Normalization of Relations

- **Normalization:**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- Practical Use of Normal Forms:Normalization is carried out in practice so that the resulting designs are of high quality and meet the desirable properties

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]

Candidate Keys:

Example:Super key

Emp_SSN	Emp_Number	Emp_Name
---------	----------	--------
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Robert

- {Emp_SSN}
- {Emp_Number}
- {Emp_SSN, Emp_Number}
- {Emp_SSN, Emp_Name}
- {Emp_SSN, Emp_Number, Emp_Name}
- {Emp_Number, Emp_Name}

Example:candidate key

a candidate key is a minimal super key with no redundant attributes

- {Emp_SSN}
- {Emp_Number}

A Primary key is selected from a set of candidate keys. This is done by database admin or database designer. We can say that either {Emp_SSN} **or** {Emp_Number} can be chosen as a primary key

<Student>

Student_ID	Student_Enroll	Student_Name	Student_Email
S02	4545	Dave	ddd@gmail.com
S34	4541	Jack	jjj@gmail.com
S22	4555	Mark	mmm@gmail.com

The following are the super keys for the above table −

```
{Student_ID}
{Student_Enroll}
{Student_Email}
{Student_ID, Student_Enroll}
{Studet_ID, Student_Name}
{Student_ID, Student_Email}
{Student_Name, Student_Enroll}
{Student_ID, Student_Enroll, Student_Name}
{Student_ID, Student_Enroll, Student_Email}
{Student_ID, Student_Enroll, Student_Name, Student_Email}
```

The following would be the candidate key from the above −

```
{Student_ID}
{Student_Enroll}
{Student_Email}
```

3.2 First Normal Form 1NF

- A relation scheme R is in first normal form (1NF) if the values in *dom* (*A*) are atomic for every attribute A in R.

- **Disallows**
  - composite attributes
  - Set-valued attributes
  - **nested relations**; a cell of an *individual tuple* is a complex relation

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

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

(c)

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

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
		20	10.0
999887777	Zelaya, AliciaJ.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

Composite attributes

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 10.9

Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Second Normal Form

- **A relation scheme R is in second normal form (2NF) with respect to a set of FDs F if it is in 1NF and every nonprime attribute is fully dependent on every key of R.**

