

## Normalization Process

Fix normal form violations by *decomposing* relations.

- for a functional dependency  $X \rightarrow Y$ , split  $R$  into relations  $XY$  and  $R-Y$ 
  - $X$  becomes the PK in  $XY$

When decomposing a relation  $R$  to satisfy a normal form, must ensure

- lossless join
  - cannot get spurious tuples
- dependency preservation
  - each functional dependency of  $R$  is represented in some relation or can be derived from those that are

for FD  $X \rightarrow Y$ , splitting  $R$  into  $XY$  and  $R-Y$  is lossless as long as  $X \cap Y = \emptyset$  i.e.  $X$  and  $Y$  do not have any attributes in common  
more generally, splitting  $R$  into  $R_1$  and  $R_2$  is lossless as long as  $R_1 \cap R_2$  is a superkey for  $R_1$  or  $R_2$

## Decomposition Example

OrderLine(OrderID, ProductID, Quantity, ProductName)

ProductID  $\rightarrow$  ProductName violates 2NF

| OrderID | ProductID | Quantity | ProductName |
|---------|-----------|----------|-------------|
| 1001    | P7        | 2        | Widget      |
| 1002    | P7        | 5        | Widget      |
| 1002    | P9        | 1        | Gadget      |

same product name repeats on every row with a given product ID  
a product ID can be repeated because it is only part of a key

OrderLine(OrderID, ProductID, Quantity)

Product(ProductID, ProductName)

| OrderID | ProductID | Quantity |
|---------|-----------|----------|
| 1001    | P7        | 2        |
| 1002    | P7        | 5        |
| 1002    | P9        | 1        |

| ProductID | ProductName |
|-----------|-------------|
| P7        | Widget      |
| P9        | Gadget      |

ProductID  $\rightarrow$  ProductName

## Decomposition Example

Employee(EmpID, Name, DeptID, DeptName)

DeptID  $\rightarrow$  DeptName violates 3NF  
EmpID  $\rightarrow$  DeptID [employee is in only one department]

| EmpID | Name | DeptID | DeptName |
|-------|------|--------|----------|
| 1     | Ana  | D1     | Sales    |
| 2     | Ben  | D1     | Sales    |
| 3     | Cal  | D2     | Support  |

same department name repeats on every row with a given department ID  
a department ID can be repeated because it is not a key

Employee(EmpID, Name, DeptID)

Department(DeptID, DeptName)

| EmpID | Name | DeptID |
|-------|------|--------|
| 1     | Ana  | D1     |
| 2     | Ben  | D1     |
| 3     | Cal  | D2     |

EmpID  $\rightarrow$  DeptID

| DeptID | DeptName |
|--------|----------|
| D1     | Sales    |
| D2     | Support  |

DeptID  $\rightarrow$  DeptName

## Decomposition Example

Assignment(EmpID, SSN, Project, Hours)

EmpID  $\rightarrow$  SSN violates BCNF  
SSN  $\rightarrow$  EmpID [each employee has exactly one SSN and vice versa]  
EmpID, Project  $\rightarrow$  Hours

| EmpID | SSN | Project | Hours |
|-------|-----|---------|-------|
| E1    | 111 | Alpha   | 10    |
| E1    | 111 | Beta    | 5     |
| E2    | 222 | Alpha   | 20    |

the SSN is repeated for every project an employee works on

Employee(EmpID, SSN)

Assignment(EmpID, Project, Hours)

| EmpID | SSN |
|-------|-----|
| E1    | 111 |
| E2    | 222 |

EmpID  $\rightarrow$  SSN  
SSN  $\rightarrow$  EmpID

| EmpID | Project | Hours |
|-------|---------|-------|
| E1    | Alpha   | 10    |
| E1    | Beta    | 5     |
| E2    | Alpha   | 20    |

EmpID, Project  $\rightarrow$  Hours

## Preserving Dependencies

TASK(project,manager,task,deadline,budget)

- project → manager [every project has a single manager]
- project,task → deadline [each of the project's tasks has its own deadline]
- manager,deadline → budget [budget specific to manager and deadline]

Is this in 2NF?

- no, because non-key manager depends on project, a subset of a key

Decompose to satisfy 2NF.

TASK(project,task,deadline,budget)  
MANAGER(project,manager)

...but this is not dependency-preserving because manager,deadline → budget is not represented in either relation (or derivable from ones that are)

14

## Preserving Dependencies

TASK(project,manager,task,deadline,budget)

- project → manager [every project has a single manager]
- project,task → deadline [each of the project's tasks has its own deadline]
- manager,deadline → budget [budget specific to manager and deadline]

Try a different decomposition. Observe:

- project → manager violates 2NF because non-key manager depends on project, a subset of a key
- manager,deadline → budget violates 3NF because non-key budget depends on non-key manager,deadline

Decompose – split manager,deadline → budget first, then project → manager:

BUDGET(manager,deadline,budget)  
TASK(project,task,deadline)  
MANAGER(project,manager)

This is dependency-preserving, lossless, and BCNF.

