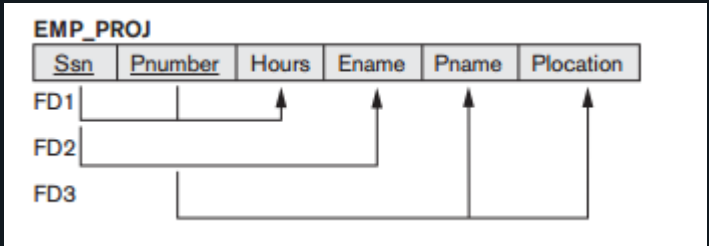
2NF and 3NF with respect to

2 NF and 3NF Based on Primary Key

Second Normal Form (2NF)

- A relation schema R is in 2NF if it is in 1NF and if every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- It is based on the concept of full functional dependency.
- A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more; i.e.for any attribute $A \in X$, $(X - \{A\})$ does *not* functionally determine Y.

Example:



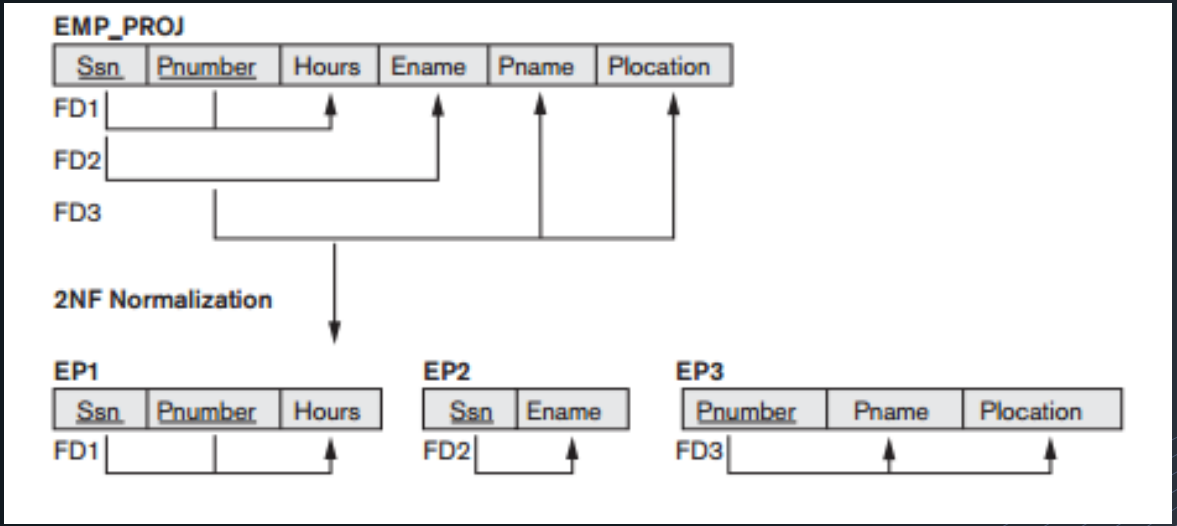
{Ssn, Pnumber} → Hours is a full dependency, because neither Ssn → Hours nor Pnumber → Hours holds

A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds; that is, for some $A \in X$, $(X - \{A\}) \rightarrow Y$
Example: the dependency {Ssn, Pnumber} → Ename is partial because Ssn → Ename holds.

EMP_PROJ is already in 1 NF because right hand side of each FD1, FD2, FD3 are atomic.
Test for 2NF:
The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn, Pnumber} of EMP_PROJ, thus violating the 2NF test.

If a relation schema is not in 2NF, it can be second normalized or 2NF normalized into a number of 2NF relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.

Normalizing EMP_PROJ relation into three relation EP1, EP2, EP3

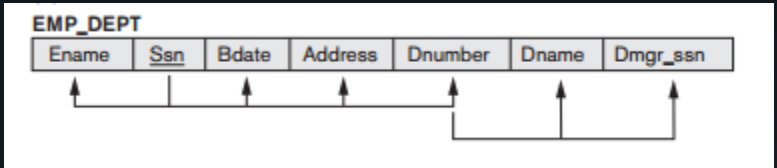


NOTE: Ename and
Pname, Plocation are stored in separate relation.

