Normalization

Fundamentals of Databases

Dr. Sandra Sampaio
School of Computer Science
University of Manchester

Acknowledgements

These slides are minor adaptations of material authored and made available to instructors by

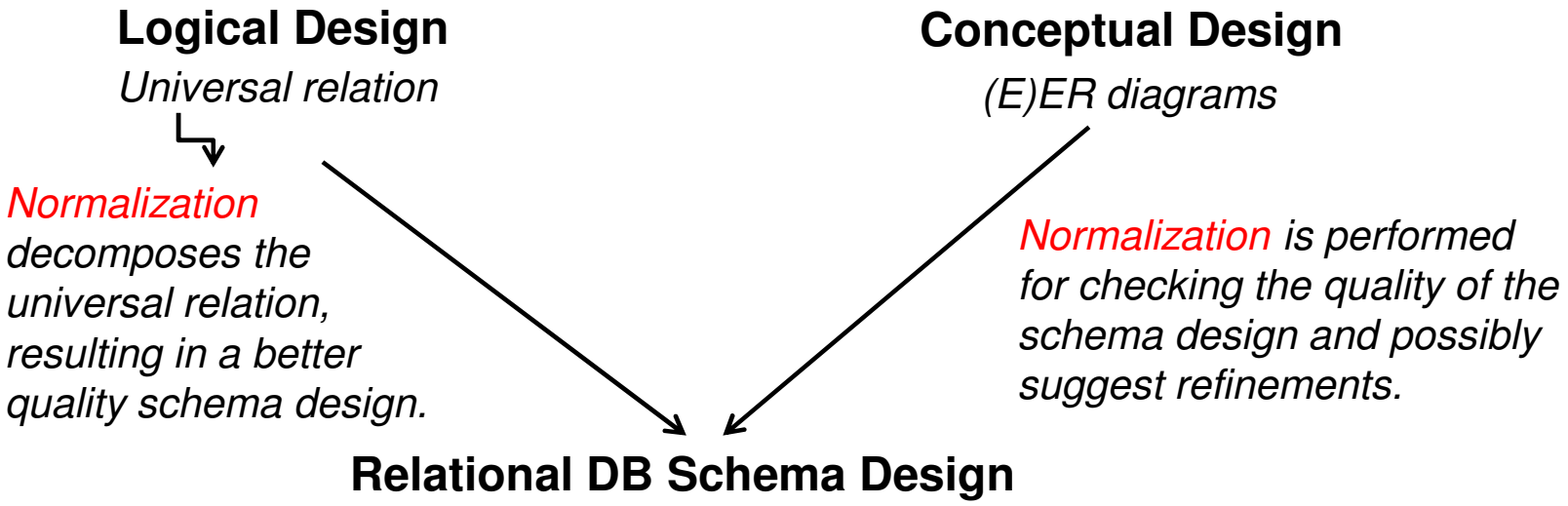
R. Ramakrishnan and J. Gehrke

to accompany their textbook

Database Management Systems, 3rd Edition

Normalization of Relations

- Normalization is the process of decomposing unsatisfactory (i.e., badly designed) relations by breaking up their attributes into smaller relations.
- A normal form is used to certify whether a relation schema is at a particular quality level in its design, by using the keys and functional dependencies (FDs) of the relation.
- 2NF, 3NF, BCNF normal forms are based on keys and FDs of a relation schema.



Practical Use of Normal Forms

- Normalization is carried out in practice so that the resulting schema designs are of high quality and meet the desirable properties.
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are hard to understand or to detect.
- Database designers do not need to normalize to the highest possible normal form (usually up to 3NF or BCNF).
- De-normalization is the process of storing the join of higher normal form relations as a base relation, which is in a lower normal form.

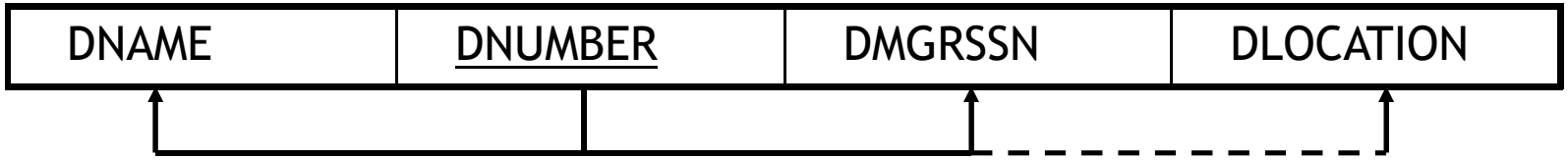
Reviewing Material from Previous Lectures: Definitions of Keys and Prime Attributes

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes with the property that no two tuples t1 and t2 will have t1[S] = t2[S]
- A **key** K is a superkey with the additional property that removal of any attribute from K will cause K not to be a superkey any more.
- If a relation schema has more than one key, each is called a **candidate key**. One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys.
- A **prime attribute** must be a member of some candidate key. A **nonprime attribute** it is not a member of any candidate key.
- A functional dependency X->Y is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more.
  - In other words, for any attribute A in X, {X – A} does not functionally determine Y.