- TASK ∩ MANAGER = { project }, a superkey for MANAGER
- (TASK ∩ MANAGER) ∩ BUDGET = { manager,deadline }, a super key for BUDGET

15

## Preserving Dependencies

Tutoring(Student,Subject,Tutor)

- Student,Subject → Tutor [each student has one tutor per subject]
- Tutor → Subject [each tutor teaches only one subject]

| Student | Subject | Tutor |
|---------|---------|-------|
| Ana     | Math    | Smith |
| Ben     | Math    | Smith |
| Ana     | Physics | Jones |

violates BCNF  
the same tutor and subject combination is repeated for all of a given tutor's students

Tutor(Tutor,Subject)

Tutoring(Student,Tutor)

| Tutor | Subject |
|-------|---------|
| Smith | Math    |
| Jones | Physics |

| Student | Tutor |
|---------|-------|
| Ana     | Smith |
| Ben     | Smith |
| Ana     | Jones |

not dependency preserving

either leave in 3NF where Student,Subject→Tutor is enforced by the PK and accept the redundancy or go with BCNF and enforce the FD through other mechanisms (e.g. trigger)

Tutor → Subject

Consider the following relation:

REGISTERED(student,course,instructor,major)

The functional dependencies are:

- student → major
- course → instructor

no additional candidate keys – {student,course} are only on the left and must be in any candidate key while {major,instructor} are only on the right and can always be derived – there's nothing left!

1NF – has a PK

not 2NF – student and course are both proper subsets of a key and major and instructor are both non-key

– decompose by pulling out XY for each problematic FD, which is both of them

REGISTERED(student,course)

MAJOR(student,major)

INSTRUCTOR(course,instructor)

- this is dependency-preserving – some relation contains both X and Y for every FD  $X \rightarrow Y$
- lossless – columns present in two tables are a superkey for one

now 3NF and BCNF – student and course are both PKs

Consider the following relations:

LOTS(propertyid, county, lotnum, area, price, taxrate)

The functional dependencies are:

- county, lotnum  $\rightarrow$  propertyid
- county  $\rightarrow$  taxrate
- area  $\rightarrow$  price

additional candidate  
key: {county, lotnum}

1NF – has a PK

not 2NF – county is a proper subset of one of the candidate keys  
– decompose by pulling out XY for each problematic FD

TAXRATE(county, taxrate)

LOTS(propertyid, county, lotnum, area, price)

- this is dependency-preserving – some relation contains both X and Y for every FD  $X \rightarrow Y$
- lossless – columns present in two tables are a superkey for one

not 3NF – area  $\rightarrow$  price has non-keys depending on non-keys

– decompose by pulling out XY for each problematic FD

TAXRATE(county, taxrate)

PRICE(area, price)

LOTS(propertyid, county, lotnum, area)

- this is dependency-preserving – some relation contains both X and Y for every FD  $X \rightarrow Y$
- lossless – columns present in two tables are a superkey for one

8

## LOTS

LOTS(propertyid, county, lotnum, area, price, taxrate)

- propertyid  $\rightarrow$  county, lotnum [propertyid also a candidate key]
- county  $\rightarrow$  taxrate [tax rate is determined by the county]
- area  $\rightarrow$  price [price is based on the area]
- area  $\rightarrow$  county [each county has unique lot sizes]

Candidate keys –

- {county, lotnum} – PK
- {propertyid} – propertyid  $\rightarrow$  county, lotnum
- {area, lotnum} – area  $\rightarrow$  county

shortcut if you already have a primary key –  
for any FD  $X \rightarrow A$  where A is part of a known key, replace  
A with X and trim to a minimal key  
repeat with each new key until nothing else is found

CPSC 343: Database Theory and Practice • Fall 2026

139

## LOTS

candidate keys: {county, lotnum},  
{propertyid}, {area, lotnum}

LOTS(propertyid, county, lotnum, area, price, taxrate)

- propertyid  $\rightarrow$  county, lotnum [propertyid also a candidate key]
- county  $\rightarrow$  taxrate [tax rate is determined by the county]
- area  $\rightarrow$  price [price is based on the area]
- area  $\rightarrow$  county [each county has unique lot sizes]

Is this in 2NF?

- no, because non-key taxrate depends on county, a subset of a key
- (note propertyid  $\rightarrow$  county, lotnum does *not* violate 2NF because county and lotnum are key attributes)
- no, because non-key price depends on area, a subset of a key

Decompose to satisfy 2NF.

LOTS(propertyid, county, lotnum, area)

TAXRATE(county, taxrate)

PRICE(area, price)

dependency-preserving  
lossless

10

## LOTS

candidate keys: {county, lotnum},  
{propertyid}, {area, lotnum}

LOTS(propertyid, county, lotnum, area)  
TAXRATE(county, taxrate)  
PRICE(area, price)

- propertyid  $\rightarrow$  county, lotnum [propertyid also a candidate key]
- county  $\rightarrow$  taxrate [tax rate is determined by the county]
- area  $\rightarrow$  price [price is based on the area]
- area  $\rightarrow$  county [each county has unique lot sizes]

Is this in 3NF?

- yes (area  $\rightarrow$  county is the only one without a superkey on the left, but county is a key attribute)

CPSC 343: Database Theory and Practice • Fall 2026

142

## LOTS

candidate keys: {county,lotnum},  
{propertyid}, {area,lotnum}

LOTS(propertyid, county, lotnum, area)  
TAXRATE(county, taxrate)  
PRICE(area, price)

- propertyid → county, lotnum [propertyid also a candidate key]
- county → taxrate [tax rate is determined by the county]
- area → price [price is based on the area]
- area → county [each county has unique lot sizes]

Is this in BCNF?

- no, because county depends on non-superkey area

Decompose to satisfy BCNF.

LOTS(propertyid, lotnum, area)      lossless  
TAXRATE(county, taxrate)      not dependency-preserving –  
PRICE(area, price)      stop at 3NF instead  
COUNTY(area, county) } AREA(area, price, county)