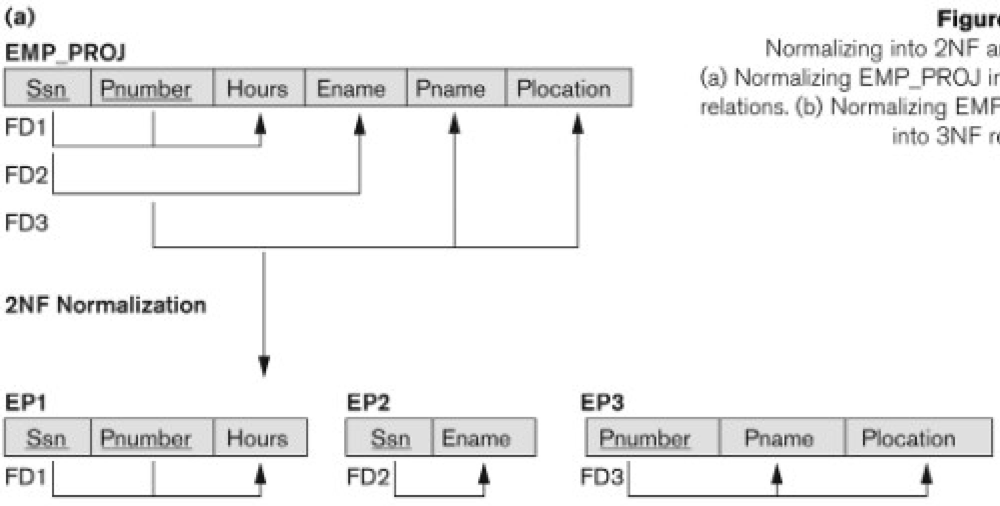
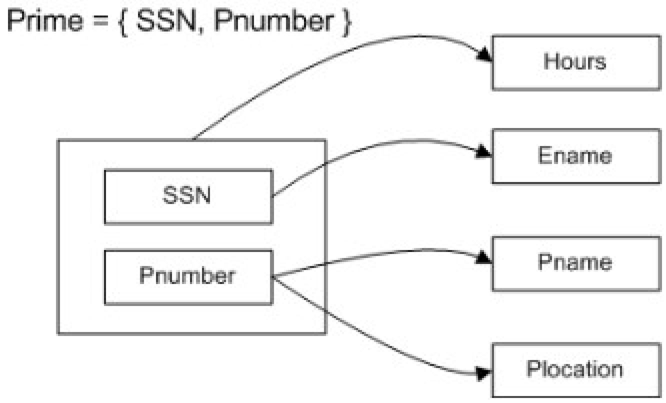
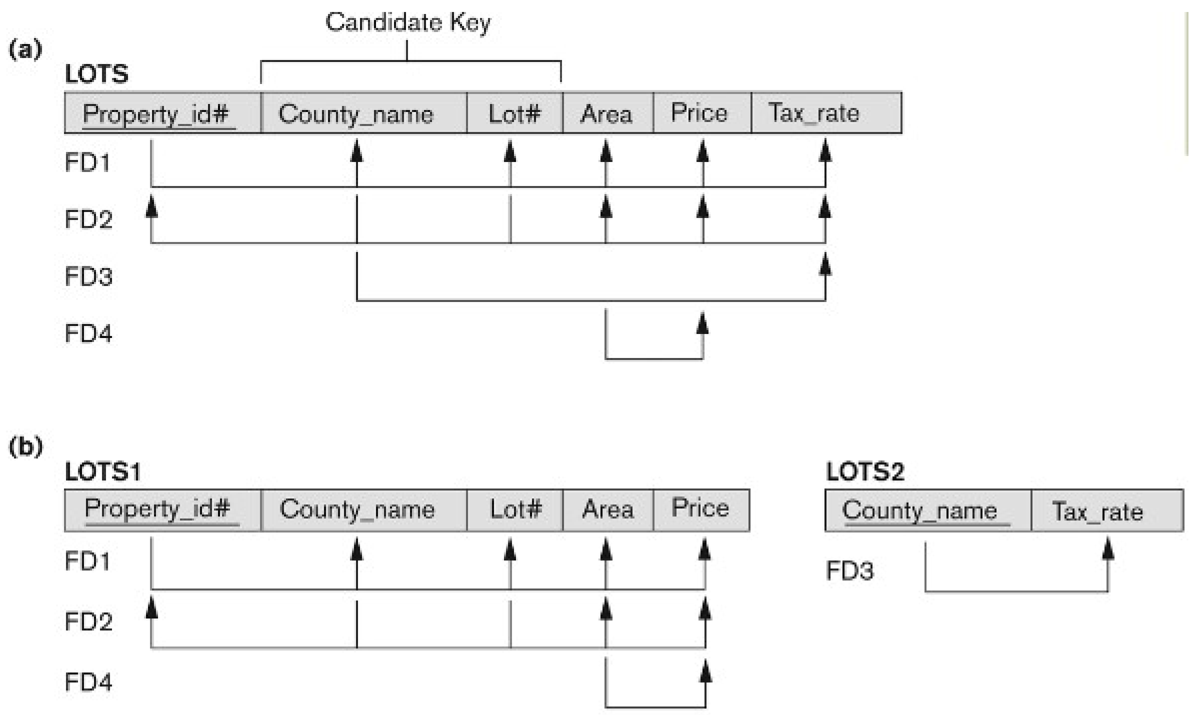


Figure 10.10
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.




