Database Systems

-Relational Database Design

Dr. Rudra Pratap Deb Nath

Associate Professor

Department of Computer Science and Engineering

University of Chittagong

rudra@cu.ac.bd

4th Semester 2024

Motivation

- A good design prevents problems that occur later during usage
- Changing a database schema before it is in use is much cheaper
- Redundancy is the source of many problems and should be avoided by meeting certain design criteria

Learning Goals

- Understand the concepts of functional dependencies, attribute closure and normal forms
- Describe the quality of a design by using normal forms
- Improve a database design by decomposition of relations

Anomalies
Why is this a bad schema?

empid	name	rank	office	courseId	title	ects
2125	Socrates	C4	226	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
2137	Kant	C4	7	4630	Constructive Criticism	4

Modification Anomalies

empid	name	rank	office	courseId	title	ects
2125	Socrates	C4	226	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
2137	Kant	C4	7	4630	Constructive Criticism	4
-	-	-	-	4000	Data Structures	4

Insert anomaly: Insertion of a new course is not allowed.

Modification Anomalies

empid	name	rank	office	courseId	title	ects
2125	Socrates	C4	338	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
2137	Kant	C4	7	4630	Constructive Criticism	4
-	-	-	-	4000	Data Structures	4

Update anomaly: Update the teacher's office to 338 who teaches Ethics.

Modification Anomalies

empid	name	rank	office	courseId	title	ects
2125	Socrates	C4	338	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
?	?	?				
-	-	-	-	4000	Data Structures	4

Delete anomaly: No more ”Constructive Criticism” course.

A Good Decomposition

courseOffers

professor ⋈ course

empid	name	rank	office	courseid	title	ects
2125	Socrates	C4	226	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
2137	Kant	C4	7	4630	Constructive Criticism	4

Requirements:

- Includes all the attributes ✓
- Lossless-join decomposition ✓

$\pi_{empid,name,rank,office}$(courseOffers)

professor

empid	name	rank	office
2125	Socrates	C4	226
2126	Russel	C4	232
2128	Russel	C2	230
2133	Popper	C3	52
2134	Augustinus	C3	309
2137	Kant	C4	7

$\pi_{courseid,title,ects,empid}$(courseOffers)

course

courseid	title	ects	empid
5001	Basics	4	2137
5041	Ethics	4	2125
5043	Theory of Cognition	3	2126
5049	DBS	2	2125
4052	Logics	4	2125
5052	Theory of Science	3	2126
5216	Bioethics	2	2128
5259	Advanced Algorithms	2	2133
5022	Belief and Knowledge	2	2134
4630	Constructive Criticism	4	2137

A Bad Decomposition

courseOffers

empid	name	rank	office	courseid	title	ects
2125	Socrates	C4	226	5041	Ethics	4
2125	Socrates	C4	226	5049	DBS	2
2125	Socrates	C4	226	4052	Logics	4
2137	Kant	C4	7	5001	Basics	4
2126	Russel	C4	232	5043	Theory of Cognition	3
2126	Russel	C4	232	5052	Theory of Science	3
2128	Russel	C2	230	5216	Bioethics	2
2133	Popper	C3	52	5259	Advanced Algorithms	2
2134	Augustinus	C3	309	5022	Belief and Knowledge	2
2137	Kant	C4	7	4630	Constructive Criticism	4

Requirements:

- Includes all the attributes ✓
- Lossless decomposition ✗

$\pi_{courseid,title,ects,name}$(courseOffers)

course

courseid	title	ects	name
5001	Basics	4	Kant
5041	Ethics	4	Socrates
5043	Theory of Cognition	3	Russel
5049	DBS	2	Socrates
4052	Logics	4	Socrates
5052	Theory of Science	3	Russel
5216	Bioethics	2	Russel
5259	Advanced Algorithms	2	Popper
5022	Belief and Knowledge	2	Augustinus
4630	Constructive Criticism	4	Kant

$\pi_{empid,name,rank,office}$(courseOffers)

professor

empid	name	rank	office
2125	Socrates	C4	226
2126	Russel	C4	232
2128	Russel	C2	230
2133	Popper	C3	52
2134	Augustinus	C3	309
2137	Kant	C4	7

Functional Dependencies

- Let $\boldsymbol{\alpha} \subseteq \mathcal{R}$ and $\boldsymbol{\beta} \subseteq \mathcal{R}$ be sets of attributes
- A **functional dependency** $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ holds on the schema $\mathcal{R}$ if for all legal instances R of $\mathcal{R}$:

$$\forall t1,t2 \in R : t1.\alpha = t2.\alpha \Rightarrow t1.\beta = t2.\beta$$

- The α values **uniquely identify** the β values
- α **functionally determines** β
- If β⊆α, $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is called **trivial: Always valid**
- If $\boldsymbol{\alpha} \cap \boldsymbol{\beta} = \emptyset$, $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is called **non-trivial** : the validity should be checked.

