Database Design and Normal Forms

Database Design
- coming up with a ‘good’ schema is very important

How do we characterize the “goodness” of a schema?
If two or more alternative schemas are available
  how do we compare them?
What are the problems with “bad” schema designs?

Normal Forms:
  Each normal form specifies certain conditions
  If the conditions are satisfied by the schema
    certain kind of problems are avoided

Details follow....
An Example

*student* relation with attributes: studName, rollNo, sex, studDept
*department* relation with attributes: deptName, officePhone, hod

Several students belong to a department.
studDept gives the name of the student’s department.

Correct schema:

<table>
<thead>
<tr>
<th>Student</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>studName</td>
<td>rollNo</td>
</tr>
</tbody>
</table>

Incorrect schema:

<table>
<thead>
<tr>
<th>Student Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>studName</td>
</tr>
</tbody>
</table>

What are the problems that arise?
Problems with bad schema

Redundant storage of data:
   Office Phone & HOD info - stored redundantly
   - once with each student that belongs to the department
   - wastage of disk space

A program that updates Office Phone of a department
   - must change it at several places
     - more running time
     - error - prone

Transactions running on a database
   - must take as short time as possible to increase transaction throughput
Update Anomalies

Another kind of problem with bad schema

Insertion anomaly:
No way of inserting info about a new department unless
we also enter details of a (dummy) student in the department

Deletion anomaly:
If all students of a certain department leave
and we delete their tuples,
information about the department itself is lost

Update Anomaly:
Updating officePhone of a department
• value in several tuples needs to be changed
• if a tuple is missed - inconsistency in data
# Normal Forms

First Normal Form (1NF) - included in the definition of a relation

Second Normal Form (2NF) - defined in terms of functional dependencies

Third Normal Form (3NF) - defined in terms of functional dependencies

Boyce-Codd Normal Form (BCNF)

Fourth Normal Form (4NF) - defined using multivalued dependencies

Fifth Normal Form (5NF) or Project Join Normal Form (PJNF) - defined using join dependencies
Functional Dependencies

A functional dependency (FD) $X \rightarrow Y$ (read as $X$ determines $Y$) ($X \subseteq R$, $Y \subseteq R$) is said to hold on a schema $R$ if in any instance $r$ on $R$, if two tuples $t_1$, $t_2$ ($t_1 \neq t_2$, $t_1 \in r$, $t_2 \in r$) agree on $X$ i.e. $t_1[X] = t_2[X]$ then they also agree on $Y$ i.e. $t_1[Y] = t_2[Y]$

Note: If $K \subset R$ is a key for $R$ then for any $A \in R$, $K \rightarrow A$ holds because the above if .....then condition is vacuously true
Functional Dependencies – Examples

Consider the schema:

   Student (studName, rollNo, sex, dept, hostelName, roomNo)

Since rollNo is a key, rollNo → {studName, sex, dept, hostelName, roomNo}

Suppose that each student is given a hostel room exclusively, then

   hostelName, roomNo → rollNo

Suppose boys and girls are accommodated in separate hostels, then

   hostelName → sex

FDs are additional constraints that can be specified by designers
Trivial / Non-Trivial FDs

An FD $X \rightarrow Y$ where $Y \subseteq X$
   - called a *trivial* FD, it always holds good

An FD $X \rightarrow Y$ where $Y \not\subseteq X$
   - *non-trivial* FD

An FD $X \rightarrow Y$ where $X \cap Y = \emptyset$
   - *completely non-trivial* FD
Deriving new FDs

Given that a set of FDs F holds on R
we can infer that a certain new FD must also hold on R

For instance,
given that $X \rightarrow Y$, $Y \rightarrow Z$ hold on R
we can infer that $X \rightarrow Z$ must also hold

How to systematically obtain all such new FDs?

Unless *all* FDs are known, a relation schema is not fully specified
Entailment relation

We say that a set of FDs $F \models \{ X \rightarrow Y \}$
(read as $F$ entails $X \rightarrow Y$ or $F$ logically implies $X \rightarrow Y$)
if in every instance $r$ of $R$ on which FDs $F$ hold,
FD $X \rightarrow Y$ also holds.

