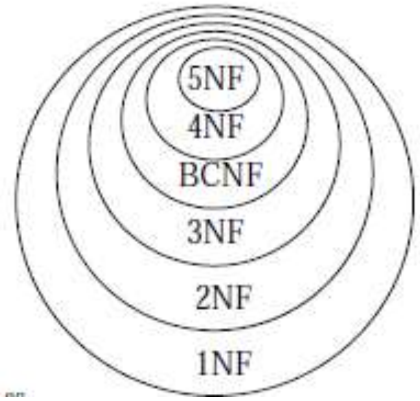
NORMALIZATION

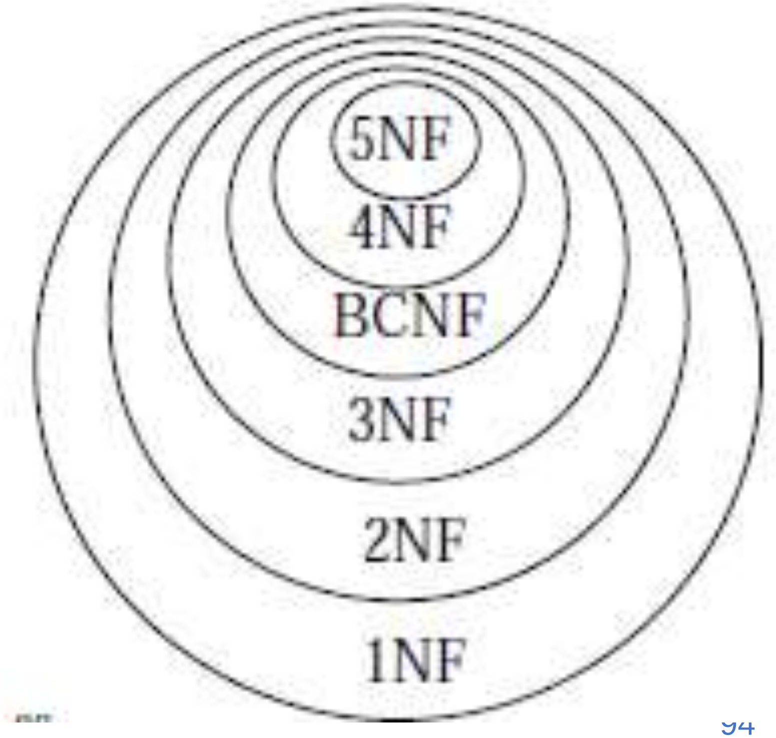
THE COMPLETE BOOK by Ullman

Chapter 3: Design Theory for Relational Databases



Normalization

- The process of decomposing "bad" relation by breaking it into smaller relations
- 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*



Normal Forms

- <u>1st Normal Form (1NF)</u> = trivial (All tables are flat)
- *<u>2nd Normal Form</u> = obsolete (remove partial Dependencies)*
- **<u>3rd Normal Form (3NF)</u>**
- **<u>Boyce-Codd Normal Form (BCNF)</u>**

DB designs based on *functional dependencies*, intended to prevent data ***anomalies***

Major focus is on these dependencies

- *<u>4th Normal Forms</u> = remove Multi-values dependencies*

1^{st} Normal Form (1NF)

Student	Courses
Mary	{CS145,CS229}
Joe	{CS145,CS106}
…	…

Violates 1NF.

<u>Student</u>	<u>Courses</u>
Mary	CS145
Mary	CS229
Joe	CS145
Joe	CS106

In 1^{st} NF

1NF Constraint: Types must be atomic!

1st Normal Form (1NF)

DEPARTMENT

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

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

Normalization

1. Boyce-Codd Normal Form
2. Decompositions & 3NF
3. MVDs

Boyce-Codd Normal Form

Conceptual Design

Now that we know how to find FDs, it's a straight-forward process:

1. *Search for "bad" FDs*

2. *If there are any, then keep decomposing the table into sub-tables until no more bad FDs*

3. *When done, the database schema is normalized*

Recall: there are several normal forms…

Boyce-Codd Normal Form (BCNF)

- Main idea is that we define "good" and "bad" FDs as follows:
  - *$X \rightarrow A$ is a "good FD" if X is a (super)key*
    - In other words, if A is the set of all attributes
  - *$X \rightarrow A$ is a "bad FD" otherwise*
- We will try to eliminate the "bad" FDs!