Functional Dependencies(find correct ones)

- $\{\text{empid}\} \rightarrow \{\text{name}\}$
- $\{\text{rank}\} \rightarrow \{\text{name}\}$
- $\{\text{name}\} \rightarrow \{\text{empi}d\}$
- $\{\text{name}, \text{empid}\} \rightarrow \{\text{rank}, \text{off}ice\}$

empid	name	rank	office
2125	Socrates	C4	226
2126	Russel	C4	232
2128	Russel	C2	230
2133	Popper	C3	52
2134	Augustinus	C3	309
2137	Kant	C4	7

Given the following relation, which functional dependencies are fulfilled?

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

A	B	C	D
a1	b1	c1	d
a2	b2	c2	d21
a2	b2	c2	d3
a3	b1	c3	d3

** Functional dependencies are semantic constraints that need to be true for all possible instances

Super Key

- $\alpha \subseteq \mathcal{R}$ is a **super key** if $\alpha \rightarrow \mathcal{R}$, i.e., α determines all attribute values
- The set of all attributes is a super key: $\mathcal{R} \rightarrow \mathcal{R}$
- Super keys are not necessarily minimal
- **empid** is a superkey in our example
- But { empid, personid } is also a superkey

empid	personid	name	rank	office
2125	123654	Socrates	C4	226
2126	234569	Russel	C4	232
2128	234570	Russel	C2	230
2133	112255	Popper	C3	52
2134	36528	Augustinus	C3	309
2137	12365	Kant	C4	7

Super Key

- Maximum number of possible super key of a relation with n attributes $:= 2^n - 1$
- What is the maximum number of the super key of the given relation?

A	B	C	D
1	1	5	1
2	1	7	1
3	1	7	1
4	2	7	1
5	2	5	1
6	2	5	2

Candidate Key

- β is fully functionally dependent on $\boldsymbol{\alpha}$ if $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ and the functional dependency $\boldsymbol{\alpha}' \rightarrow \boldsymbol{\beta}$ does not hold for any $\boldsymbol{\alpha}'$, $\boldsymbol{\alpha}' \subset \boldsymbol{\alpha}$
- $\boldsymbol{\alpha} \subseteq \mathcal{R}$ is a candidate key if $\mathcal{R}$ is fully functionally dependent on $\boldsymbol{\alpha}$
- Candidate key is the super key whose proper subset is not a super key, i.e., minimal super key
- One of the candidate keys is chosen as primary key.

Deriving functional dependencies

- { empid } → { empid, name, dept, building, number }
- { building } → { dept }
- { dept } → { building }
- { building, number } → { empid }

empid	name	dept	building	number
2125	Socrates	PHI	2	3
2126	Russel	SCI	7	3
2128	Russel	PHI	2	5
2133	Popper	CS	9	5
2134	Augustinus	THE	5	7
…	…	…	…	…

- { dept, number } → { empid }
- { building, number } → { empid, name, dept, building, number }

The closure of a set of functional dependencies

- Let F be a set of functional dependencies, its **closure** is denoted as **F^+**
- F^+ includes all the functional dependencies that can be derived from F, i.e., all FDs **logically implied** by dependencies in F
- Inference rules (Armstrong axioms) can be used to compute F^+

Armstrong axioms

- Let α, β, γ, δ be subsets of attributes in $\mathcal{R}$
- **Reflexivity**: if $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$
- **Augmentation**: if $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$
- **Transitivity**: if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$
- These axioms are **sound** and **complete**
- They generate all the possible correct funcional dependencies and only those that are correct