Armstrong came up with several inference rules for deriving new FDs from a given set of FDs

We define $F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}$
$F^+$: Closure of $F$
Armstrong’s Inference Rules (1/2)

1. Reflexive rule
   \[ F \models \{ X \rightarrow Y \mid Y \subseteq X \} \text{ for any } X. \text{ Trivial FDs} \]

2. Augmentation rule
   \[ \{ X \rightarrow Y \} \models \{ XZ \rightarrow YZ \}, \ Z \subseteq R. \text{ Here } XZ \text{ denotes } X \cup Z \]

3. Transitive rule
   \[ \{ X \rightarrow Y, Y \rightarrow Z \} \models \{ X \rightarrow Z \} \]

4. Decomposition or Projective rule
   \[ \{ X \rightarrow YZ \} \models \{ X \rightarrow Y \} \]

5. Union or Additive rule
   \[ \{ X \rightarrow Y, X \rightarrow Z \} \models \{ X \rightarrow YZ \} \]

6. Pseudo transitive rule
   \[ \{ X \rightarrow Y, WY \rightarrow Z \} \models \{ WX \rightarrow Z \} \]
Armstrong's Inference Rules (2/2)

Rules 4, 5, 6 are not really necessary.

For instance, Rule 5: \( \{X \rightarrow Y, X \rightarrow Z\} \models \{X \rightarrow YZ\} \) can be proved using 1, 2, 3 alone

1) \( X \rightarrow Y \) \quad \text{given}
2) \( X \rightarrow Z \) \quad \text{given}
3) \( X \rightarrow XY \) Augmentation rule on 1
4) \( XY \rightarrow ZY \) Augmentation rule on 2
5) \( X \rightarrow ZY \) Transitive rule on 3, 4.

Similarly, 4, 6 can be shown to be unnecessary.
But it is useful to have 4, 5, 6 as short-cut rules
Sound and Complete Inference Rules

Armstrong showed that Rules (1), (2) and (3) are sound and complete. These are called Armstrong’s Axioms (AA)

Soundness:
Every new FD \( X \rightarrow Y \) derived from a given set of FDs \( F \) using Armstrong's Axioms is such that \( F \models \{X \rightarrow Y\} \)

Completeness:
Any FD \( X \rightarrow Y \) logically implied by \( F \) (i.e. \( F \models \{X \rightarrow Y\} \)) can be derived from \( F \) using Armstrong’s Axioms
Proving Soundness
Suppose $X \rightarrow Y$ is derived from $F$ using AA in some $n$ steps. If each step is correct then overall deduction would be correct.

Single step: Apply Rule (1) or (2) or (3)

Rule (1) – obviously results in correct FDs

Rule (2) – $\{X \rightarrow Y\} \models \{XZ \rightarrow YZ\}$, $Z \subseteq R$

Suppose $t_1, t_2 \in r$ agree on $XZ$

$\Rightarrow t_1, t_2$ agree on $X$

$\Rightarrow t_1, t_2$ agree on $Y$ (since $X \rightarrow Y$ holds on $r$)

$\Rightarrow t_1, t_2$ agree as $YZ$

Hence Rule (2) gives rise to correct FDs

Rule (3) – $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

Suppose $t_1, t_2 \in r$ agree on $X$

$\Rightarrow t_1, t_2$ agree on $Y$ (since $X \rightarrow Y$ holds)

$\Rightarrow t_1, t_2$ agree on $Z$ (since $Y \rightarrow Z$ holds)
Proving Completeness of Armstrong’s Axioms (1/4)

Define $X^+_F$ (closure of $X$ wrt F)
$$ = \{ A \mid X \rightarrow A \text{ can be derived from F using AA} \}, \ A \in R $$

Claim 1:
$X \rightarrow Y$ can be derived from F using AA iff $Y \subseteq X^+$

(If) Let $Y = \{ A_1, A_2, \ldots, A_n \}$. $Y \subseteq X^+$
$$ \Rightarrow X \rightarrow A_i \text{ can be derived from F using AA (} 1 \leq i \leq n) $$
By union rule, it follows that $X \rightarrow Y$ can be derived from F.