1st Normal Form (1NF) -Eliminate Repeating Groups (1)

Disallows attributes whose values for an individual tuple are **non-atomic**.

(a) DEPT



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

1NF -Eliminate Repeating Groups (2)

(b) DEPT

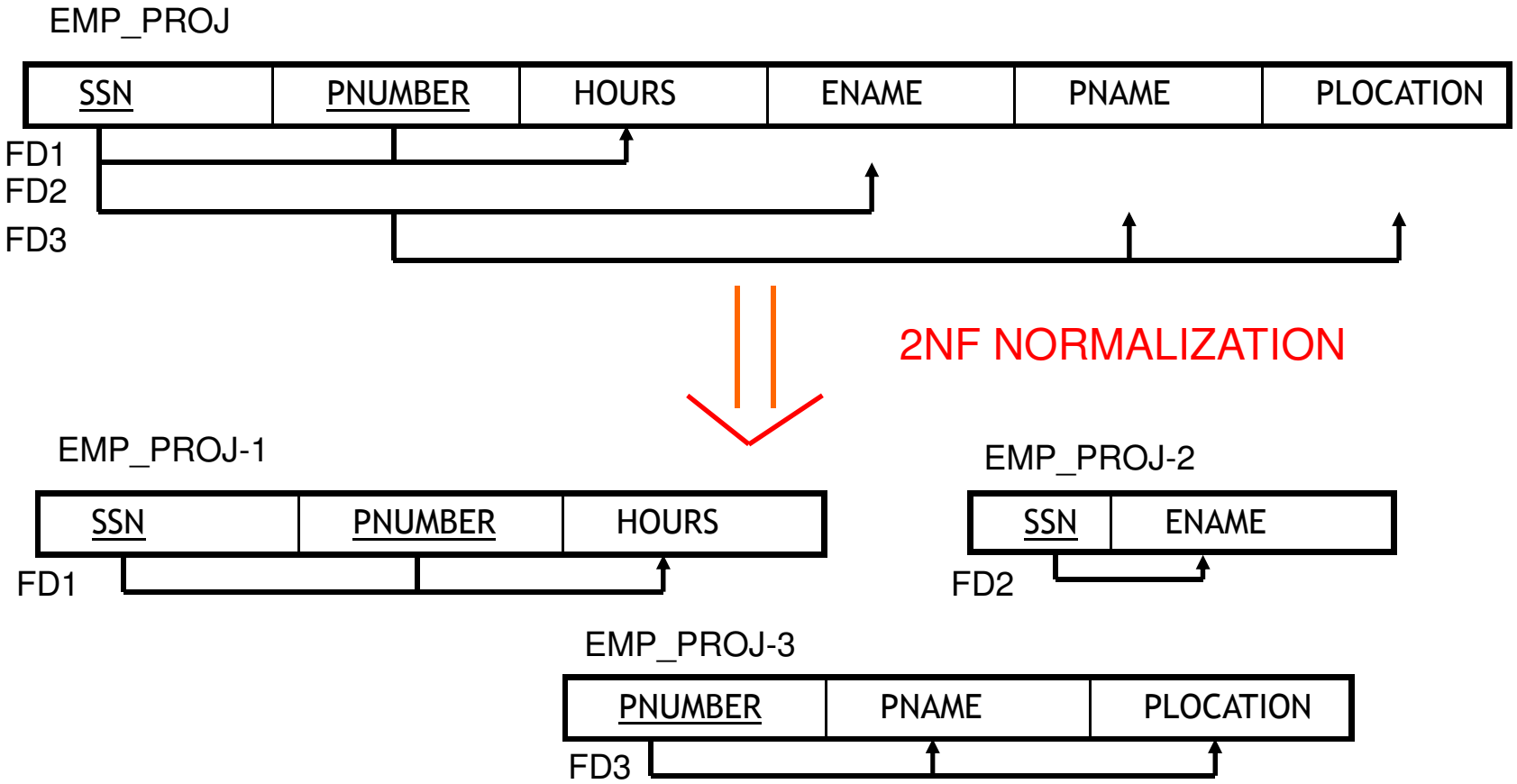
DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	{Bellaire}
Research	5	333445555	{Sugarland}
Research	5	333445555	{Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

The 1NF is a requirement implicit in the definition of Relational model, and, as such it must always hold. However, newer database systems are relaxing this requirement. But this is out of the scope of this course...

