CS315: Database Systems
Normalization Theory

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs315/

2nd semester, 2022-23

Tue 10:30-11:45, Thu 12:00-13:15

Database Design

- Central question: how to design a “good” database?

Database Design

- Central question: how to design a “good” database?
- Two ways of answering it: informally and formally

Database Design

- Central question: how to design a “good” database?
- Two ways of answering it: informally and formally
- Informal
  - Schemas should represent distinct entities
  - Little or no redundancy in storage
  - No modification anomaly
  - Less number of or no null values
  - No spurious tuple

Database Design

- Central question: how to design a “good” database?
- Two ways of answering it: informally and formally
- Informal
  - Schemas should represent distinct entities
  - Little or no redundancy in storage
  - No modification anomaly
  - Less number of or no null values
  - No spurious tuple
- Formally: *Normalization theory*

Modification Anomaly

- Consider the following schema: (roll, name, courseid, title)

Modification Anomaly

- Consider the following schema: (roll, name, courseid, title)
- Update anomaly
  - Changing title of a course causes updates to many students

Modification Anomaly

- Consider the following schema: (roll, name, courseid, title)
- Update anomaly
  - Changing title of a course causes updates to many students
- Insert anomaly
  - Admitting a student immediately requires a course and vice versa

Modification Anomaly

- Consider the following schema: (roll, name, courseid, title)
- Update anomaly
  - Changing title of a course causes updates to many students
- Insert anomaly
  - Admitting a student immediately requires a course and vice versa
- Delete anomaly
  - Deleting a course may delete all the corresponding students

Modification Anomaly

- Consider the following schema: (roll, name, courseid, title)
- Update anomaly
  - Changing title of a course causes updates to many students
- Insert anomaly
  - Admitting a student immediately requires a course and vice versa
- Delete anomaly
  - Deleting a course may delete all the corresponding students
- Thus, a bad design

Lossless Decomposition

- Must preserve losslessness of the corresponding (natural) join
- Lossy decomposition

Suppose

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

is decomposed into

roll	name
1	AB
2	AB
3	CD

and

name	batch
AB	2011
AB	2012
CD	2014

Lossless Decomposition

- Must preserve losslessness of the corresponding (natural) join
- Lossy decomposition

Suppose

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

is decomposed into

roll	name
1	AB
2	AB
3	CD

and

name	batch
AB	2011
AB	2012
CD	2014

whose join produces

roll	name	batch
1	AB	2011
1	AB	2012
2	AB	2011
2	AB	2012
3	CD	2014

with two spurious tuples

Lossless Decomposition

- Lossless decomposition

Suppose

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

is decomposed into

roll	name
1	AB
2	AB
3	CD

and

roll	batch
1	2011
2	2012
3	2014

Lossless Decomposition

- Lossless decomposition

Suppose

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

is decomposed into

roll	name
1	AB
2	AB
3	CD

and

roll	batch
1	2011
2	2012
3	2014

whose join produces

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

with no loss

Lossless Decomposition

- Lossless decomposition

Suppose

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

is decomposed into

roll	name
1	AB
2	AB
3	CD

and

roll	batch
1	2011
2	2012
3	2014

whose join produces

roll	name	batch
1	AB	2011
2	AB	2012
3	CD	2014

with no loss

- Try to preserve functional dependencies

Functional Dependencies

- Functional dependencies (FDs) are *constraints* derived from the meaning of and relationships among attributes
- A set of attributes X functionally determines Y, denoted by $X \rightarrow Y$, if the value of X determines a *unique* value of Y
  - roll → name; roll → batch
- For any two tuples t_1 and t_2 in any *legal* instance of $r(R)$, if $t_1.X = t_2.X$ then $t_1.Y = t_2.Y$
- A FD $X \rightarrow Y$ is trivial if it is satisfied for *all* instances of a relation
  - $Y \subseteq X$

Functional Dependencies

- Functional dependencies (FDs) are *constraints* derived from the meaning of and relationships among attributes
- A set of attributes X functionally determines Y, denoted by $X \rightarrow Y$, if the value of X determines a *unique* value of Y
  - roll → name; roll → batch
- For any two tuples t_1 and t_2 in any *legal* instance of $r(R)$, if $t_1.X = t_2.X$ then $t_1.Y = t_2.Y$
- A FD $X \rightarrow Y$ is trivial if it is satisfied for *all* instances of a relation
  - $Y \subseteq X$
- A candidate key functionally determines all attributes

Functional Dependencies

- Functional dependencies (FDs) are *constraints* derived from the meaning of and relationships among attributes
- A set of attributes X functionally determines Y, denoted by $X \rightarrow Y$, if the value of X determines a *unique* value of Y
  - roll → name; roll → batch
- For any two tuples t_1 and t_2 in any *legal* instance of $r(R)$, if $t_1.X = t_2.X$ then $t_1.Y = t_2.Y$
- A FD $X \rightarrow Y$ is trivial if it is satisfied for *all* instances of a relation
  - $Y \subseteq X$
- A candidate key functionally determines all attributes
- Functional dependencies and keys define normal forms for relations
- Normal forms are formal measures of how "good" a database design is