(Only If) $X \rightarrow Y$ can be derived from F using AA
By projective rule $X \rightarrow A_i \ (1 \leq i \leq n)$
Thus by definition of $X^+$, $A_i \in X^+$
$$ \Rightarrow Y \subseteq X^+ $$
Completeness of Armstrong’s Axioms (2/4)

Completeness:

\[(F \models \{X \rightarrow Y\}) \Rightarrow X \rightarrow Y \text{ follows from F using AA}\]

We will prove the contrapositive:

\[X \rightarrow Y \text{ can’t be derived from F using AA}\]

\[\Rightarrow F \not\models \{X \rightarrow Y\}\]

\[\Rightarrow \exists \text{ a relation instance } r \text{ on } R \text{ st all the FDs of F hold on } r \text{ but } X \rightarrow Y \text{ doesn’t hold.}\]

Consider the relation instance \( r \) with just two tuples:

\[X^+ \text{ attributes } \quad \text{Other attributes}\]

\[r:\quad 1 \quad 1 \quad 1 \ldots 1 \quad 1 \quad 1 \quad 1 \ldots 1\]
\[1 \quad 1 \quad 1 \ldots 1 \quad 0 \quad 0 \quad 0 \ldots 0\]
Claim 2: All FDs of $F$ are satisfied by $r$

Suppose not. Let $W \rightarrow Z$ in $F$ be an FD not satisfied by $r$
Then $W \subseteq X^+$ and $Z \not\subseteq X^+$
Let $A \in Z - X^+$

Now, $X \rightarrow W$ follows from $F$ using AA as $W \subseteq X^+$ (claim 1)

$X \rightarrow Z$ follows from $F$ using AA by transitive rule

$Z \rightarrow A$ follows from $F$ using AA by reflexive rule as $A \in Z$

$X \rightarrow A$ follows from $F$ using AA by transitive rule

By definition of closures, $A$ must belong to $X^+$
- a contradiction.

Hence the claim.
Completeness Proof (4/4)

Claim 3: $X \rightarrow Y$ is not satisfied by $r$

Suppose not

Because of the structure of $r$, $Y \subseteq X^+$

$\Rightarrow X \rightarrow Y$ can be derived from $F$ using AA

contradicting the assumption about $X \rightarrow Y$

Hence the claim

Thus, whenever $X \rightarrow Y$ doesn’t follow from $F$ using AA,

$F$ doesn’t logically imply $X \rightarrow Y$

Armstrong’s Axioms are complete.
Consequence of Completeness of AA

\[ X^+ = \{ A \mid X \rightarrow A \text{ follows from } F \text{ using AA} \} \]
\[ = \{ A \mid F \models X \rightarrow A \} \]

Similarly

\[ F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \} \]
\[ = \{ X \rightarrow Y \mid X \rightarrow Y \text{ follows from } F \text{ using AA} \} \]
Computing closures

The size of $F^+$ can sometimes be exponential in the size of $F$. For instance, $F = \{A \rightarrow B_1, A \rightarrow B_2, \ldots, A \rightarrow B_n\}$

$F^+ = \{A \rightarrow X\}$ where $X \subseteq \{B_1, B_2, \ldots, B_n\}$.

Thus $|F^+| = 2^n$

Computing $F^+$: computationally expensive

Fortunately, checking if $X \rightarrow Y \in F^+$

can be done by checking if $Y \subseteq X^+_F$

Computing attribute closure ($X^+_F$) is easier
Computing $X^+_F$

We compute a sequence of sets $X_0, X_1, \ldots$ as follows:

\[
X_0 := X; \quad // X \text{ is the given set of attributes}
\]
\[
X_{i+1} := X_i \cup \{A \mid \text{there is a FD } Y \rightarrow Z \text{ in } F \text{ and } A \in Z \text{ and } Y \subseteq X_i\}
\]

Since $X_0 \subseteq X_1 \subseteq X_2 \subseteq \ldots \subseteq X_i \subseteq X_{i+1} \subseteq \ldots \subseteq R$

and $R$ is finite,

There is an integer $i$ s.t $X_i = X_{i+1} = X_{i+2} = \ldots$

and $X^+_F$ is equal to $X_i$. 
Normal Forms – 2NF

Full functional dependency:
   An FD $X \rightarrow A$ for which there is no proper subset $Y$ of $X$
   such that $Y \rightarrow A$
   (A is said to be fully functionally dependent on $X$)

