

An Introduction to the Database Management Systems

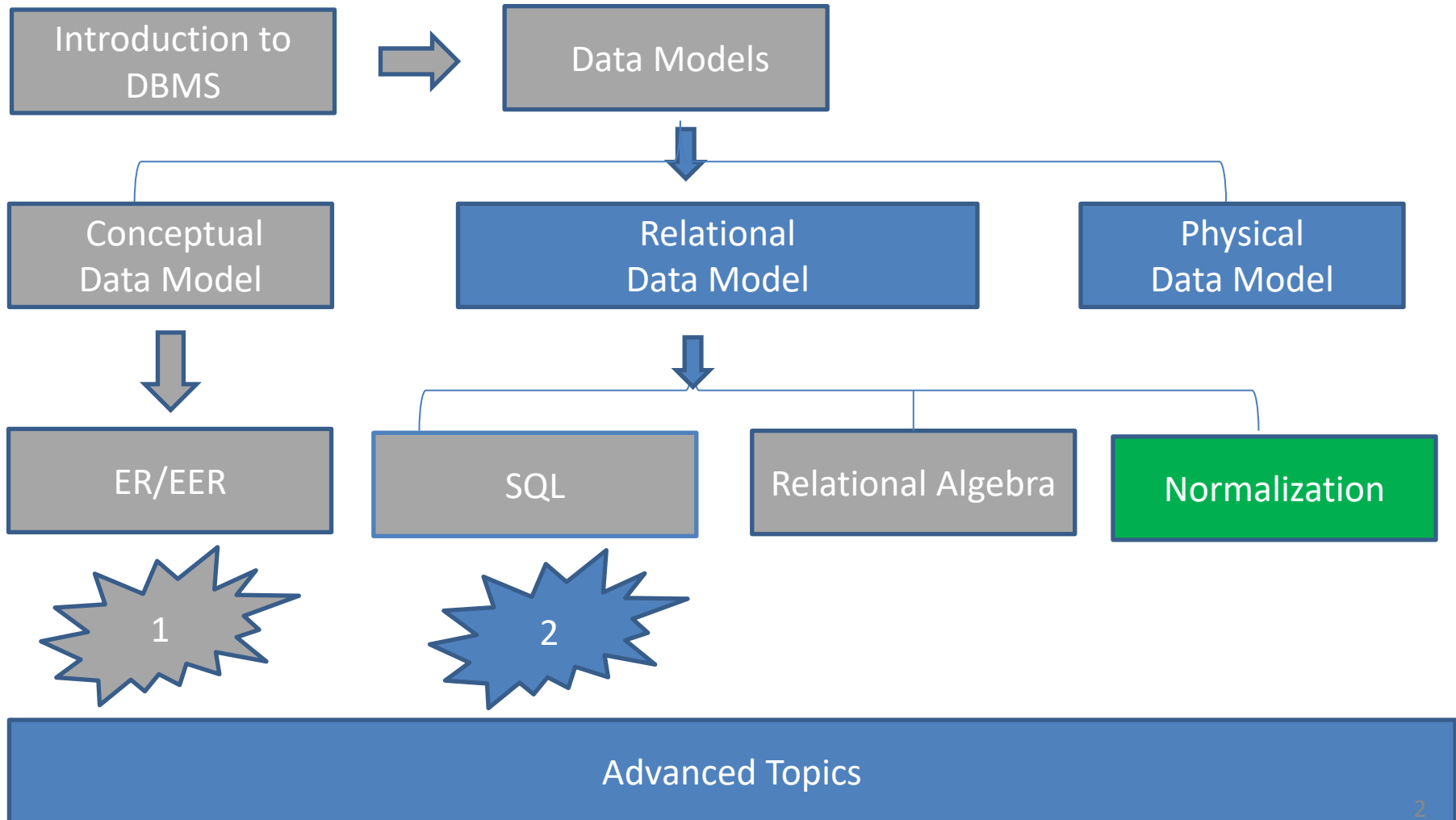
By
Hossein Rahmani

Slides originally by Book(s) Resources




Road Map

(Might change!)