Armstrong axioms

- **Union**: if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$
- **Decomposition**: if $\alpha \rightarrow \beta\gamma$ then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$
- **Pseudotransitivity**: if $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$ then $\alpha\gamma \rightarrow \delta$
- These rules are not essential, but sound and ease the derivation process
- It is possible to prove that these rules are sound using only the Armstrong axioms

Example

- Functional dependencies F:
  - $A \rightarrow BC$
  - $CD \rightarrow E$
  - $B \rightarrow D$
  - $E \rightarrow A$

Reflexivity: if $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$
Augmentation: if $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$
Transitivity: if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

- Derived FDs:
  - $E \rightarrow A$ and $A \rightarrow BC$, then $E \rightarrow BC$
  - $B \rightarrow D$, then $CB \rightarrow CD$
  - $CB \rightarrow CD$ and $CD \rightarrow E$, then $CB \rightarrow E$

Class assignment- 7.6,

Closure of a set of attributes

- How can you test whether α is a super key using F?
- Given $\mathcal{R}$(A, B, C, D, E) , $FD: \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E\}$, Determine the set of attributes can be determined by A, B, C, and D.
- To test whether α is a super key, we need to compute the set of attributes functionally determined α
- The closure of a set of attributes α with respect to a set of FDs F is denoted as α^+, $\alpha^+ = \{A | \alpha \rightarrow A \in F^+\}$

Closure of a set of attributes

- If $\alpha \rightarrow \beta \in F^+$, then $\beta \subseteq \alpha^+$
- Relation R: {[A, B, C, D, E]}
- α = { A, B } and F is
  - $A \rightarrow BC$
  - $CD \rightarrow E$
  - $B \rightarrow D$
  - $E \rightarrow A$
- result = { A, B, C, D, E }

```
attrClosure(F : set of FDs; α : set of attributes)
  result = α
  repeat
   for each β → γ in F do
    if β ⊆ result then
      result = result ∪ γ
    end if
   end for
  until (result does not change)
```

More on Attribute Closure Algorithm

Closure of a set of attributes

- The attribute closure algorithm computes the closure correctly. It terminates and when it terminates, all parts of the closure are in the result
- The attribute closure algorithm can be used to test if a functional dependency holds and if a given set of attributes is a super key
- If $\alpha^+ = \mathcal{R}$, then α is a super key of $\mathcal{R}$

```
attrClosure(F : set of FDs; α : set of attributes)
  result = α
  repeat
    for each β → γ in F do
      if β ⊆ result then
        result = result ∪ γ
      end if
    end for
  until (result does not change)
```

Class assignment- 7.30 (A,B), * R{A, B, C, D, E}, FD={$A \rightarrow B, D \rightarrow E$} Find SKs

Finding all candidate keys of a relation

- Short cut method
- First take the closure of all attributes, then discard attributes from the super key if it can be discarded using FDs.
- Once the candidate key are found, check whether there are any prime attributes available in the right-hand side of the FDs. If yes, make another super key by replacing the prime attributes of the CK with the left side of the FDs. Repeat the process.

Class assignment- *Given that R{A, B, C, D}, FD={$A \rightarrow B, B \rightarrow C$, C→ A} Find CKs
** Given that R{A, B, C, D, E, F}, FD={$\mathrm{AB} \rightarrow C, C \rightarrow DE, E \rightarrow F, D \rightarrow A, C \rightarrow B$} Find CKs

Normalization of relations by decomposition

- Decompose a relational schema $\mathcal{R}$ into multiple relational schemas $\mathcal{R}_1,\ldots,\mathcal{R}_n$ to eliminate problems in the original design
- Normal forms describe the quality of a design
- 1NF, 2NF, 3NF, BCNF, 4NF…..
- Prohibit particular functional dependencies in a relation to avoid redundancy, null values, and anomalies
- Good ER modeling typically directly leads to 3NF (or higher NF) relations
- Normalization eliminates problems caused by FDs among attributes of an entity type.
- Two properties of a decomposition: 1) Lossless decomposition (mandatory) and 2) Dependency preservation (optional in BCNF)

Valid and lossless decomposition

- A decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ is **valid** if $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, i.e., no attributes in $\mathcal{R}$ get lost
  - $R_1 := \pi_{\mathcal{R}_1}(R)$
  - $R_2 := \pi_{\mathcal{R}_2}(R)$
- A decomposition of $\mathcal{R}$ into $\mathcal{R}_1$ and $\mathcal{R}_2$ is **lossless** if $R = R_1 \bowtie R_2$ for all possible instances R, R_1, R_2 of $\mathcal{R}$, $\mathcal{R}_1$, $\mathcal{R}_2$
  - In a lossless (or lossless-join) decomposition all data in the original schema must be reconstructible with a natural join from the instances of the new schemas