2NF: A relation schema $R$ is in 2NF if
   every non-prime attribute is fully functionally dependent on any
   key of $R$

prime attribute: A attribute that is part of some key
non-prime attribute: An attribute that is not part of any key
Example

1) Book (authorName, title, authorAffiliation, ISBN, publisher, pubYear )
   Keys: (authorName, title), ISBN
   Not in 2NF as authorName → authorAffiliation
   (authorAffiliation is not fully functionally dependent on the first key)
2) Student (rollNo, name, dept, sex, hostelName, roomNo, admitYear)
   Keys: rollNo, (hostelName, roomNo)
   Not in 2NF as hostelName → sex

   student (rollNo, name, dept, hostelName, roomNo, admitYear)
   hostelDetail (hostelName, sex)
   - There are both in 2NF
Transitive Dependencies

Transitive dependency:
An FD \( X \rightarrow Y \) in a relation schema \( R \) for which there is a set of attributes \( Z \subseteq R \) such that
\[
X \rightarrow Z \text{ and } Z \rightarrow Y \text{ and } Z \text{ is not a subset of any key of } R
\]

Ex: student (rollNo, name, dept, hostelName, roomNo, headDept)
Keys: rollNo, (hostelName, roomNo)
rollNo \( \rightarrow \) dept; dept \( \rightarrow \) headDept hold
So, rollNo \( \rightarrow \) headDept a transitive dependency

Head of the dept of dept D is stored redundantly in every tuple where D appears.
Relation is in 2NF but redundancy still exists.
Normal Forms – 3NF

Relation schema R is in 3NF if it is in 2NF and no non-prime attribute of R is transitively dependent on any key of R

student (rollNo, name, dept, hostname, roomNo, headDept) is not in 3NF

Decompose:  
student (rollNo, name, dept, hostname, roomNo)  
depInfo (dept, headDept)
both in 3NF

Redundancy in data storage - removed
Another definition of 3NF

Relation schema R is in 3NF if for any nontrivial FD \( X \rightarrow A \) either (i) \( X \) is a superkey or (ii) \( A \) is prime.

Suppose some \( R \) violates the above definition
⇒ There is an FD \( X \rightarrow A \) for which both (i) and (ii) are false
⇒ \( X \) is not a superkey and \( A \) is non-prime attribute

Two cases arise:
1) \( X \) is contained in a key – \( A \) is not fully functionally dependent on this key
   - violation of 2NF condition
2) \( X \) is not contained in a key
   \( K \rightarrow X, X \rightarrow A \) is a case of transitive dependency
   \( (K – \text{any key of } R) \)
Motivating example for BCNF

gradeInfo (rollNo, studName, course, grade)

Suppose the following FDs hold:
1) rollNo, course → grade
2) studName, course → grade
3) rollNo → studName
4) studName → rollNo

For 1,2 lhs is a key. For 3,4 rhs is prime
So gradeInfo is in 3NF

But studName is stored redundantly along with every course
being done by the student
Boyce - Codd Normal Form (BCNF)

Relation schema R is in BCNF if for every nontrivial FD $X \rightarrow A$, $X$ is a *superkey* of R.

In gradeInfo, FDs 3, 4 are nontrivial but lhs is not a superkey.
So, gradeInfo is not in BCNF

Decompose:

- gradeInfo (rollNo, course, grade)
- studInfo (rollNo, studName)

Redundancy allowed by 3NF is disallowed by BCNF

BCNF is stricter than 3NF
3NF is stricter than 2NF
Decomposition of a relation schema

If R doesn’t satisfy a particular normal form, we decompose R into smaller schemas.

What’s a decomposition?

\[ R = (A_1, A_2, \ldots, A_n) \]
\[ D = (R_1, R_2, \ldots, R_k) \text{ st } R_i \subseteq R \text{ and } R = R_1 \cup R_2 \cup \ldots \cup R_k \]

(Ri’s need not be disjoint)

Replacing R by R₁, R₂, …, Rₖ – process of decomposing R

Ex: gradeInfo (rollNo, studName, course, grade)
   R₁: gradeInfo (rollNo, course, grade)
   R₂: studInfo (rollNo, studName)
Desirable Properties of Decompositions