Normalization

- Informal Design Guidelines for Relation Schemas
- Functional Dependencies
- Normal Forms 

Normal forms

- Invented by Codd as a test on relational database schemas
 - The tests ('normal forms') grow more severe. The more severe the test, the higher the normal form, the more robust the database
 - If a schema does not pass the test, it is decomposed in partial schemas that do pass the test
 - It is not always necessary to reach the highest possible normal form

1NF - First Normal Form

- Attributes can only be single-valued
 - Is a basic demand of most relational databases
- Example of a non-1NF relation (see next slide). This normally is already ruled out by the definition of a relation (so using the relational database model automatically ensures 1NF)

Non-1NF - example

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations
-------	----------------	----------	------------

(b)

DEPARTMENT

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

(c)

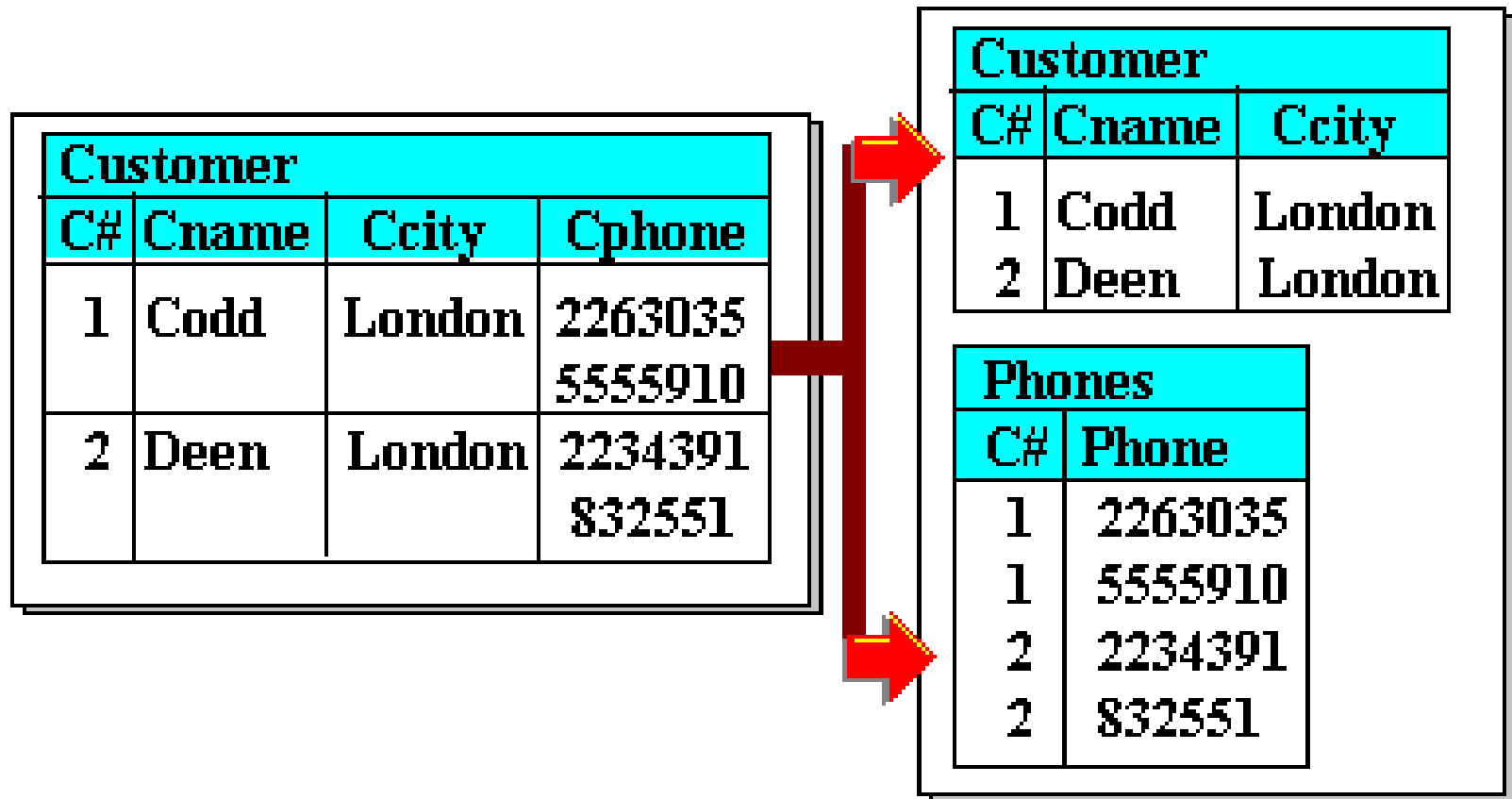
DEPARTMENT

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

1NF - First Normal Form

- Solutions for a multi-valued attribute A in R:
 1. Preferred: create new relation S with A and a foreign key to R
 2. Extend the key of R with an index number for the values of A (redundancy!); e.g. department has no. 5A, or 5B, or 5C
 3. Determine the maximum number of values per tuple for A (say *k*) and replace attribute by *k* attributes (say, loc1, loc2, and loc3). This introduces null-values!

1NF - First Normal Form



2NF - Second Normal form