Boyce-Codd Normal Form (BCNF)

- Why does this definition of "good" and "bad" FDs make sense?
- If X is *not* a (super)key, it functionally determines *some* of the attributes; therefore, those other attributes can be duplicated
  - *Recall: this means there is redundancy*
  - *And redundancy like this can lead to data anomalies!*

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

Boyce-Codd Normal Form

BCNF is a simple condition for removing anomalies from relations:

A relation R is **in BCNF** if:

if $\{\mathbf{A_1, ..., A_n}\} \rightarrow \mathbf{B}$ is a *non-trivial* FD in R

then $\{\mathbf{A_1, ..., A_n}\}$ **is a superkey** for R

Equivalently: $\forall$ sets of attributes X, either ($X^+ = X$) or (X^+ = all attributes)

In other words: there are no “bad” FDs

Example

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

{EmpID} → {Name,Phone,Position}

This FD is *good* because EmpID is a superkey

{Position} → {Phone}

This FD is *bad* because Position is **not** a superkey

⟹ **Not** in BCNF

What is the key?
{EmpID}

Example 2

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

{SSN} → {Name,City}

This FD is *bad* because it is **not** a superkey

⟹ **Not** in BCNF

What is the key?
{SSN, PhoneNumber}

Example 2

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Madison

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

{SSN} → {Name,City}

This FD is now *good* because it is the key

Let’s check anomalies:

- Redundancy ?
- Update ?
- Delete ?

Now in BCNF!

BCNF Decomposition Algorithm

Input: A relation R_0 and a set of FDs F.

1. Set D := $\{R_0\}$;
2. While there is a relation schema R in *D* that is not in BCNF do {

    Choose a ***R*** in ***D*** that is not in BCNF

    Find a FD ***X*** → ***Y*** in R that violates BCNF

    Decompose R in *D* by two relations R_1= $\mathbf{X^+}$ **and** $\mathbf{R_2}$**= (X U Z),** *where Z is the set of attributes not in* X^+

   }

Find a non-trivial "bad" FD X->Y, i.e. X is not a superkey

BCNF Decomposition Algorithm

Input: A relation R_0 and a set of FDs F.

1. Set D := $\{R_0\}$;
2. While there is a relation schema R in *D* that is not in BCNF

   do {

   Choose a ***R*** in ***D*** that is not in BCNF

   Find a FD ***X*** → ***Y*** in *R* that violates BCNF

   Decompose *R* in *D* by two relations R_1= $\boldsymbol{X^+}$ **and** $\mathbf{R_2}$**= (X U Z),** *where Z is the set of attributes not in* X^+

   }

$\boldsymbol{X^+}$ is set of the attributes that ***X functionally determines***

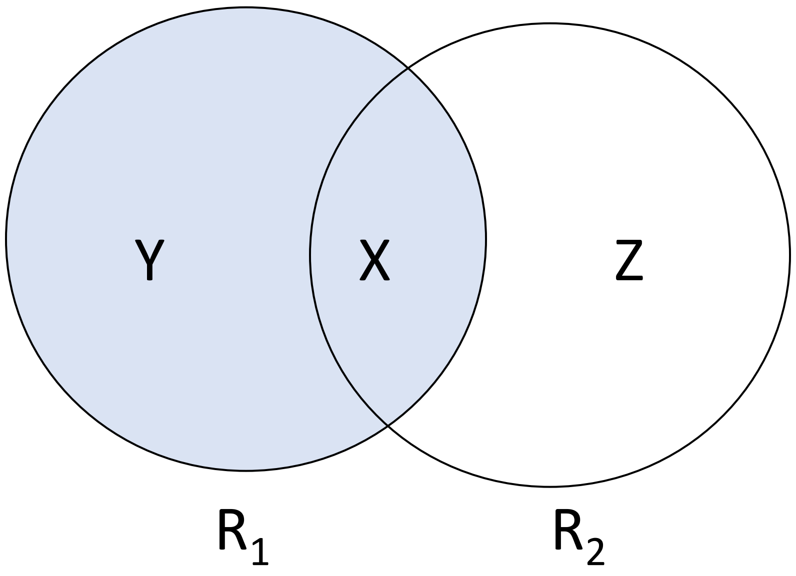
And Z is **set of attributes that it *doesn't***

BCNF Decomposition Algorithm

Input: A relation R_0 and a set of FDs F.

1. Set D := $\{R_0\}$;
2. While there is a relation schema R in *D* that is not in BCNF
   do {
   Choose a ***R*** in ***D*** that is not in BCNF
   Find a FD ***X*** → ***Y*** in *R* that violates BCNF
   Decompose *R* in *D* by two relations R_1= $\boldsymbol{X^+}$ **and** $\mathbf{R_2}$**= (X U Z),** *where Z is the set of attributes not in* X^+
   }

Split into one relation with X plus the attributes that X determines (i.e. Y)…



BCNF Decomposition Algorithm

Input: A relation R_o and a set of FDs F.