Lossless decomposition

- Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be a decomposition of $\mathcal{R}$ and $F_{\mathcal{R}}$ the set of FDs in $\mathcal{R}$
- $\mathcal{R}_1$ and $\mathcal{R}_2$ is a **lossless** decomposition if we can derive **at least one** of the following FDs:
  - $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_1 \in F^+_{\mathcal{R}}$,i.e., common attributes are super key in $\mathcal{R}_1$ or
  - $(\mathcal{R}_1 \cap \mathcal{R}_2) \rightarrow \mathcal{R}_2 \in F^+_{\mathcal{R}}$,i.e., common attributes are super key in $\mathcal{R}_2$
- A decomposition that is not lossless is said to be **lossy**

Example

courseOffers

empid	name	courseid	ects
2125	Socrates	5041	4
2125	Socrates	5049	2
2125	Socrates	4052	4
2137	Kant	5001	4
2126	Russel	5043	3
2126	Russel	5052	3
2128	Russel	5216	2

The relationship empid, name, courseid, and ects got lost

$\pi_{empid,name}$ (courseOffers)

empid	name
2125	Socrates
2126	Russel
2128	Russel
2133	Popper
2134	Augustinus
2137	Kant

Professor

professor $\bowtie$ course

empid	name	courseid	ects
2125	Socrates	5041	4
2125	Socrates	5049	2
2125	Socrates	4052	4
2137	Kant	5001	4
2126	Russel	5043	3
2126	Russel	5052	3
2126	**Russel**	**5216**	**2**
2128	**Russel**	**5043**	**3**
2128	**Russel**	**5052**	**3**
2128	Russel	5216	2

$\pi_{courseid,ects,name}$(courseOffers)

courseid	ects	name
5001	4	Kant
5041	4	Socrates
5043	3	Russel
5049	2	Socrates
4052	4	Socrates
5052	3	Russel
5216	2	Russel

course

Dependency preservation

Student_Mark		
ID	Name	GPA
1	Bob	3.5
2	Bob	3.6
3	Alice	3.5
4	Alice	3.6

Student	
ID	Name
1	Bob
2	Bob
3	Alice
4	Alice

Mark	
Name	GPA
Bob	3.5
Bob	3.6
Alice	3.5
Alice	3.6

$FD: \{\ ID \rightarrow (Name),$
$ID \rightarrow GPA,$
$ID \rightarrow (Name, GPA),$
$(Name, GPA), \rightarrow ID$
$\}$

$FD_1: \{\ ID \rightarrow (Name), \}$

$FD_2: \{\ \}$

$$\text{FD} \not\equiv FD_1 \cup FD_2$$

Dependency preservation

- All functional dependencies that hold for $\mathcal{R}$ must be verifiable in the new schemas $\mathcal{R}_1,\ldots, \mathcal{R}_n$
  - We can check all dependencies locally on $\mathcal{R}_1,\ldots,\mathcal{R}_n$
  - We avoid the alternative: computing the join $\mathcal{R}_1 \bowtie \cdots \bowtie \mathcal{R}_n$ to test if an FD is violated
- Let $F_{\mathcal{R}}$, $F_{\mathcal{R}_i}$ be the set of FDs in $\mathcal{R}$ and $\mathcal{R}_i$, the decomposition of $\mathcal{R}$ into $\mathcal{R}_1,\ldots, \mathcal{R}_n$ is **dependency preserving** if $F_{\mathcal{R}} \equiv (F_{\mathcal{R}_1} \cup \cdots \cup F_{\mathcal{R}_n})$, i.e.,

$$F^+_{\mathcal{R}} = (F_{\mathcal{R}_1} \cup \cdots \cup F_{\mathcal{R}_n})^+$$

Checking dependency preservation

- The decomposition $\mathcal{R}_1,\ldots,\mathcal{R}_n$ is **dependency preserving if** for each FD there is an $\mathcal{R}_i$ in $\mathcal{R}_1,\ldots,\mathcal{R}_n$ that contains all involved attributes
- The decomposition $\mathcal{R}_1,\ldots,\mathcal{R}_n$ is **dependency preserving if** for all $\alpha \rightarrow \beta$ in F:
  $\beta \subseteq$ attrClosureD(F, α, $\mathcal{R}_1,\ldots,\mathcal{R}_n$)

```
attrClosureD(F : set of FDs;
  α : set of attributes; R1…Rn : a decomposition of R)
  result = α
  repeat
    for each Ri in R1…Rn do
        t = (result ∩ Ri)+ ∩ Ri
        result = result ∪ t
    end for
  until (result does not change)
```

Is this decomposition dependency preserving?