- Definition: $X \rightarrow Y$ is a partial functional dependency if there is an attribute A in X s.t. $X - \{A\} \rightarrow Y$
- $X \rightarrow Y$ is total if it is not partial
- 2NF: each non-primary attribute is totally dependent on primary key (and not on parts of the primary key)

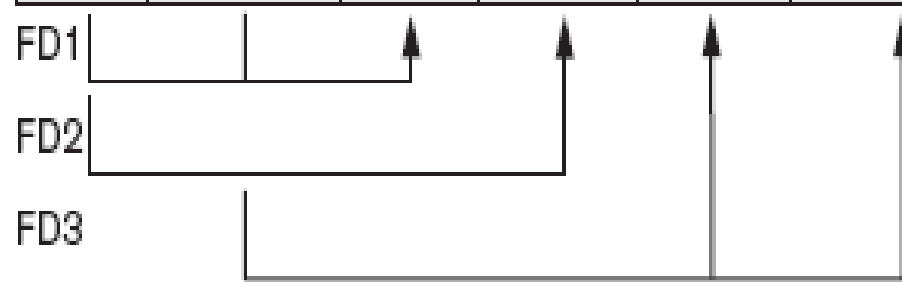
2NF - Normalizing

- Break up the relation such that every partial key with their dependent attributes is in a separate relation. Only keep those attributes that depend totally on the primary key
- Example (see next slide)

2NF - example

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------



2NF Normalization

EP1

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------



EP2

<u>Ssn</u>	Ename
------------	-------



EP3

<u>Pnumber</u>	Pname	Plocation
----------------	-------	-----------



3NF - Third Normal Form

- Definition: $X \rightarrow Y$ is a transitive dependency if there is a Z that is not (part of) a candidate key s.t. $X \rightarrow Z$ and $Z \rightarrow Y$
- 3NF: no non-primary attribute is transitively depending on the primary key

3NF - Normalizing

- Break up the relation such that the attributes that are depending on not-key attributes appear in a separate table (together with the attributes on which they depend)
- Example (see next slide)

3NF - example

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
-------	------------	-------	---------	---------	-------	----------

Functional dependencies for EMP_DEPT:
Ssn → Ename, Bdate, Address, Dnumber
Dnumber → Dname, Dmgr_ssn

3NF Normalization

ED1

Ename	<u>Ssn</u>	Bdate	Address	Dnumber
-------	------------	-------	---------	---------