Third Normal Form (3NF)

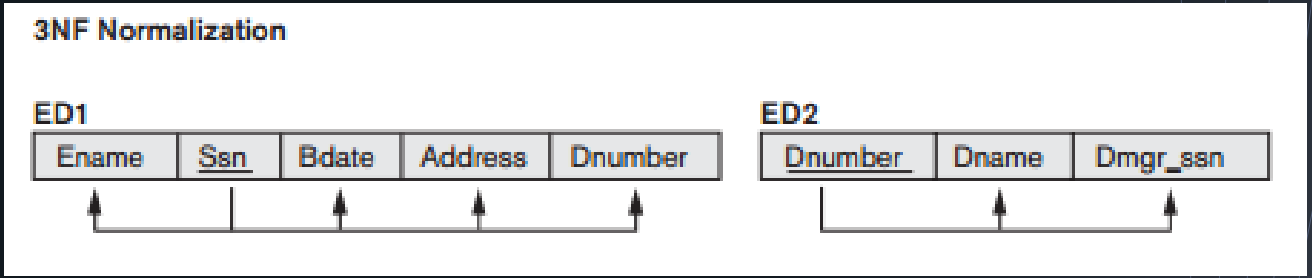
- Third normal form (3NF) is based on the concept of *transitive dependency*.
- A functional dependency $X \rightarrow Y$ in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
- Thus, a relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent
  on the primary key.

Example: In below EMP_DEPT relation 2 FD exits. Ssn → Ename, Bdate, Address, Dnumber
Dnumber → Dname, Dmgr_ssn



In the above FDs Dnumber is transitive dependent because Ssn → Dnumber and Dnumber → Dmgr_ssn hold and Dnumber is neither a key itself nor a subset of the key of EMP_DEPT.

Decomposing EMP_DEPT into ED1 and ED2



NOTE: Dnumber is copied in both relation.

2 NF and 3NF Based on Candidate Key

NOTE: As a general definition of prime attribute, an attribute that is part of any candidate key will be considered as prime.

Second Normal Form (2NF)

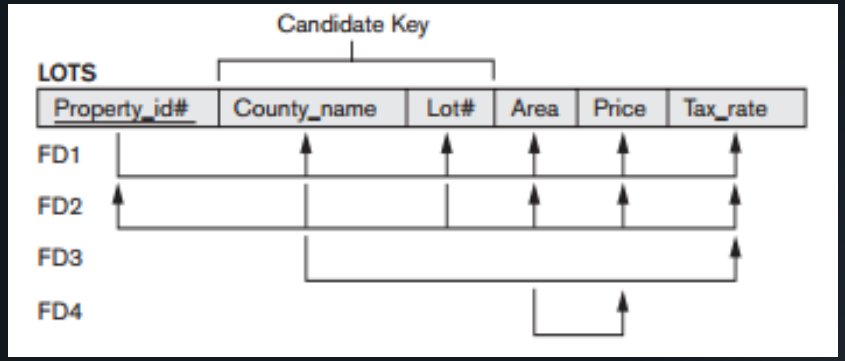
A relation schema *R* is in 2NF if every nonprime attribute *A* in *R* is fully functionally dependent on *every* key of *R*.

Third Normal Form (3NF)

A relation schema R is in 3NF if, whenever a nontrivial functional dependency X → A holds in R,
either (a) X is a superkey of R, or
(b) A is a prime attribute of R.

Example: LOTS relation with candidate key={Property_id#} and {Country_name, Lot#}. We assign Property_id# as primary key of LOTS relation. There are 4 FDs present.

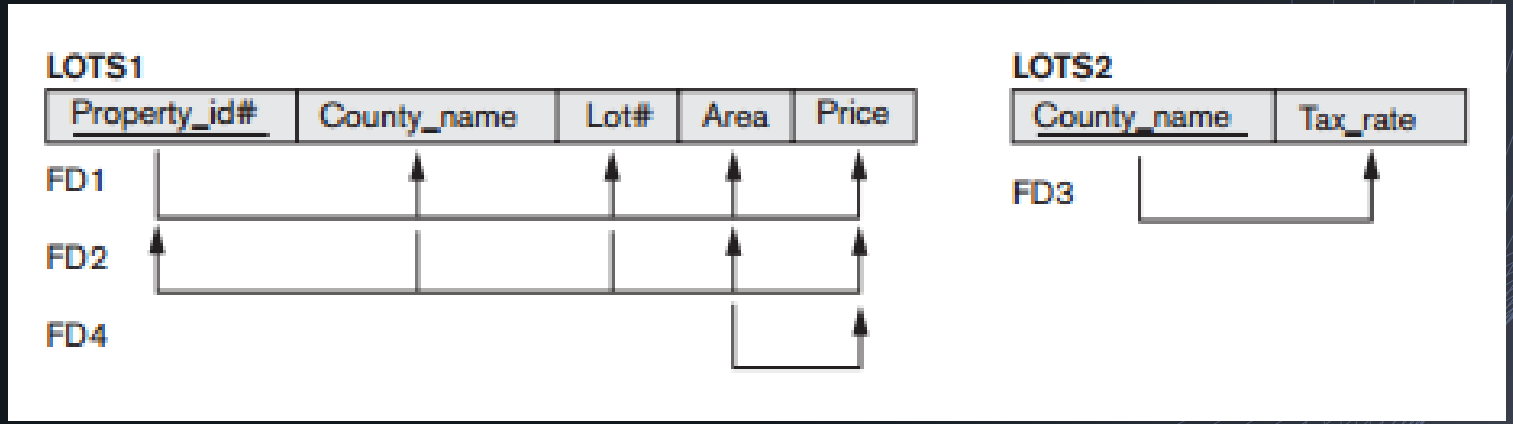
Q: Normalize the relation LOTS to 2NF and then 3NF