(a)
Candidate Key
LOTS
Property_id#
County_name
Lot#
Area
Price
Tax_rate
FD1
FD2
FD3
FD4
(b)
LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4
LOTS2
County_name
Tax_rate
FD3

Student(IDSt, StudentName, IDProf, ProfessorName, Grade)

1. The attribute ProfessorName is functionally dependent on attribute IDProf (IDProf --> ProfessorName)

2. The attribute StudentName is functionally dependent on IDSt (IDSt --> StudentName)

3. The attribute Grade is fully functional dependent on IDSt and IDProf (IDSt, IDProf --> Grade)

Students

IDSt	LastName	IDProf	Prof	Grade
1	Mueller	3	Schmid	5
2	Meier	2	Borner	4
3	Tobler	1	Bernasconi	6

Startsituation

Result after normalisation

Students

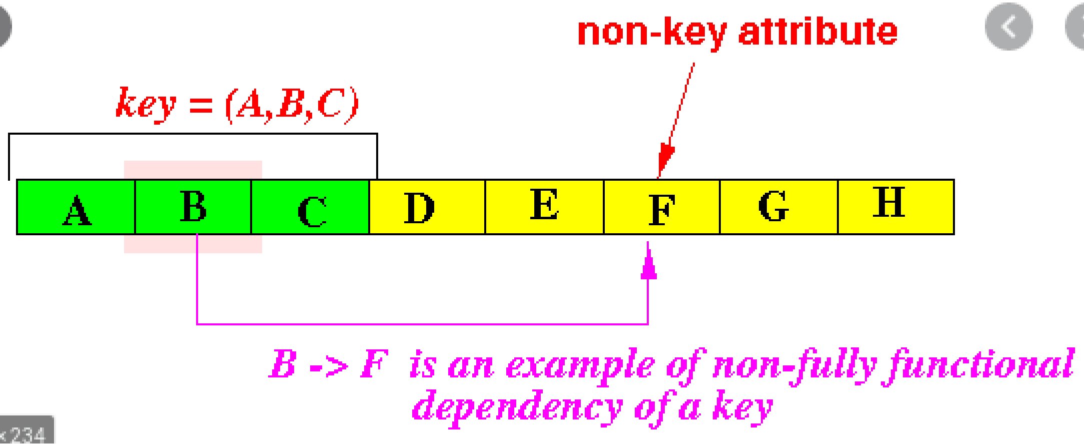
ID	LastName
1	Mueller
2	Meier
3	Tobler

Professors

IDProf	Professor
1	Bernasconi
2	Borner
3	Schmid

Grades

IDStIDProf	Grade	
1	3	5
2	2	4
3	1	6


non-key attribute
key = (A,B,C)
A
B
C
D
E
F
G
H
B -> F is an example of non-fully functional dependency of a key

- A relation schema R is in **third normal form (3NF)** if whenever a FD $X \rightarrow A$ holds in R, then either:
  - (a) X is a superkey of R, or
  - (b) A is a prime attribute of R