Functional dependencies for ED1:
Ssn → Ename, Bdate, Address, Dnumber

ED2

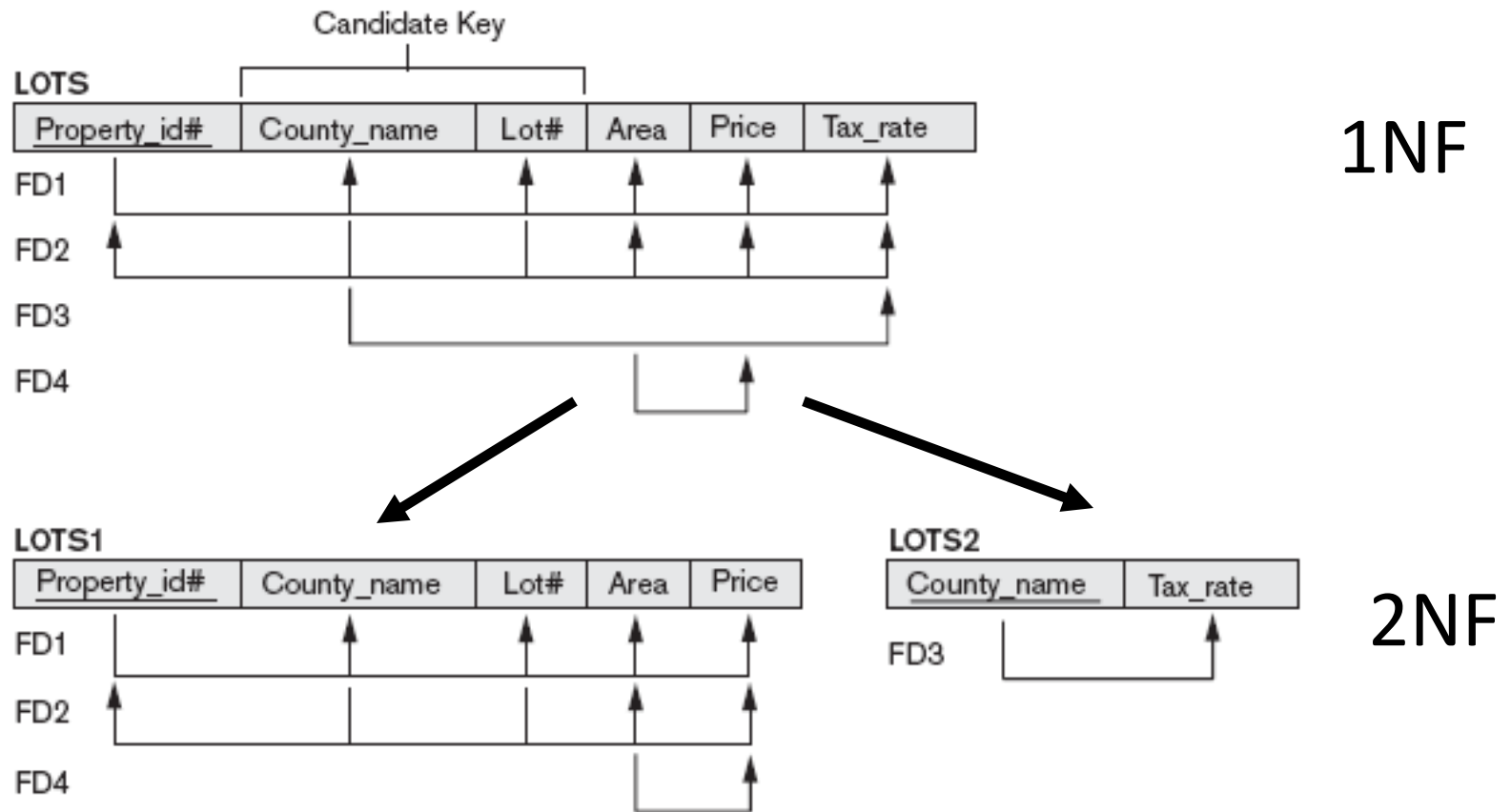
<u>Dnumber</u>	Dname	Dmgr_ssn
----------------	-------	----------

Functional dependencies for ED2:
Dnumber → Dname, Dmgr_ssn

General form 2NF and 3NF

- Put the same demands on **all candidate keys** (super keys) – which is more severe
 - 2NF: every non-key attribute is totally dependent on all keys
 - 3NF: no non-key attribute is transitively dependent on any key
- **Other formulation:**
 - if $X \rightarrow A$ then A is prime or X is a super key
- Example (see next slide)