1. Set D := $\{R_o\}$;
2. While there is a relation schema R in *D* that is not in BCNF

   do {

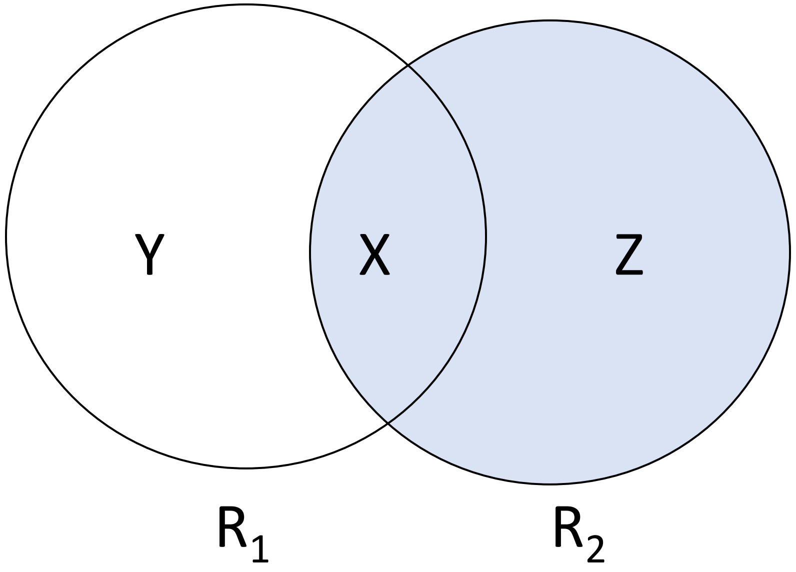
   Choose a ***R*** in ***D*** that is not in BCNF

   Find a FD $\boldsymbol{X \rightarrow Y}$ in R that violates BCNF

   Decompose R in *D* by two relations $R_1 = \boldsymbol{X^+}$ **and R_2 = (X U Z),** *where Z is the set of attributes not in X^+*

   }

And one relation with X plus the attributes X *does not* determine (i.e Z)



BCNF Decomposition Algorithm

Input: A relation R_0 and a set of FDs F.

1. Set D := $\{R_0\}$;
2. While there is a relation schema R in *D* that is not in BCNF
   do {
      Choose a ***R*** in ***D*** that is not in BCNF
      Find a FD $\boldsymbol{X \rightarrow Y}$ in *R* that violates BCNF
      Decompose *R* in *D* by two relations $R_1 = \boldsymbol{X^+}$
      and $\mathbf{R_2 = (X \cup Z)}$**,** *where Z is the set of attributes not in* X^+
   }

Proceed until no more "bad" FDs!

Example

Input: A relation R_0 and a set of FDs F.

1. Set D := $\{R_0\}$;
2. While there is a relation schema *R* in *D* that is not in BCNF
   do {
   Choose a ***R*** in ***D*** that is not in BCNF
   Find a FD ***X*** → ***Y*** in *R* that violates BCNF
   Decompose *R* in *D* by two relations R_1= ***X***$^+$ **and** $\mathbf{R_2}$**= (X U Z),** *where Z is the set of attributes not in X*$^+$
   }

R(A,B,C,D,E)

{A} → {B,C}
{C} → {D}

Key ??