Not all decomposition of a schema are useful

We require two properties to be satisfied

(i) Lossless join property
   - the information in an instance \( r \) of \( R \) must be preserved in the instances \( r_1, r_2, \ldots, r_k \) where \( r_i = \pi_{R_i}(r) \)

(ii) Dependency preserving property
   - if a set \( F \) of dependencies hold on \( R \) it should be possible to enforce \( F \) by enforcing appropriate dependencies on each \( r_i \)
Lossless join property

F – set of FDs that hold on R
R – decomposed into R₁, R₂,…,Rₖ
Decomposition is lossless wrt F if
for every relation instance r on R satisfying F,
\[ r = \pi_{R₁}(r) \ast \pi_{R₂}(r) \ast \ldots \ast \pi_{Rₖ}(r) \]

R = (A, B, C); R₁ = (A, B); R₂ = (B, C)

<table>
<thead>
<tr>
<th>r</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b₁</td>
<td>c₃</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r₁</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b₁</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r₂</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>b₂</td>
<td>c₂</td>
<td></td>
</tr>
<tr>
<td>b₁</td>
<td>c₃</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r₁ \ast r₂</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>a₁</td>
<td>b₁</td>
<td>c₃</td>
<td></td>
</tr>
<tr>
<td>a₂</td>
<td>b₂</td>
<td>c₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b₁</td>
<td>c₁</td>
<td></td>
</tr>
</tbody>
</table>

Lossless joins are also called non-additive joins

Original info is distorted

Spurious tuples
Dependency Preserving Decompositions

Decomposition D = (R₁, R₂,…,Rₖ) of schema R preserves a set of dependencies F if

\[(\pi_{R_1}(F) \cup \pi_{R_2}(F) \cup \ldots \cup \pi_{R_k}(F))^+ = F^+\]

Here, \(\pi_{R_i}(F) = \{ (X \rightarrow Y) \in F^+ | X \subseteq R_i, Y \subseteq R_i \}\)
(called projection of F onto Rᵢ)

Informally, any FD that logically follows from F must also logically follow from the union of projections of F onto Rᵢ’s.

Then, D is called dependency preserving.
An example

Schema $R = (A, B, C)$
FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

Decomposition $D = (R_1 = \{A, B\}, R_2 = \{B, C\})$

$\pi_{R_1}(F) = \{A \rightarrow B, B \rightarrow A\}$

$\pi_{R_2}(F) = \{B \rightarrow C, C \rightarrow B\}$

$\left(\pi_{R_1}(F) \cup \pi_{R_2}(F)\right)^+ = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B, A \rightarrow C, C \rightarrow A\} = F^+$

Hence Dependency preserving
Testing for lossless decomposition property (1/6)

R – given schema with attributes $A_1, A_2, \ldots, A_n$
F – given set of FDs
D – $\{R_1, R_2, \ldots, R_m\}$ given decomposition of R

Is D a lossless decomposition?

Create an $m \times n$ matrix $S$ with columns labeled as $A_1, A_2, \ldots, A_n$
and rows labeled as $R_1, R_2, \ldots, R_m$

Initialize the matrix as follows:
- set $S(i,j)$ as symbol $b_{ij}$ for all $i,j$.
- if $A_j$ is in the scheme $R_i$, then set $S(i,j)$ as symbol $a_j$, for all $i,j$
Testing for lossless decomposition property (2/6)

After $S$ is initialized, we carry out the following process on it:

repeat
   for each functional dependency $U \rightarrow V$ in $F$ do
      for all rows in $S$ which agree on $U$-attributes do
         make the symbols in each $V$-attribute column the same in all the rows as follows:
            if any of the rows has an “$a$” symbol for the column
               set the other rows to the same “$a$” symbol in the column
            else // if no “$a$” symbol exists in any of the rows
               choose one of the “$b$” symbols that appears in one of the rows for the $V$-attribute and
               set the other rows to that “$b$” symbol in the column
   until no changes to $S$

At the end, if there exists a row with all “$a$” symbols then $D$ is lossless otherwise $D$ is a lossy decomposition
Testing for lossless decomposition property (3/6)

R = (rollNo, name, advisor, advisorDept, course, grade)

FD’s = { rollNo → name; rollNo → advisor; advisor → advisorDept 
          rollNo, course → grade }