- R = {| A, B, C |}
- F = { A → B, B → C |}
- R1 = {| A, B |}, R2 = {| B, C |}
- Yes!
- A → B can be checked locally in R1
- B → C can be checked locally in R2

Is this decomposition dependency preserving?

- R = {| A, B, C |}
- F = { A → B, B → C |}
- R1 = {| A, B |}, R2 = {| A, C |}

result = { B }

result = { B }

result = { B }

C is not included in result
Then, the decomposition is not dependency preserving

```
attrClosureD(F : set of FDs;
  α : set of attributes; R1…Rn : a decomposition of R)
  result = α
  repeat
   for each Ri in R1…Rn do
     t = (result ∩ Ri)+ ∩ Ri
     result = result ∪ t
   end for
  until (result does not change)
```

Lossless Decomposition example

- dept1: {[name, street, number, city]}, dept2: {[name, postcode]}
- FDs
  - { name } → { street, number, city, postcode }
  - { street, number, city } → { postcode }
  - { street, number, city } → { name }
- Is dept1 ∩ dept2 a **superkey** of dept1 or dept2?

✓ ✓

Example

- department: {[name, street, number, city, postcode]}
- FDs
  - { name } → { street, number, city, postcode }
  - { street, number, city } → { postcode }
  - { street, number, city } → { name }
- Decomposition
  - dept1: {[name, street, number, city]}, dept2: {[name, postcode]}
- Is this decomposition lossless? Is this decomposition dependency preserving?

Example

- dept1: {[name, street, number, city]}, dept2: {[name, postcode]}
- FDs
  - { name } → { street, number, city, postcode } ✓
  - { street, number, city } → { postcode }
  - { street, number, city } → { name }

result = { name }

t = { name, street, number, city }

result = { name, street, number, city }

t = { name, postcode }

result = { name, street, number, city, postcode }

```
attrClosureD(F : set of FDs;
  α : set of attributes; R1…Rn : a decomposition of R)
  result = α
  repeat
    for each Ri in R1…Rn do
      t = (result ∩ Ri)+ ∩ Ri
      result = result ∪ t
    end for
  until (result does not change)
```

Example

- dept1: {[name, street, number, city]}, dept2: {[name, postcode]}
- FDs
  - { name } → { street, number, city, postcode } ✓
  - { street, number, city } → { postcode } ✓
  - { street, number, city } → { name } ✓

result = { street, number, city }

t = { name, street, number, city }

result = { name, street, number, city }

t = { name, postcode }

result = { name, street, number, city, postcode }

```
attrClosureD(F : set of FDs;
  α : set of attributes; R1…Rn : a decomposition of R)
  result = α
  repeat
    for each Ri in R1…Rn do
      t = (result ∩ Ri)+ ∩ Ri
      result = result ∪ t
    end for
  until (result does not change)
```

Class assignment

- department: {[name, street, number, city, postcode]}
- FDs
  - { name } → { street, number, city, postcode }
  - { street, number, city } → { postcode }
  - { street, number, city } → { name }
- Decomposition
  - dept1: {[name, street, number]}, dept2: {[name, **city,** postcode]}
- Is this decomposition lossless? Is this decomposition dependency preserving?