SO no non-prime attribute A in R is transitively dependent on the primary key

Example

Key: { SID }

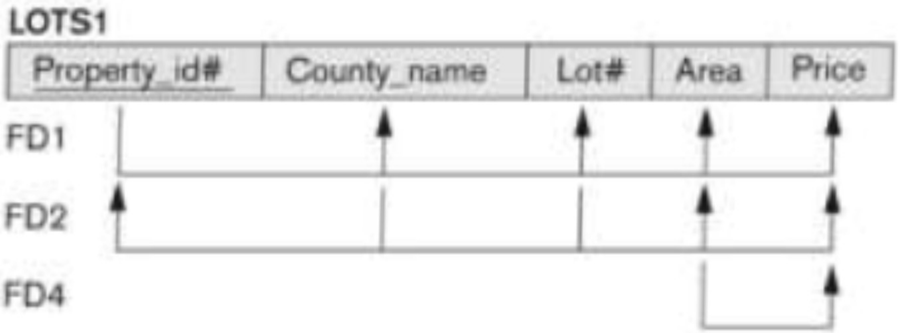
SID→Building

Building →Fee

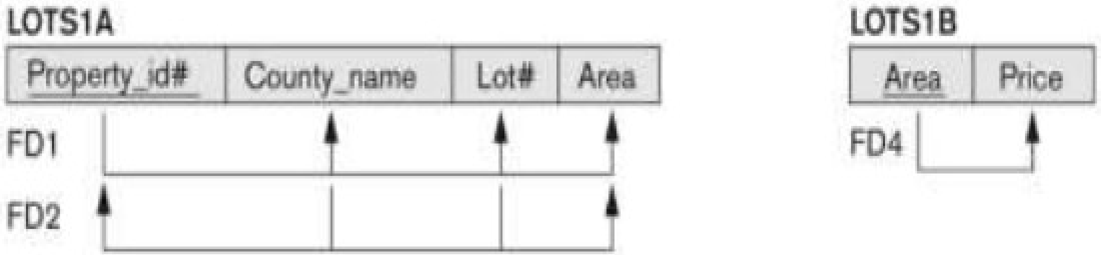
Building →Mgr

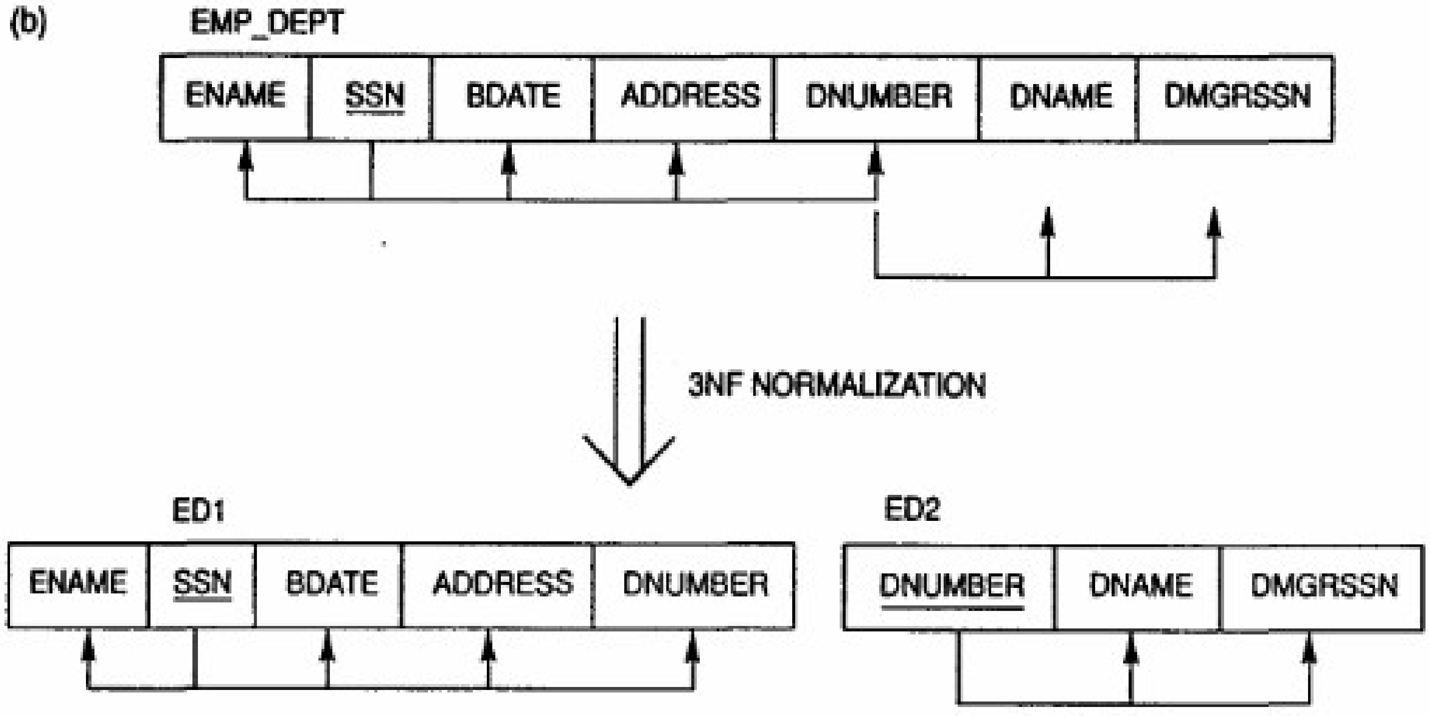
SID	Building	Fee	Manager
100	Fenn	300	Mr. T
300	ΔΠ	400	Ali
200	Holiday Inn	400	Tyson

(b)


LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4

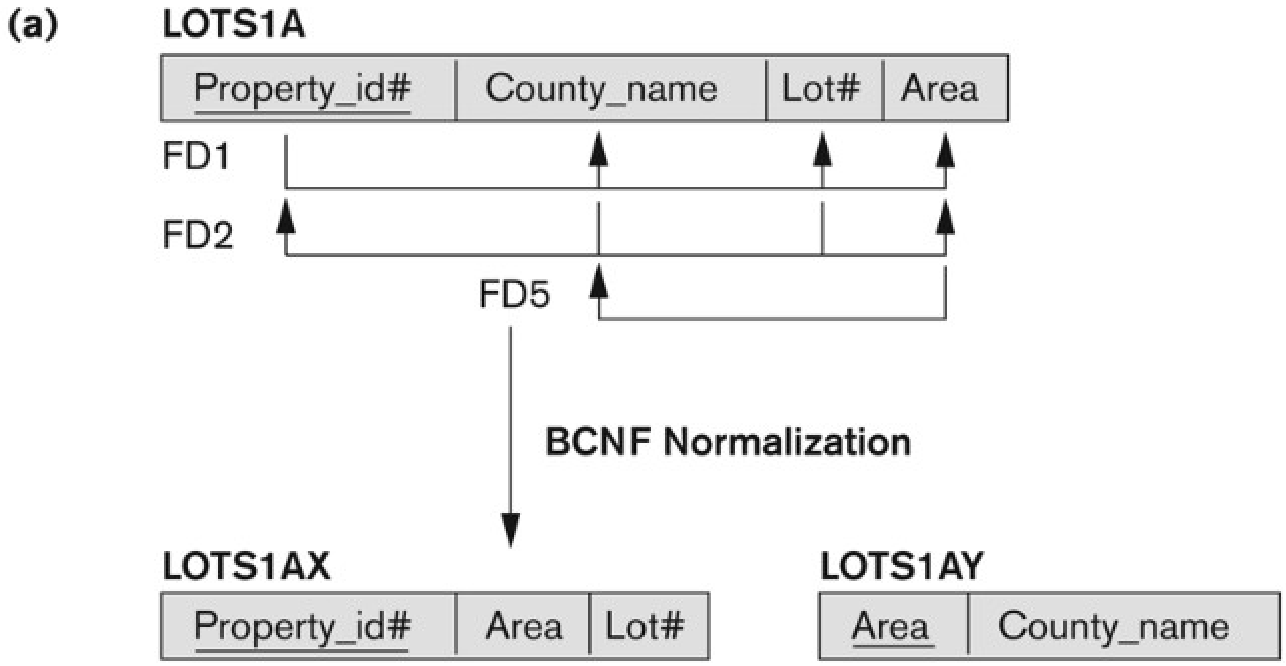
(c)

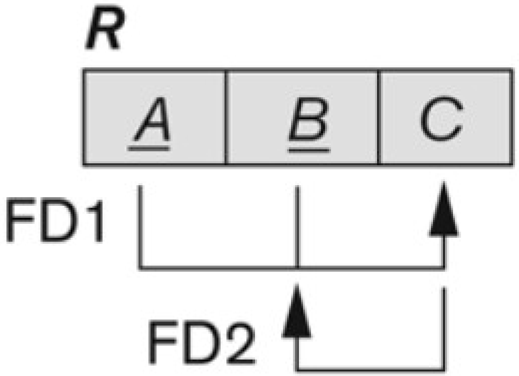

LOTS1A
Property_id#
County_name
Lot#
Area
FD1
FD2
LOTS1B
Area
Price
FD4


(b)
EMP_DEPT
ENAME
SSN
BDATE
ADDRESS
DNUMBER
DNAME
DMGRSSN
3NF NORMALIZATION
ED1
ENAME
SSN
BDATE
ADDRESS
DNUMBER
ED2
DNUMBER
DNAME
DMGRSSN

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X -> A** holds in R, then **X is a superkey** of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)


(a)
LOTS1A
Property_id#
County_name
Lot#
Area
FD1
FD2
FD5
BCNF Normalization
LOTS1AX
Property_id#
Area
Lot#
LOTS1AY
Area
County_name


R
A
B
C
FD1
FD2

Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC**.

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.

R(A, B, C, D)

R1 (A, D, C) R2(D, B)

Breaking, table into two tables, one with A, D and C while the other with D and B.