D : { R₁ = (rollNo, name, advisor), R₂ = (advisor, advisorDept),
      R₃ = (rollNo, course, grade) }

Matrix S : (Initial values)

<table>
<thead>
<tr>
<th></th>
<th>rollNo</th>
<th>name</th>
<th>advisor</th>
<th>advisor Dept</th>
<th>course</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>a₁</td>
<td>a₂</td>
<td>a₃</td>
<td>b₁₄</td>
<td>b₁₅</td>
<td>b₁₆</td>
</tr>
<tr>
<td>R₂</td>
<td>b₂₁</td>
<td>b₂₂</td>
<td>a₃</td>
<td>a₄</td>
<td>b₂₅</td>
<td>b₂₆</td>
</tr>
<tr>
<td>R₃</td>
<td>a₁</td>
<td>b₃₂</td>
<td>b₃₃</td>
<td>b₃₄</td>
<td>a₅</td>
<td>a₆</td>
</tr>
</tbody>
</table>
Testing for lossless decomposition property (4/6)

R = (rollNo, name, advisor, advisorDept, course, grade)
FD’s = { rollNo → name; rollNo → advisor; advisor → advisorDept
        rollNo, course → grade }
D : { R_1 = (rollNo, name, advisor), R_2 = (advisor, advisorDept),
        R_3 = (rollNo, course, grade) }

Matrix S : (After enforcing rollNo → name & rollNo → advisor)

<table>
<thead>
<tr>
<th></th>
<th>rollNo</th>
<th>name</th>
<th>advisor</th>
<th>advisor Dept</th>
<th>course</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>a_1</td>
<td>a_2</td>
<td>a_3</td>
<td>b_{14}</td>
<td>b_{15}</td>
<td>b_{16}</td>
</tr>
<tr>
<td>R_2</td>
<td>b_{21}</td>
<td>b_{22}</td>
<td>a_3</td>
<td>a_4</td>
<td>b_{25}</td>
<td>b_{26}</td>
</tr>
<tr>
<td>R_3</td>
<td>a_1</td>
<td>b_{32}</td>
<td>a_2</td>
<td>b_{34}</td>
<td>a_5</td>
<td>a_6</td>
</tr>
</tbody>
</table>
Testing for lossless decomposition property (5/6)

R = (rollNo, name, advisor, advisorDept, course, grade)

FD’s = { rollNo → name; rollNo → advisor; advisor → advisorDept
           rollNo, course → grade }

D : { R₁ = (rollNo, name, advisor), R₂ = (advisor, advisorDept),
       R₃ = (rollNo, course, grade) }

Matrix S : (After enforcing advisor → advisorDept )

<table>
<thead>
<tr>
<th></th>
<th>rollNo</th>
<th>name</th>
<th>advisor</th>
<th>advisor Dept</th>
<th>course</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>a₁</td>
<td>a₂</td>
<td>a₃</td>
<td>b₁₄ a₄</td>
<td>b₁₅</td>
<td>b₁₆</td>
</tr>
<tr>
<td>R₂</td>
<td>b₂₁</td>
<td>b₂₂</td>
<td>a₃</td>
<td>a₄</td>
<td>b₂₅</td>
<td>b₂₆</td>
</tr>
<tr>
<td>R₃</td>
<td>a₁</td>
<td>b₃₂</td>
<td>a₂</td>
<td>b₃₃ a₃</td>
<td>b₃₄ a₄</td>
<td>a₅</td>
</tr>
</tbody>
</table>

No more changes. Third row with all a symbols. So a lossless join.
Testing for lossless decomposition property (6/6)

R – given schema. F – given set of FDs
The decomposition of R into R₁, R₂ is lossless wrt F if and only if
either \( R₁ \cap R₂ \rightarrow (R₁ – R₂) \) belongs to \( F⁺ \) or
\( R₁ \cap R₂ \rightarrow (R₂ – R₁) \) belongs to \( F⁺ \)

Eg. gradeInfo (rollNo, studName, course, grade)
with FDs = \{rollNo, course → grade; studName, course → grade;
rollNo → studName; studName → rollNo\}
decomposed into
grades (rollNo, course, grade) and studInfo (rollNo, studName)
is lossless because
rollNo → studName
A property of lossless joins