Class assignment

- Given $\mathcal{R}$(A, B, C, D) , $FD: \{A \to B, C \to D\}$, check whether the following relations preserve dependency: $\mathcal{R}_1$(A,C) and $\mathcal{R}_3$(B, D)
- Given $\mathcal{R}$(A, B, C, D) , $FD: \{A \to B, B \to C, C \to D, D \to A\}$, check whether the following relations preserve dependency: **a)** $\mathcal{R}_1$(A,B), $\mathcal{R}_2$(B,C) and $\mathcal{R}_3$(C, D) b) $\mathcal{R}_1$(A,B,C) and $\mathcal{R}_3$(C, D, E)
- Given $\mathcal{R}$(A, B, C, D, E) , $FD: \{A \to BCDE, BC \to AED, , D \to E\}$, check whether the following relations preserve dependency: $\mathcal{R}_1$(A, B), $\mathcal{R}_2$(B,C), and $\mathcal{R}_3$(C,D,E)
- Given $\mathcal{R}$(A, B, C, D, E) , $FD: \{A \to BCD, B \to AE, BC \to AED, D \to E, C \to DE\}$, check whether the following relations preserve dependency: $\mathcal{R}_1$(A, B), $\mathcal{R}_2$(B,C), and $\mathcal{R}_3$(C, D, E)
- ODD groups with alternative solutions and EVEN groups with by checking FD’s locally.

Functional Dependencies

- A good relation design has **no redundancy**, **no update anomalies**, ability to **represent all the information**
- A bad design can be converted to a good design by **decomposing** large relational schemas into smaller ones
- A good decomposition is **lossless** and **dependency preserving**

Normal Forms

- Normal Forms define characteristics of relational schemas
- NFs forbid certain combinations of FDs in a relation
- NFs avoid redundancies and anomalies
- NFs are guidelines to obtain good decompositions
- Examples: **1NF, 2NF, 3NF, BCNF**, 4NF

First normal form (1NF)

- A relation $\mathcal{R}$ is in 1NF if the domains of all its attributes are atomic (No composite of multi-valued domains)

Empid	Name	Address	phone
2125	Bob	Hathazari, Chittagong, Bangladesh	P1, p2
2130	Alice	Fatickchari, Chittagong, Bangladesh	p3
2199	Blak	Haraldslund, Aalborg, Denmark	P4,p6

Not 1NF

Empid	Name	sub-dis	district	country	phone
2125	Bob	Hathazari	Chittagong	Bangladesh	p1
2125	Bob	Hathazari	Chittagong	Bangladesh	p2
2130	Alice	Fatickchari,	Chittagong	Bangladesh	p3
2199	Blak	Haraldslund	Aalborg	Denmark	p4
2199	Blak	Haraldslund	Aalborg	Denmark	p6

1NF

Second normal form (2NF)

- $\mathcal{K}_j$ is a candidate key in $\mathcal{R}$
- A is a **prime** attribute if A ∈ ($\mathcal{K}_1$ ∪ … ∪ $\mathcal{K}_i$)
- A relational schema $\mathcal{R}$ with FDs F is in 2NF if each non-prime attribute A in $\mathcal{R}$ is fully functionally dependent on each candidate key $\mathcal{K}_j$ in $\mathcal{R}$
  - ($\mathcal{K}_j$ → A) ∈ F^+ and $\mathcal{K}_j$ is left reduced (fully functionally dependent)
- 2NF **prevents partial dependencies**
- There are no functional dependencies of non-prime attributes on only a subset of key attributes

** No proper subset of CKs → $Non\ prime\ attribute(s)$

Example

empid	name	courseid	ects
2125	Socrates	5041	4
2125	Socrates	5049	2
2125	Socrates	5001	4
2137	Kant	5001	4

- Candidate keys: { {empid, courseid } }
- Key attributes (prime): { empid, courseid }
- Non-key attributes (non-prime): { name, ects }
- FDs: { empid } → { name } and { courseid } → { ects }

$(\mathcal{K}_j \rightarrow A) \in F^+$ and $\mathcal{K}_j$ is left reduced (fully functionally dependent)

- { empid, courseid } → { name } but as { empid } → { name }, then { name } is not fully functionally dependent on { empid, courseid }

Eliminating partial dependencies

Not 2NF:

empid	name	courseid	ects
2125	Socrates	5041	4
2125	Socrates	5049	2
2125	Socrates	4052	4
2137	Kant	5001	4

2NF:

empid	name
2125	Socrates
2137	Kant

courseid	ects
5041	4
5049	2
4052	4
5001	4

empid	courseid
2125	5041
2125	5049
2125	4052
2137	5001

2NF decomposition

- Given $\mathcal{R}$(A, B, C, D) , $FD: \{A \rightarrow B, C \rightarrow D\}$, check whether the relation is in 2NF, if not decompose it.