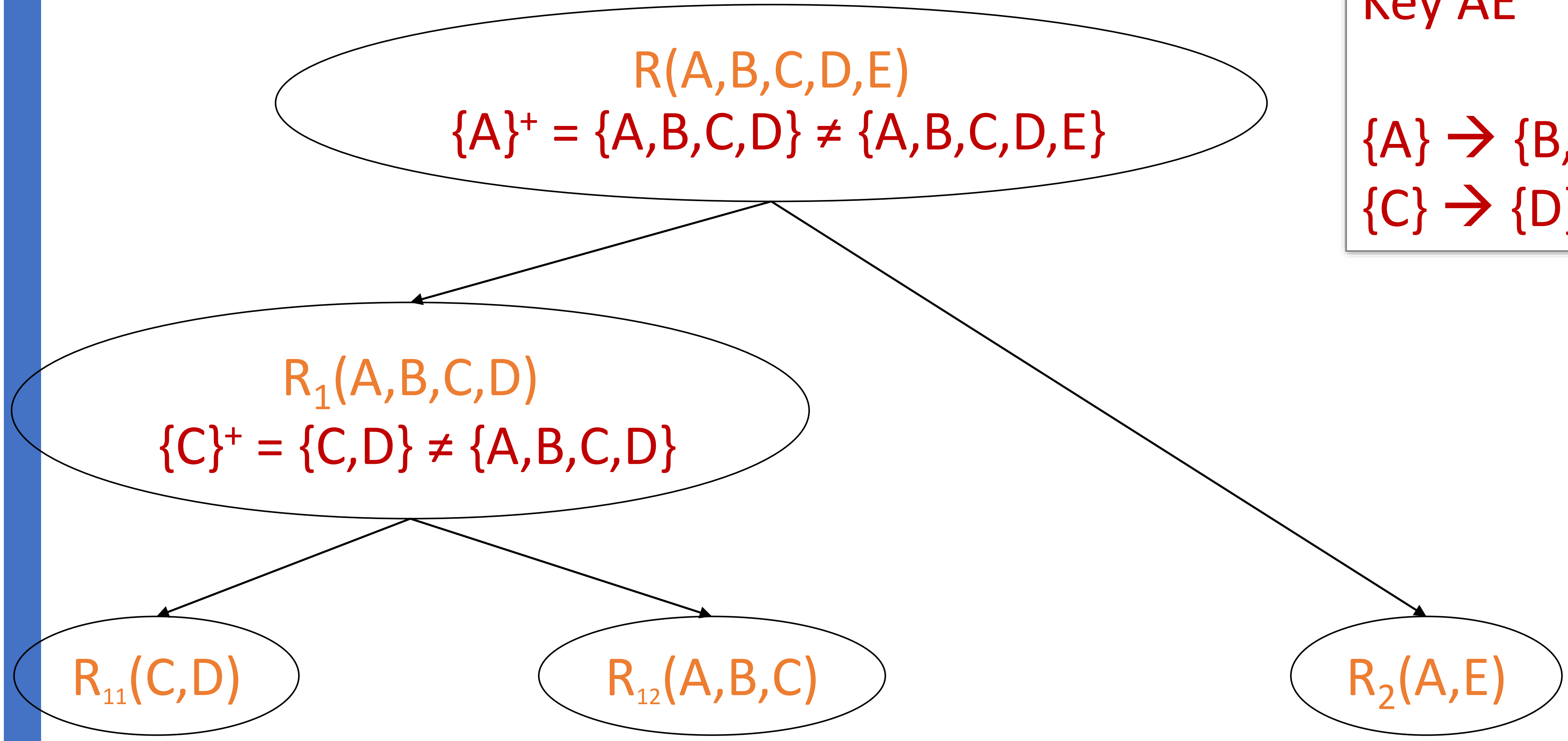
Example

R(A,B,C,D,E)

Key AE

$\{A\} \rightarrow \{B,C\}$

$\{C\} \rightarrow \{D\}$



ACTIVITY: BCNF

Consider the relation **Contracts(Cid, Sid, Pid, dept, part, qty)**

- *In this relation, the contract with Cid is an agreement that supplier Sid (supplierid) will supply Q items of Part to project Pid (projectid) associated with department Dept*
- FD's
  - *Cid is a key*
    - *Cid -> Cid, Sid, Pid, Dept, Part, Qty.*
  - *A project purchases a given part using a single contract*
    - Pid, Part -> *Cid.*
  - *A department purchases at most one part from a supplier*
    - *Sid, Dept -> Part.*
  - *Each project deals with a single supplier*
    - Pid -> Sid

Is the relation Contract in BCNF?

Key ?

1. Cid
2. Pid, Part
3. Pid, Dept

ACTIVITY: BCNF Decomposition

Consider the relation ***Contracts(Cid, Sid, Pid, dept, part, qty)***

- FD's
  - *Cid -> Cid, Sid, Pid, Dept, Part, Qty.*
  - *Pid, Part -> Cid.*
  - ***Sid, Dept -> Part.***
  - ***Pid -> Sid***

Lets take Sid, Dept -> Part

- **R2(Sid, Dept, Part)**
  - Now its in BCNF
- *Contracts(Cid, Sid, Pid, Dept, qty)*
  - *Still not in BCNF because of* Pid -> Sid
  - ***R3(Pid , Sid)***
  - ***Contracts(Cid, Pid, dept, qty)***

Another decomposition

Lets start with Pid -> Sid

- R2(Pid, Sid)
  - Now its in BCNF
- *Contracts(Cid, Pid, Dept, Part, qty)*
  - *In BCNF*
  - *But lost dependency* ***Sid, Dept -> Part***

Decompositions

Theory of dependencies can tell us

- about ***redundancy*** and
- give us clues about ***possible decompositions***

But it *cannot discriminate* between decomposition alternatives.

A designer has to consider the alternatives and choose one based on the semantics of the application