BCNF and Normalization

Dr. Bing Zhou

Relational Schema Design

- Goal of relational schema design is to avoid redundancy and anomalies.

- *Redundancy*:

Bad Design

name	*addr*	*beersLiked*	*manf*	*favBeer*
Janeway Janeway Spock	Voyager Voyager Enterprise	Export G.I. Lager Export	Molson Gr. Is. Molson	G.I. Lager G.I. Lager Export

- Redundancy

- Update anomaly
  - if Janeway is transferred to *Intrepid*, will we remember to change each of her tuples?

- Deletion anomaly
  - If nobody likes Export, we lose track of the fact that Molson manufactures Export.

Another Example

Number	DeptName	CourseName	Classroom	Enrollment	StudentName	Address
4604	CS	E-Business	211 McBryde	32	Adam	71 Main Street
6722	CS	Advanced DB	210 McBryde	15	Adam	71 Main Street
4322	Electrical	DB	220 McBryde	29	Suri	54 Elm Street
5722	CS	DB	311 Durham	34	Suri	54 Elm Street
5722	CS	DB	311 Durham	34	Joe	33 Astoria Ave
6722	CS	Advanced DB	210 McBryde	15	Joe	33 Astoria Ave

Relational Decomposition

- Accepted way to eliminate anomalies is to "decompose" relations.

Student(SSN, sName, address,
HScode, HSname, HScity, GPA, priority)

S1(SSN, sName, addr, HScode, GPA, priority)

S2(HScode, HSname, HScity)

$\bar{A} \cup \bar{B} = \bar{C}$ $\quad$ $S1 \bowtie S2 = \text{Student}$

```
Student(SSN, sName, address,
        HScode, HSname, HScity, GPA, priority)
```

S1 (SSN, sName, addr, HScode, HSname, HScity)

S2 (sName, HSname, GPA, priority)

$\bar{A} \cup \bar{B} = \bar{C}$

$S1 \bowtie S2 \stackrel{?}{=} \text{Student}$

Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- *trivial* if the B's are a subset of the A's,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$
- *non-trivial* if at least one B is not among the A's,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$
- *completely non-trivial* if none of the B's are among the A's, i.e.,
  $\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset$.
- *Trivial dependency rule*: The FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is equivalent to the FD $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$, where the C's are those B's that are not A's, i.e.,
  $\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}$.
- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.

Boyce-Codd Normal Form

- Condition on the FDs in a relation that guarantees that anomalies do not exist.

- A relation R is in *Boyce-Codd Normal Form* (BCNF) if and only if for every non-trivial FD $A_1A_2\ldots A_n \rightarrow B$ for R, $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R.
- Informally, the left side of every non-trivial FD must be a superkey.
- A relation R *violates* BCNF if it has an FD such that the attributes of the left side of an FD do not form a superkey.

Boyce-Codd Normal Form

Relation R with FDs is in BCNF if:

For each $\bar{A} \rightarrow B$, $\bar{A}$ is a Super key

BCNF violation

$\bar{A}$	B	rest
a	b	—
a	b	—
	⋮	

Student(SSN, sName, address,
HScode, HSname, HScity, GPA, priority)

✓ SSN → sName, address, GPA

✓ GPA → priority

✓ HScode → HSname, HScity

Keys:
{SSN, HScode}

Every FD have a Key on LHS?

Apply(SSN, cName, state, date, major)

$\underline{\text{SSN, cName, state}} \rightarrow \text{\underline{date}, \underline{major}}$

Key

In BCNF.

Checking for BCNF Violations

- List all FDs.
- Ensure that left hand side of each FD is a superkey.
- We have to first find all the keys!
- Is Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address) in BCNF?
- FDs are

  Number DepartmentName $\longrightarrow$ CourseName
  Number DepartmentName $\longrightarrow$ Classroom
  Number DepartmentName $\longrightarrow$ Enrollment
- What is $\{\texttt{Number}, \texttt{DepartmentName}\}^+$?

{Number, DepartmentName, Coursename, Classroom, Enrollment}.