Sol: FD1 and FD2 holds because both are fully functionally dependent on candidate key.
FD3 : Country_name → Tax_Rate. This dependency violates 2NF, because Tax_Rate is partially dependent on candidate key (left hand side of FD3 contain only country_name i.e. half of candidate key(Country_name, Lot#))
FD4: Area → Price does not violate 2NF. Because left hand side of FD4 does not contain half of candidate key.

Decomposing LOTS to LOTS1 and LOTS2

2NF Normalization: relation that are in 2NF are LOTS1 and LOTS2

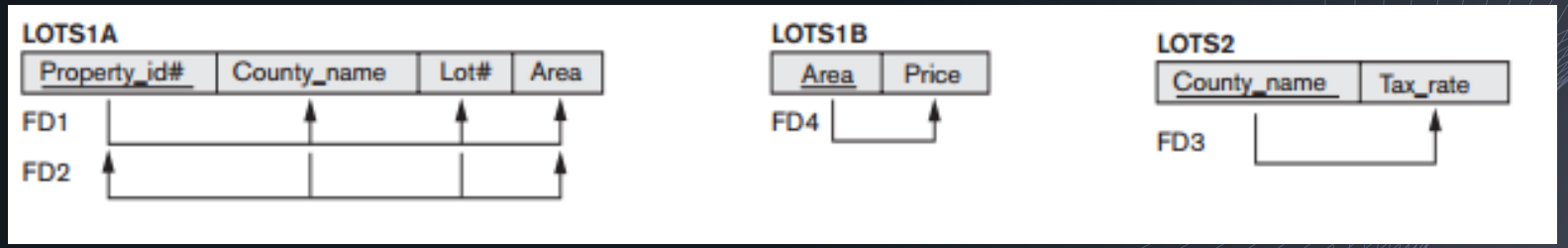


NOTE: Tax_Rate is stored in separate relation.

3NF: LOTS2 is in 3NF because Country_name is a superkey of LOTS2.
In LOTS1 FD1 is in 3NF because Property_id# is a superkey of LOTS1 and FD2 Country_name, Lot# is a superkey
FD4 is not in 3NF because neither Area is a superkey nor Price is a prime attribute in LOTS1.

Decomposing LOTS1 to LOTS1A and LOTS1B

3NF Normalization: relation that are in 3NF are LOTS1A, LOTS1B and LOTS2

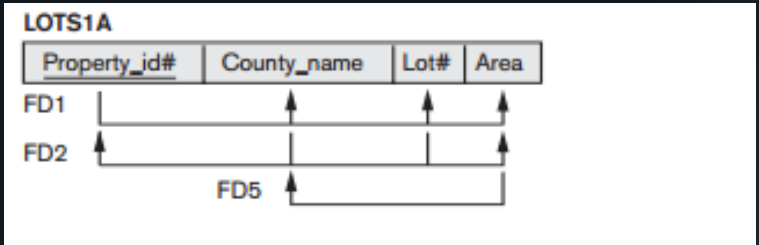


NOTE: Area is copied in both relation (LOTS1A and LOTS1B).

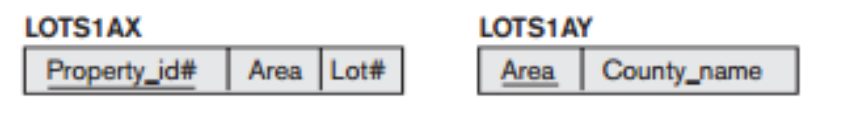
Boyce-Codd Normal Form (BCNF)

- A relation schema *R* is in **BCNF** if whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in *R*, then *X* is a superkey of *R*.
- Every relation in BCNF is also in 3NF but not vice-versa.

Example: In LOTS1A, FD1 and FD2 is in BCNF because left hand side of both FDs has a superkey.
FD5 is not in BCNF because Area → Country_name, Area is not superkey. NOTE that FD5 is in 3NF because right hand side of in FD5 Country_name is prime attribute.



Decomposing LOTS1A into LOTS1AX and LOTS1AY. NOTE FD2 is lost in decomposition.
BCNF normalized:



Thursday will share some solved ques. of Normal Forms and how to find Key, Cover, Minimal cover and Equivalence of two F.Ds