Third normal form (3NF)

- A relational schema $\mathcal{R}$ is in 3NF if at least one of the following conditions holds for each of its FDs $\alpha \rightarrow \beta$ with $\alpha,\beta \subseteq \mathcal{R}$:
  - $\beta \subseteq \alpha$, i.e., the FD is **trivial**
  - α is a **super key** of $\mathcal{R}$
  - Any attribute B in $\beta - \alpha$ is **prime** (part of a candidate key)
- Non-prime attributes must not determine other Non-prime attributes
- 3NF prevents partial and transitive dependencies, except transitive dependencies with prime attributes as endpoints
- 3NF “includes” 2NF

Example: 3NF

empid	name	dept	building
2125	Socrates	PHI	2
2126	Russel	SCI	7
2128	Russel	PHI	2
2133	Popper	CS	9
2134	Augustinus	THE	5

- Candidate keys: { { empid } }
- Key attributes (prime): { empid }
- Non-key attributes (non-prime): { name, de
- FDs: { empid } → { name, dept, building } and { dept } → { building }

	{ empid } → { name, dept, building }	{ dept } → { building }
β ⊆ α, i.e., the FD is **trivial**	✘	✘
α is a **super key** of $\mathcal{R}$	✓	✘
Any attribute B in β - α is **prime**	✘	✘

Eliminating transitive dependency

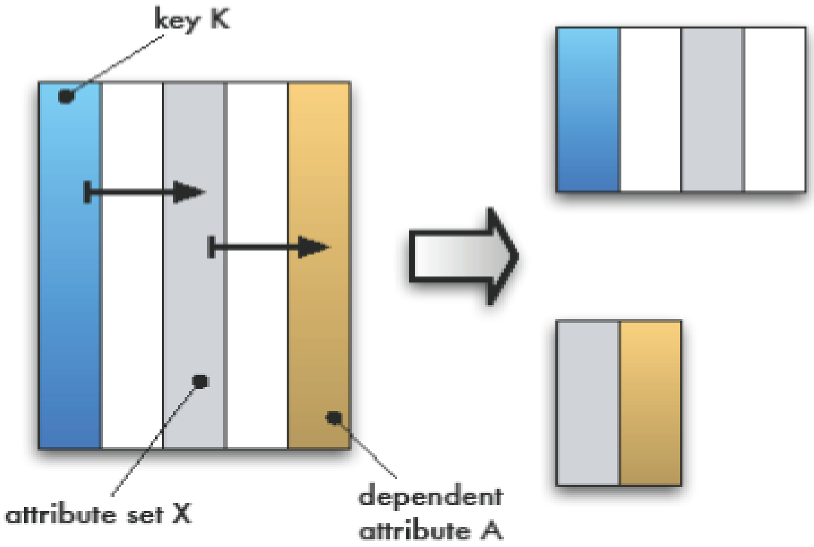
Not 3NF:

empid	name	dept	building
2125	Socrates	PHI	2
2126	Russel	SCI	7
2128	Russel	PHI	2
2133	Popper	CS	9
2134	Augustinus	THE	5

3NF:

empid	name	dept
2125	Socrates	PHI
2126	Russel	SCI
2128	Russel	PHI
2133	Popper	CS
2134	Augustinus	THE

dept	building
PHI	2
SCI	7
CS	9
THE	5



3NF decomposition

- Given $\mathcal{R}$(A, B, C, D) , $FD\colon\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$, check whether the relation is in 3NF, if not decompose it.

Boyce Codd Normal Form (BCNF)

- A relational schema $\mathcal{R}$ is in BCNF if at least one of the following conditions holds for each of its FDs $\alpha \rightarrow \beta$ with $\alpha, \beta \subseteq \mathcal{R}$:
  - $\beta \subseteq \alpha$, i.e., the FD is **trivial**
  - α is a **super key** of $\mathcal{R}$
  - ~~Any attribute B in $\beta - \alpha$ is **prime** (part of a candidate key)~~
- Difference to 3NF: no third option
- BCNF is a stricter form of 3NF (“includes” 3NF)
- BCNF prevents all **transitive** dependencies, even those with prime attributes as endpoints

BCNF: Example

empid	name	dept
2125	Socrates	PHI
2126	Russel	SCI
2128	Russel	PHI
2133	Popper	CS
2134	Augustinus	THE