- Therefore, the key is {Number, DepartmentName, StudentName, Address}
- The relation is not in BCNF.

Decomposition into BCNF

- Suppose R is a relation schema that violates BCNF.
- We can decompose R into a set S of new relations such that
  1. each relation in S is in BCNF and
  2. we can "recover" R from the relations in S, i.e., the relations in S "faithfully" represent the data in R.
- Let X be the set of all attributes of R.
- Suppose the FD $A_1A_2 \dots A_m \to B$ violates BCNF.
- Decomposition algorithm:
  1. Compute $\{A_1A_2 \dots, A_m\}^+$ and augment the FD to $A_1A_2 \dots A_m \to \{A_1, A_2 \dots, A_m\}^+$.
  2. Decompose R into two relations containing
     2.1 all the attributes in $\{A_1, A_2 \dots, A_m\}^+$
     2.2 all the attributes on the left side of the FD and all the attributes of R not on the right side of the FD, i.e., $X - \{A_1, A_2 \dots, A_m\}^+ \cup \{A_1, A_2 \dots, A_m\}$.
  3. Find FDs in the new relations and decompose them if they are not in BCNF.

Decomposing `Courses`

- Schema is `Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address)`.
- BCNF-violating FD is
  `Number DepartmentName` $\rightarrow$ `CourseName Classroom Enrollment`.
  - What is $\{$`Number, DepartmentName`$\}^+$?

$\{$`Number, DepartmentName, Coursename, Classroom, Enrollment`$\}$.

- Decompose `Courses` into
  `Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment)` and
  `Courses2(Number, DepartmentName, StudentName, Address)`.

Decomposing `Courses`

- Decompose `Courses` into

  `Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment)` and

  `Courses2(Number, DepartmentName, StudentName, Address)`.

Number	DeptName	CourseName	Classroom	Enrollment
4604	CS	E-Business	211 McBryde	32
6722	CS	Advanced DB	210 McBryde	15
4322	Electrical	DB	220 McBryde	29
5722	CS	DB	311 Durham	34

Number	DeptName	StudentName	Address
4604	CS	Adam	71 Main Street
6722	CS	Adam	71 Main Street
4322	Electrical	Suri	54 Elm Street
5722	CS	Suri	54 Elm Street
5722	CS	Joe	33 Astoria Ave
6722	CS	Joe	33 Astoria Ave

- Are there any BCNF violations in the two new relations?

Another Example of Decomposition

- Schema is `Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)`
- What are the FDs?

  `ID` $\longrightarrow$ `Name FavouriteAdvisorId`

  `AdvisorId` $\longrightarrow$ `AdvisorName`
- What is the key? {`ID, AdvisorId`}
- Is there a BCNF violation? Yes.
- Use `ID` $\longrightarrow$ `Name Level FavouriteAdvisorId` to decompose.
  - {`ID`}$^+$ is {`ID, Name, FavouriteAdvisorId`}
  - Schemas for new relations are

    `Students1(ID, Name, FavouriteAdvisorId)`

    `Students2(ID, AdvisorId, AdvisorName)`

Another Example of Decomposition (2)

- What are the FDs in Student1(ID, Name, FavouriteAdvisorId)? There are none that violate BCNF
- What are the FDs in Students2(ID, AdvisorId, AdvisorName)?
  - AdvisorId $\longrightarrow$ AdvisorName
- Repeat the decomposition process.
- Use AdvisorId $\longrightarrow$ AdvisorName to decompose.
  - $\{\texttt{AdvisorId}\}^+$ is

    {AdvisorId, AdvisorName}
  - Schemas for new relations are

    Students2(ID, AdvisorId)

    Students3(AdvisorId, AdvisorName)

Student(SSN, sName, address, HScode, HSname, HScity, GPA, priority)

SSN → sName, address, GPA   GPA → priority

HScode → HSname, HScity

Key: {SSN, HScode}

S1(HScode, HSname, HScity)