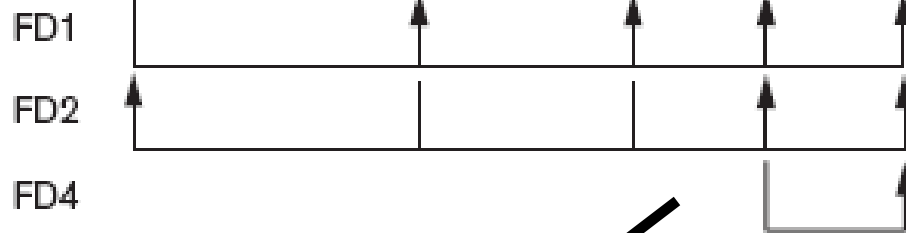
General form 2NF and 3NF - example



General form 2NF and 3NF - example

LOTS1

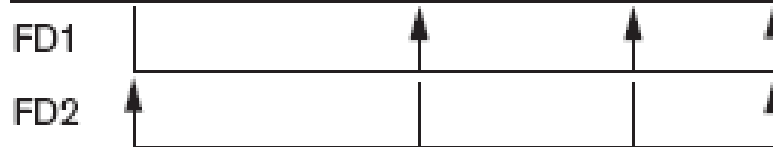
<u>Property_id#</u>	County_name	Lot#	Area	Price
---------------------	-------------	------	------	-------



2NF

LOTS1A

<u>Property_id#</u>	County_name	Lot#	Area
---------------------	-------------	------	------



LOTS1B

<u>Area</u>	Price
-------------	-------



3NF

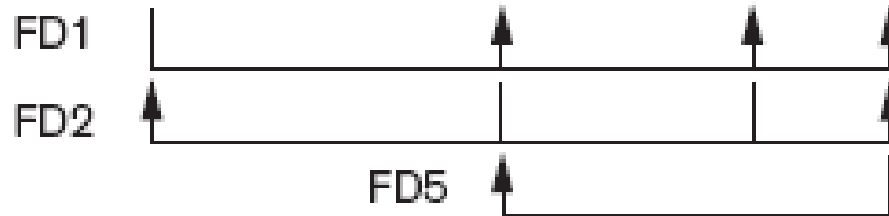
Boyce-Codd Normal Form

- Simpler, but stronger than 3NF
- BCNF: for each *non-trivial* dependency $X \rightarrow A$ holds that X is a super key
- Difference: in 3NF if A is a prime attribute, X does not have to be super key
- In many cases a 3NF schema is also BCNF

BCNF example

LOTS1A

<u>Property_id#</u>	County_name	Lot#	Area
---------------------	-------------	------	------



BCNF Normalization

LOTS1AX

<u>Property_id#</u>	Area	Lot#
---------------------	------	------

LOTS1AY

<u>Area</u>	County_name
-------------	-------------

Decompositions

- Only adhering to a normal form is not enough
- We must not lose attributes in the process!
- Non-additive Join-property:
 - a natural join of the result of a decomposition should result in the original table, without spurious tuples
- There exist algorithms to automatically find good decompositions

ER-schema to relational schemas

- A relational database schema that is mapped from an ER-schema is often in BCNF, but always in 3NF (so, check if BCNF is applicable and useful)
- Many CASE-tools can map an ER-schema automatically into a good relational schema (e.g., SQL create-table commands)

Summary

- Informal guidelines for good design
- Functional dependency
 - Basic tool for analyzing relational schemas
- Normalization:
 - 1NF, 2NF, 3NF, BCNF

Quiz 4

- Suppose you are given a relation R with four attributes $ABCD$. For each of the following sets of FDs, assuming those are the only dependencies that hold for R , do the following:
 - (a) Identify the candidate key(s) for R .
 - (b) Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF).
 - (c) If R is not in BCNF, decompose it into a set of BCNF relations that preserve the dependencies.

Quiz 4

- 1. $C \rightarrow D, C \rightarrow A, B \rightarrow C$
- 2. $B \rightarrow C, D \rightarrow A$
- 3. $ABC \rightarrow D, D \rightarrow A$
- 4. $A \rightarrow B, BC \rightarrow D, A \rightarrow C$
- 5. $AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B$