Candidate keys: { { empid } }

Key attributes (prime): { empid }

Non-key attributes (non-prime): { name, dept }

FDs: { empid } → { name, dept }

$\beta \subseteq \alpha$, i.e., the FD is **trivial** ✗

α is a **super key** of $\mathcal{R}$ ✓

BCNF: Decomposition

```
bcnf(F : set of FDs; R : relational schema)
  result = { R }
  while there is an Ri ∈ result that is not in BCNF do
    let α → β be a BCNF violating FD of Ri, such that α ∩ β = ∅ and α is not a super key in Ri
    decompose Ri into Ri1 := α ∪ β and Ri2 := Ri - β
    result = (result – Ri) ∪ Ri1 ∪ Ri2
  end while
```

empid	name	dept	building
2125	Socrates	PHI	2
2126	Russel	SCI	7
2128	Russel	PHI	2
2133	Popper	CS	9
2134	Augustinus	THE	5

- Candidate keys: { { empid } }
- Key attributes: { empid }
- FDs: { empid } → { name, dept, building } and { dept } → { building }

result: { {[empid, name, dept, building]} }

result: { {[dept, building]}, {[empid, name, dept]} }

result is lossless and dependency preserving

Another example

bcnf(F : set of FDs; R : relational schema)
  result = { R }
  while there is an Ri ∈ result that is not in BCNF **do**
    let α → β be a BCNF violating FD of Ri, such that α ∩ β = ∅ and α **is not a super key** in Ri
    decompose Ri into Ri1 := α ∪ β and Ri2 := Ri - β
    result = (result – Ri) ∪ Ri1 ∪ Ri2
  end while

dept	faculty	building	floor
PHI	HUM	2	3
SCI	ENG	7	3
CS	TECH	9	3
CS	TECH	9	2

- Candidate keys: { {floor, building} }
- Key attributes (prime): { floor, building }
- FDs: { floor, building } → { dept }, { dept } → { building, faculty }
- result: { {[dept, faculty, building, floor]} }
  result: { {[dept, building, faculty]}, {[dept, floor]} }

result is lossless but it isn't dependency preserving

Decomposition

- It is always possible to decompose a relational schema $\mathcal{R}$ with FDs F into
  - 3NF relational schemas $\mathcal{R}_1, \ldots, \mathcal{R}_n$ so that the decomposition is lossless and **dependency preserving**
  - BCNF relational schemas $\mathcal{R}_1, \ldots, \mathcal{R}_n$ so that the decomposition is lossless

Normalize a relation

- Determine which normal forms are fulfilled (identify highest normal form and normalize it) (restrict your consideration to 3NF and BCNF)

Given $\mathcal{R}$(A, B, C, D, E, F, G, H) , FD: $\{A \rightarrow BD, B \rightarrow C, E \rightarrow FG, AE \rightarrow H\}$,

1. $R: \{[A, B, C, D, E, F]\}$
   $A \rightarrow AB$
   $B \rightarrow AB$
   $CDE \rightarrow CDEF$
   $F \rightarrow ABF$
2. $R: \{[A, B, C, D]\}$
   $A \rightarrow B$
   $B \rightarrow A$
   $D \rightarrow C$
   $C \rightarrow D$
3. $R: \{[A, B, C, D, E, F]\}$
   $ABD \rightarrow CEF$
   $CEF \rightarrow ABD$
4. $R: \{[A, B, C, D, E]\}$
   $A \rightarrow B$
   $B \rightarrow C$
   $C \rightarrow D$
   $D \rightarrow E$
   $E \rightarrow A$
5. $R: \{[A, B, C, D, E, F]\}$
   $ABD \rightarrow CEF$
   $CEF \rightarrow ABD$
   $AC \rightarrow DF$
   $BD \rightarrow CE$
   $EF \rightarrow AB$
6. $R: \{[A, B, C, D, E, F]\}$
   $AC \rightarrow DF$
   $BD \rightarrow CE$
   $EF \rightarrow AB$

Summary

Normal Form	Main Characteristics
1NF	Only atomic attributes
2NF	No partial dependencies
3NF	No transitive dependencies except with prime attributes
BCNF	No transitive dependencies

- In practice, if BCNF decomposition is imposible without losing dependency preservation, we go for the 3NF decomposition (even if it allows for some redundancy)