Armstrong's Axioms

- Given a set of FDs, additional FDs can be inferred using Armstrong's inference rules or Armstrong's axioms
  1. Reflexive: If $Y \subseteq X$, then $X \rightarrow Y$
  2. Augmentation: If $X \rightarrow Y$, then $X, Z \rightarrow Y, Z$
  3. Transitive: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

Armstrong's Axioms

- Given a set of FDs, additional FDs can be inferred using Armstrong's inference rules or Armstrong's axioms
  1. Reflexive: If $Y \subseteq X$, then $X \rightarrow Y$
  2. Augmentation: If $X \rightarrow Y$, then $X, Z \rightarrow Y, Z$
  3. Transitive: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- These rules are
  - Sound: Any other rule derived from these holds
  - Complete: Any rule which holds can be derived from these

Armstrong's Axioms

- Given a set of FDs, additional FDs can be inferred using Armstrong's inference rules or Armstrong's axioms
  1. Reflexive: If $Y \subseteq X$, then $X \rightarrow Y$
  2. Augmentation: If $X \rightarrow Y$, then $X, Z \rightarrow Y, Z$
  3. Transitive: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- These rules are
  - Sound: Any other rule derived from these holds
  - Complete: Any rule which holds can be derived from these
- Other inferred rules
  4. Decomposition: If $X \rightarrow Y, Z$, then $X \rightarrow Y$ and $X \rightarrow Z$
  5. Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow Y, Z$
  6. Pseudotransitivity: If $X \rightarrow Y$ and $W, Y \rightarrow Z$, then $W, X \rightarrow Z$

Properties of FDs

- Closure of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X using F^+

Properties of FDs

- Closure of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X using F^+
- F covers G if every FD in G can be inferred from F
- F covers G if $G^+ \subseteq F^+$

Properties of FDs

- Closure of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X using F^+
- F covers G if every FD in G can be inferred from F
- F covers G if $G^+ \subseteq F^+$
- Two sets of FDs F and G are equivalent if every FD in F can be inferred from G and vice versa
- F and G are equivalent if $F^+ = G^+$
- F and G are equivalent if F covers G and G covers F

Properties of FDs

- Closure of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- Closure of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X using F^+
- F covers G if every FD in G can be inferred from F
- F covers G if $G^+ \subseteq F^+$
- Two sets of FDs F and G are equivalent if every FD in F can be inferred from G and vice versa
- F and G are equivalent if $F^+ = G^+$
- F and G are equivalent if F covers G and G covers F
- A set of FDs is minimal if
  - Every FD in F has only a single attribute in RHS
  - Any $G \subset F$ is not equivalent to F
  - Any $F - (X \rightarrow A) \cup (Y \rightarrow A)$ where $Y \subset X$ is not equivalent to F
- Every set of FD has *at least one* equivalent minimal set

Normal Forms

- The process of decomposing relations into smaller relations that conform to certain norms is called normalization
- Keys and FDs of a relation determine which normal form a relation is in
- Different normal forms
  - 1NF: based on attributes only
  - 2NF, 3NF, BCNF: based on keys and FDs
  - 4NF: based on keys and multi-valued dependencies (MVDs)
  - 5NF or PJNF: based on keys and join dependencies
  - DKNF: based on all constraints

First Normal Form (1NF)

- A relation is in 1NF if
  - Every attribute is atomic

First Normal Form (1NF)

- A relation is in 1NF if
  - Every attribute is atomic
- Phone numbers are not atomic

First Normal Form (1NF)

- A relation is in 1NF if
  - Every attribute is atomic
- Phone numbers are not atomic
- Values like “CS315” may not be considered atomic

First Normal Form (1NF)

- A relation is in 1NF if
  - Every attribute is atomic
- Phone numbers are not atomic
- Values like "CS315" may not be considered atomic

<u>Id</u>	Name	Phones
1	A	$\{3, 4\}$
2	B	$\{5\}$

should be

First Normal Form (1NF)

- A relation is in 1NF if
  - Every attribute is atomic
- Phone numbers are not atomic
- Values like “CS315” may not be considered atomic

<u>Id</u>	Name	Phones
1	A	$\{3, 4\}$
2	B	$\{5\}$

should be

<u>Id</u>	Name	<u>Phone</u>
1	A	3
1	A	4
2	B	5

Nested Relations

- Nested relations or composite attributes should be decomposed
- Example

Faculty	Name	Course(CourseId, Title)
11	AB	(1, yz)
12	CD	(2, wx)
13	EF	(2, wx)
13	EF	(3, uv)

should be decomposed into

Nested Relations

- Nested relations or composite attributes should be decomposed
- Example

Faculty	Name	Course(CourseId, Title)
11	AB	(1, yz)
12	CD	(2, wx)
13	EF	(2, wx)
13	EF	(3, uv)

should be decomposed into

Faculty	Name
11	AB
12	CD
13	EF

and

Faculty	CourseId	Title
11	1	yz
12	2	wx
13	2	wx
13	3	uv

Prime Attribute, Full and Transitive FD

- A prime attribute must be a member of some candidate key
  - Example: roll
- A non-prime attribute is not a member of any candidate key
  - Example: gender