$D_1: (R_1, R_2, \ldots, R_K)$ lossless decomposition of $R$ wrt $F$

$D_2: (R_{i1}, R_{i2}, \ldots, R_{ip})$ lossless decomposition of $R_i$ wrt $F_i = \pi_{R_i}(F)$

Then

$D = (R_1, R_2, \ldots, R_{i-1}, R_{i1}, R_{i2}, \ldots, R_{ip}, R_{i+1}, \ldots, R_k)$ is a lossless decomposition of $R$ wrt $F$

This property is useful in the algorithm for BCNF decomposition
Algorithm for BCNF decomposition

R – given schema. F – given set of FDs

D = \{R\}  // initial decomposition
while there is a relation schema \( R_i \) in D that is not in BCNF do
{ let \( X \rightarrow A \) be the FD in \( R_i \) violating BCNF;
  Replace \( R_i \) by \( R_{i1} = R_i - \{A\} \) and \( R_{i2} = X \cup \{A\} \) in D;
}

Decomposition of \( R_i \) is lossless as
\[ R_{i1} \cap R_{i2} = X, R_{i2} - R_{i1} = A \text{ and } X \rightarrow A \]

Result: a lossless decomposition of R into BCNF relations
Dependencies may not be preserved (1/2)

Consider the schema: townInfo (stateName, townName, distName) with the FDs F: ST → D (town names are unique within a state)
D → S

Keys: ST, DT. – all attributes are prime
– relation in 3NF

Relation is not in BCNF as D → S and D is not a key

Decomposition given by algorithm: R₁: TD  R₂: DS

Not dependency preserving as \( \pi_{R₁}(F) = \) trivial dependencies
\( \pi_{R₂}(F) = \{D \rightarrow S\} \)

Union of these doesn’t imply ST → D
ST → D can’t be enforced unless we perform a join.
Dependencies may not be preserved (2/2)

Consider the schema: R (A, B, C)
with the FDs F: AB → C and C → B
Keys: AB, AC – relation in 3NF (all attributes are prime)
– Relation is not in BCNF as C → B and C is not a key

Decomposition given by algorithm: R₁: CB R₂: AC
Not dependency preserving as π₁(F) = trivial dependencies
π₂(F) = {C → B}
Union of these doesn’t imply AB → C

All possible decompositions: {AB, BC}, {BA, AC}, {AC, CB}
Only the last one is lossless!

Lossless and dependency-preserving decomposition doesn't exist.
Equivalent Dependency Sets

F, G – two sets of FDs on schema R
F is said to cover G if $G \subseteq F^+$ (equivalently $G^+ \subseteq F^+$)
F is equivalent to G if $F^+ = G^+$ (or, F covers G and G covers F)
Note: To check if F covers G,
    it’s enough to show that for each FD $X \rightarrow Y$ in G, $Y \subseteq X^+_F$
Canonical covers or Minimal covers

It is of interest to reduce a set of FDs $F$ into a “standard” form $F'$ such that $F'$ is equivalent to $F$.

We define that a set of FDs $F$ is in ‘minimal form’ if

(i) the rhs of any FD of $F$ is a single attribute
(ii) there are no redundant FDs in $F$
    that is, there is no FD $X \rightarrow A$ in $F$
    s.t $(F - \{X \rightarrow A\})$ is equivalent to $F$
(iii) there are no redundant attributes on the lhs of any FD in $F$
    that is, there is no FD $X \rightarrow A$ in $F$ s.t there is $Z \subset X$ for which
    $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ is equivalent to $F$

Minimal Covers

useful in obtaining a lossless, dependency-preserving decomposition of a scheme $R$ into 3NF relation schemas
Algorithm for computing a minimal cover

R – given Schema or set of attributes;  F – given set of fd’s on R

Step 1:  G := F

Step 2:  Replace every fd of the form $X \rightarrow A_1 A_2 A_3 \ldots A_k$ in G by $X \rightarrow A_1; X \rightarrow A_2; X \rightarrow A_3; \ldots ; X \rightarrow A_k$

Step 3: For each fd $X \rightarrow A$ in G do
    for each B in X do
        if $A \in (X - B)^+ \text{ wrt } G$ then
            replace $X \rightarrow A$ by $(X - B) \rightarrow A$

Step 4: For each fd $X \rightarrow A$ in G do
    if $(G - \{X \rightarrow A\})^+ = G^+$ then
        replace G by $G - \{X \rightarrow A\}$
3NF decomposition algorithm

R – given Schema;  F – given set of fd’s on R in minimal form

Use BCNF algorithm to get a lossless decomposition \( D = (R_1, R_2, \ldots, R_k) \)
Note: each \( R_i \) is already in 3NF (it is in BCNF in fact!)