~~S2(SSN, sName, addr, HScode, GPA, priority)~~

S3(GPA, priority)

~~S4(SSN, sName, addr, HScode, GPA)~~

S5(SSN, sName, addr, GPA)

S6(SSN, HScode)

BCNFs and Two-Attribute Relationships

- True or False: Every two-attribute relation $R(A, B)$ is in BCNF.
- The statement is true. Why?
- Consider four possible cases:
  1. There are no non-trivial FDs.
  2. $A \rightarrow B$ is the only non-trivial FD.
  3. $B \rightarrow A$ is the only non-trivial FD.
  4. Both $A \rightarrow B$ and $B \rightarrow A$ hold in R.

Decomposition into BCNF

- Suppose R is a relation schema that violates BCNF.
- We can decompose R into a set S of new relations such that
  1. each relation in S is in BCNF and
  2. we can "recover" R from the relations in S, i.e., the relations in S "faithfully" represent the data in R.
- How does the normalisation algorithm guarantee the second condition?

Candidate Normalization Algorithm

- Every two-attribute relation is in BCNF.
- Can we bring any relation R into BCNF by arbitrarily decomposing it into two-attribute relations?
- No, since we may not be able to recover R correctly from the decomposition.

Joining Relations

R

A	B
a_1	b_1
a_2	b_1
a_2	b_2

$\bowtie$

S

B	C
b_1	c_1
b_2	c_2
b_2	c_3

$=$

- Let R and S be two relations with one common attribute B.
- Relation T is the *join* of R and S, denoted $R \bowtie S$ if and only if
  - the attributes of T are the union of the attributes of R and S,
  - every tuple $t \in T$ is the *join* of two tuples $r \in R$ and $s \in S$ that agree on the attribute B, i.e., t agrees with r on all the attributes in R and with s on all attributes in S,
  - T contains all tuples formed in this manner.

Recovering Information from a Decomposition

- Suppose R is a relation schema that violates BCNF.
- We can decompose R into a set $\{S_1, S_2, \ldots S_k\}$ of new relations such that
  1. each relation $S_i, 1 \leq i \leq k$ is in BCNF and
  2. we can "recover" R from these relations: $R = S_1 \bowtie S_2 \bowtie \ldots \bowtie S_k$, i.e., the decomposition of R into $\{S_1, S_2, \ldots S_k\}$ is a *lossless-join* decomposition.

Correct Decompositions

A decomposition is *lossless* if we can recover:

R(A,B,C)

Decompose

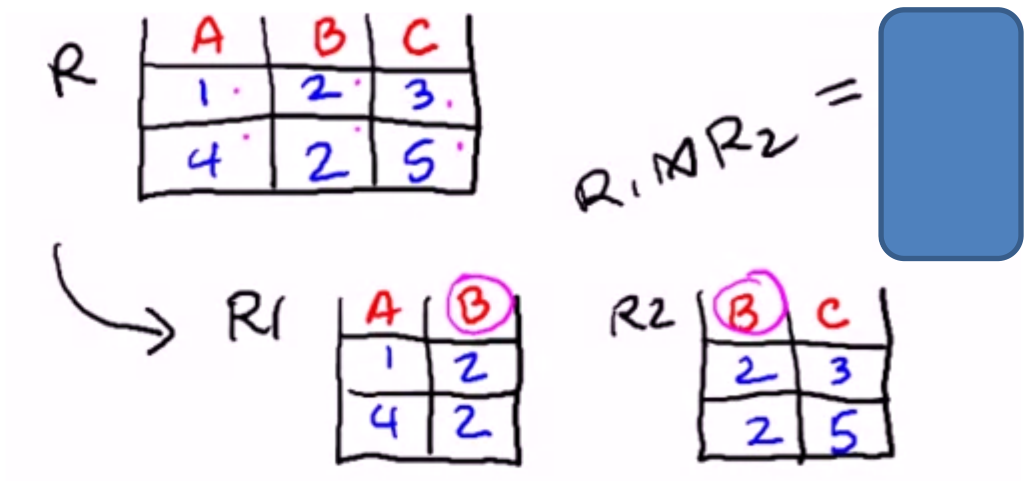
R1(A,B)　　R2(A,C)