How would you decompose relation DEPT?

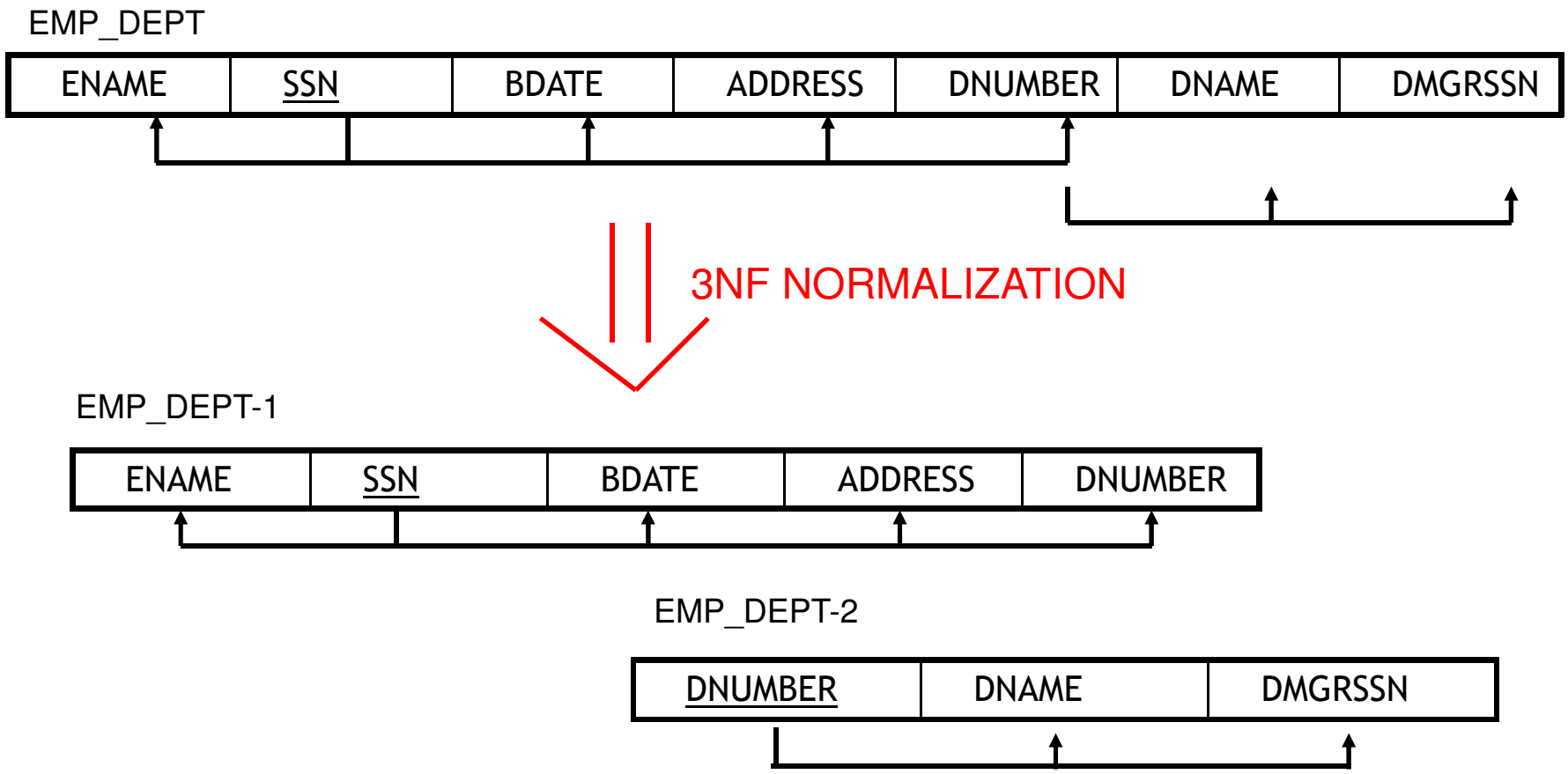
2NF -Eliminate Redundant Data

A relation schema R is in second normal form (2NF) if it is in 1NF and if every **non-prime attribute** A in R is **fully functionally dependent** on the primary key.