Algorithm: Let \( G \) be the set of fd’s not preserved in \( D \)
For each \( fd \ Z \rightarrow A \) that is in \( G \)
Add relation scheme \( S = (B_1, B_2, \ldots, B_s, A) \) to \( D \). // \( Z = \{B_1, B_2, \ldots, B_s\} \)

As \( Z \rightarrow A \) is in \( F \) which is a minimal cover,
there is no proper subset \( X \) of \( Z \) s.t \( X \rightarrow A \). So \( Z \) is a key for \( S \)!

Any other \( fd \ X \rightarrow C \) on \( S \) is such that \( C \) is in \( \{B_1, B_2, \ldots, B_s\} \).
Such fd’s do not violate 3NF because each \( B_j \)’s is prime a attribute!

Thus any scheme \( S \) added to \( D \) as above is in 3NF.

\( D \) continues to be lossless even when we add new schemas to it!
Multi-valued Dependencies (MVDs)

studCourseEmail(rollNo,courseNo,emailAddr)

a student enrolls for several courses and has several email addresses

rollNo $\rightarrow\rightarrow$ courseNo (read as rollNo multi-determines courseNo)

If (CS05B007, CS370, shyam@gmail.com)
(CS05B007, CS376, shyam@yahoo.com) appear in the data then

(CS05B007, CS376, shyam@gmail.com)
(CS05B007, CS370, shyam@yahoo.com)

should also appear for, otherwise, it implies that having gmail address has something to with doing course CS370 !!

By symmetry, rollNo $\rightarrow\rightarrow$ emailAddr
More about MVDs

Consider `studCourseGrade(rollNo, courseNo, grade)`

Note that `rollNo →→ courseNo` does not hold here even though `courseNo` is a multi-valued attribute of student

If

(1) (CS05B007, CS370, A)
(2) (CS05B007, CS376, B) appear in the data then
(3) (CS05B007, CS376, A)
(4) (CS05B007, CS370, B) will not appear !!

Attribute ‘grade’ depends on `(rollNo, courseNo)`

MVD’s arise when two unrelated multi-valued attributes of an entity are sought to be represented together.
More about MVDs

Consider

\[\text{studCourseAdvisor}(\text{rollNo}, \text{courseNo}, \text{advisor})\]

Note that rollNo $\rightarrow\rightarrow$ courseNo holds here

If \[(\text{CS05B007, CS370, Dr Ravi})\]
\[(\text{CS05B007, CS376, Dr Ravi})\]
appear in the data then swapping courseNo values gives rise to existing tuples only.

But, since rollNo $\rightarrow$ advisor and (rollNo, courseNo) is the key, this gets caught in checking for 2NF itself.
Alternative definition of MVDs

Consider $R(X,Y,Z)$

Suppose that $X \rightarrow Y$ and by symmetry $X \rightarrow Z$

Then, decomposition $D = (XY, XZ)$ should be lossless

That is, for any instance $r$ on $R$, $r = \pi_{XY}(r) \times \pi_{XZ}(r)$
MVDs and 4NF

An MVD $X \rightarrow\rightarrow Y$ on scheme $R$ is called *trivial* if either $Y \subseteq X$ or $R = X \cup Y$. Otherwise, it is called *nontrivial*.

4NF: A relation $R$ is in 4NF if it is in BCNF and for every nontrivial MVD $X \rightarrow\rightarrow A$, $X$ must be a superkey of $R$.

$\text{studCourseEmail(rollNo,courseNo,emailAddr)}$

is not in 4NF as $\text{rollNo} \rightarrow\rightarrow \text{courseNo}$ and $\text{rollNo} \rightarrow\rightarrow \text{emailAddr}$

are both nontrivial and $\text{rollNo}$ is not a superkey for the relation