Recover

R'(A,B,C)　should be the same as R(A,B,C)

R' is in general larger than R. Must ensure R' = R

Incorrect Decompositions



R’ is in general larger than R. Must ensure R’ = R

Example of Lossy-Join Decomposition

- Example: Decomposition of *R = (A, B)*

  $R_1 = (A) \quad R_2 = (B)$

A	B
α	1
α	2
β	1

r

A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_{B(r)}$

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B
α	1
α	2
β	1
β	2

Example: BCNF Decomposition

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

FDs = name->addr, name -> favBeer, beersLiked->manf

- Pick BCNF violation name->addr.
- Close the left side: $\{name\}^+ = \{name, addr, favBeer\}$.
- Decomposed relations:
  1. Drinkers1(<u>name</u>, addr, favBeer)
  2. Drinkers2(<u>name</u>, <u>beersLiked</u>, manf)

Example -- Continued

- We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.
- Is Drinkers1 in BCNF ?
  - For Drinkers1(name, addr, favBeer), relevant FD's are name->addr and name->favBeer.
  - Thus, {name} is the only key and Drinkers1 is in BCNF.

Example -- Continued

- For Drinkers2(name, beersLiked, manf), the only FD is beersLiked->manf, and the only key is

  {name, beersLiked}.

  – Violation of BCNF ?
- $beersLiked^+$ = {beersLiked, manf}, so we decompose *Drinkers2* into:

1. Drinkers3(<u>beersLiked</u>, manf)
2. Drinkers4(<u>name</u>, <u>beersLiked</u>)

Example -- Concluded

- The resulting decomposition of *Drinkers* :
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)

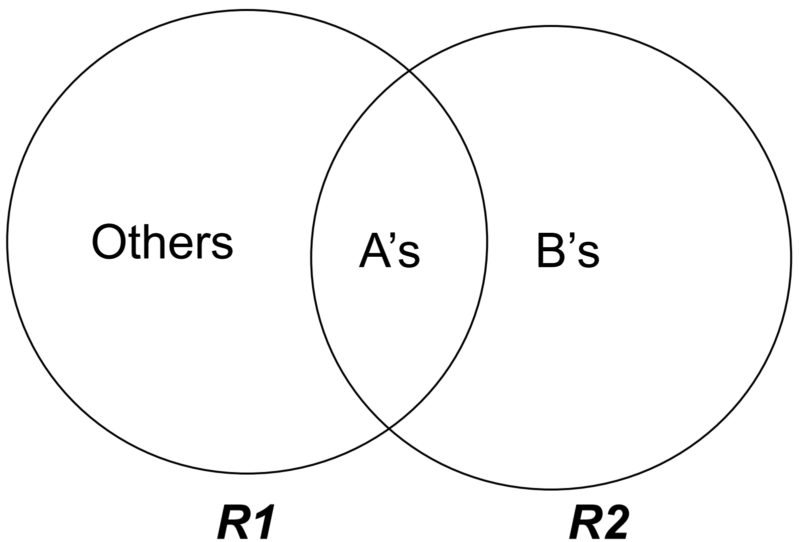
- Note:
  - *Drinkers1* tells us about drinkers,
  - *Drinkers3* tells us about beers, and
  - *Drinkers4* tells us the relationship between drinkers and the beers they like.

Summary of BCNF Decomposition

Find a dependency that violates the BCNF condition:

$$A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$$

Decompose:



Is there a 2-attribute relation that is not in BCNF ?

Continue until there are no BCNF violations left.

Relational design by decomposition

Relational design by decomposition

- "Mega" relations + properties of the data
- System decomposes based on properties
- Final set of relations satisfies normal form
  - No anomalies, no lost information
- Functional dependencies $\Rightarrow$ Boyce-Codd Normal Form
- Multivalued dependences $\Rightarrow$ Fourth Normal Form