3NF -Eliminate Columns Not Dependent On Key

A relation schema R is in third normal form (3NF) if it is in 2NF and **no** non-prime attribute A in R is **transitively dependent** on the primary key.

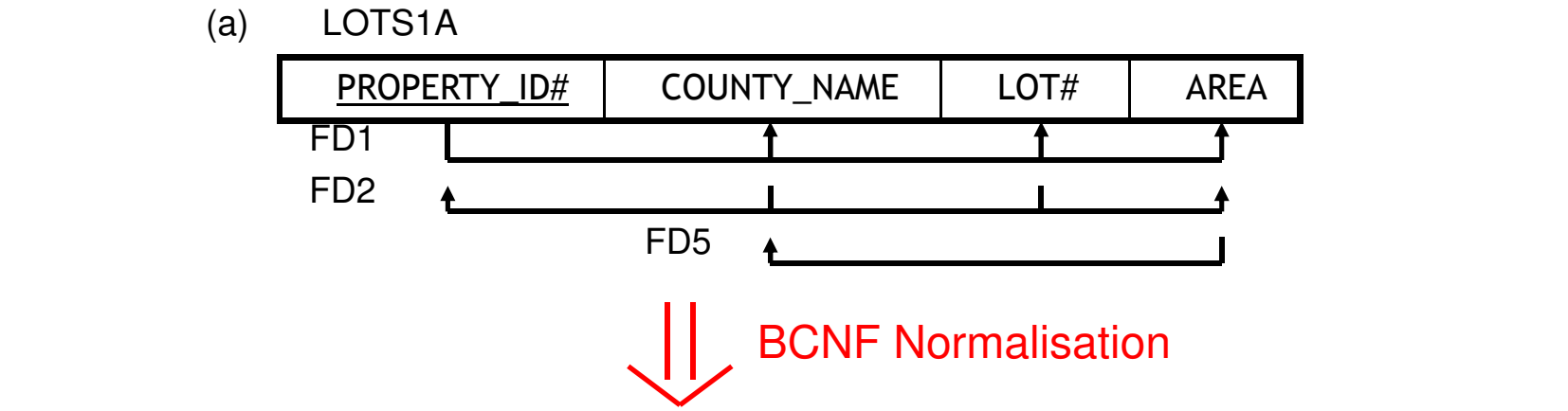


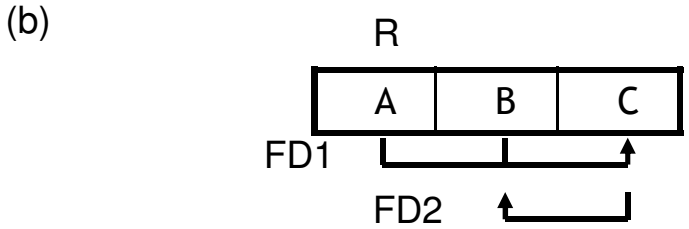
BCNF - Boyce-Codd Normal Form (1)

- A relation schema R is in Boyce-Codd Normal Form (BCNF) if it is in 3NF and if whenever an FD X -> A holds in R, then X is a superkey of R.
- This means that it is NOT in BCNF if:
  - There are multiple candidate keys.
  - The keys are composed of multiple attributes, and there are common attributes between the keys.

3NF is a compromise, used when BCNF is not achievable (e.g., performance considerations).

BCNF - Boyce-Codd Normal Form (2)





BCNF - Boyce-Codd Normal Form (3)

- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
  - No dependency in R that can be predicted using FDs alone.
  - If we are shown two tuples that agree upon the X value, we cannot differ the A value in one tuple from the A value in the other.
  - If example relation is in BCNF, the 2 tuples must be identical (since X is a key).

X	Y	A
x	y1	a
x	y2	?

FD X->A would violate BCNF, if X is not a key,
or if X does not contain A (as it is the case). Since X is a key (the relation is in BCNF), then in the 2nd tuple, the value of A is *a*, and *y1=y2*. However, since a relation is defined to be a set of tuples, we cannot have two copies of the same tuple, and so, this situation cannot arise.

Summary

- Normalization is the process of decomposing unsatisfactory relations by breaking up their attributes into smaller relations.
- A normal form is used to certify whether a relation schema is at a particular quality level in its design, by using the keys and functional dependencies (FDs) of the relation.
- Relations in 1NF disallow attributes whose values for an individual tuple are non-atomic. Since 1NF is a requirement implicit in the definition of the Relational Model, it must hold.
- The 2NF and 3NF still allow presence of redundancy which can be detected via FDs. However, BCNF ensures that no redundancy can be detected via FD information alone. And thus it is the most desirable normal form from the point of view of redundancy.