Prime Attribute, Full and Transitive FD

- A prime attribute must be a member of some candidate key
  - Example: roll
- A non-prime attribute is not a member of any candidate key
  - Example: gender
- A FD $X \rightarrow Y$ is a full functional dependency if the FD does not hold when any attribute from X is removed
  - Example: (roll, courseid) $\rightarrow$ (grade)
- It is a partial functional dependency otherwise
  - (roll, gender) $\rightarrow$ (name)

Prime Attribute, Full and Transitive FD

- A prime attribute must be a member of some candidate key
  - Example: roll
- A non-prime attribute is not a member of any candidate key
  - Example: gender
- A FD $X \rightarrow Y$ is a full functional dependency if the FD does not hold when any attribute from X is removed
  - Example: (roll, courseid) $\rightarrow$ (grade)
- It is a partial functional dependency otherwise
  - (roll, gender) $\rightarrow$ (name)
- A FD $X \rightarrow Y$ is a transitive functional dependency if it can be derived from two FDs $X \rightarrow Z$ and $Z \rightarrow Y$ where Z is not a set of prime attributes
  - Example: (roll) $\rightarrow$ (hod) since (roll) $\rightarrow$ (dept) and (dept) $\rightarrow$ (hod) hold
- It is non-transitive otherwise
  - Example: (roll) $\rightarrow$ (name)

Second Normal Form (2NF)

- A relation is in 2NF if
  - Every non-prime attribute is fully functionally dependent on every candidate key

Second Normal Form (2NF)

- A relation is in 2NF if
  - Every non-prime attribute is fully functionally dependent on every candidate key
- Alternatively, every attribute should either be
  - In a candidate key or
  - Depend fully on every candidate key

Second Normal Form (2NF)

- A relation is in 2NF if
  - Every non-prime attribute is fully functionally dependent on every candidate key
- Alternatively, every attribute should either be
  - In a candidate key or
  - Depend fully on every candidate key
- Consider (roll, courseid, grade, name, title) with FDs:
  (roll, courseid) $\rightarrow$ (grade); (roll) $\rightarrow$ (name); (courseid) $\rightarrow$ (title)

Second Normal Form (2NF)

- A relation is in 2NF if
  - Every non-prime attribute is fully functionally dependent on every candidate key
- Alternatively, every attribute should either be
  - In a candidate key or
  - Depend fully on every candidate key
- Consider (<u>roll</u>, <u>courseid</u>, grade, name, title) with FDs:
  (roll, courseid) → (grade); (roll) → (name); (courseid) → (title)
- It is not in 2NF since (name) depends partially on (roll), and (title) depends partially on (courseid)
- After 2NF normalization,

Second Normal Form (2NF)

- A relation is in 2NF if
  - Every non-prime attribute is fully functionally dependent on every candidate key
- Alternatively, every attribute should either be
  - In a candidate key or
  - Depend fully on every candidate key
- Consider ($\underline{\text{roll}}$, $\underline{\text{courseid}}$, grade, name, title) with FDs:
  (roll, courseid) → (grade); (roll) → (name); (courseid) → (title)
- It is not in 2NF since (name) depends partially on (roll), and (title) depends partially on (courseid)
- After 2NF normalization,
  - (roll, courseid, grade) with FD: (roll, courseid) → (grade)
  - (roll, name) with FD: (roll) → (name)
  - (courseid, title) with FD: (courseid) → (title)

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Alternatively, for every FD $X \to Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Consider (facultyid, name, courseid, title) with FDs: (facultyid) $\to$ (name, courseid); (courseid) $\to$ (title)

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Consider (facultyid, name, courseid, title) with FDs: (facultyid) $\rightarrow$ (name, courseid); (courseid) $\rightarrow$ (title)
- It is not in 3NF since (title) depends transitively on (facultyid) through (courseid)
- After 3NF normalization,

Third Normal Form (3NF)

- A relation is in 3NF if
  - It is in 2NF, and
  - No non-prime attribute is transitively functionally dependent on the candidate keys
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Consider (<u>facultyid</u>, name, courseid, title) with FDs: (facultyid) → (name, courseid); (courseid) → (title)
- It is not in 3NF since (title) depends transitively on (facultyid) through (courseid)
- After 3NF normalization,
  - (<u>roll</u>, name, courseid) with FD: (facultyid) → (name, courseid)
  - (<u>courseid</u>, title) with FD: (courseid) → (title)

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Also called 3.5NF

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Also called 3.5NF
- BCNF decomposition can *lose* FDs

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Also called 3.5NF
- BCNF decomposition can *lose* FDs
- Consider (Id, Dist, Lot, Area) with FDs:
  (Id) → (Dist, Lot, Area); (Dist, Lot) → (Id, Area); (Area) → (Dist)

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Also called 3.5NF
- BCNF decomposition can *lose* FDs
- Consider (Id, Dist, Lot, Area) with FDs:
  (Id) → (Dist, Lot, Area); (Dist, Lot) → (Id, Area); (Area) → (Dist)
- It is not in BCNF since (Area) is not a superkey although (Area) → (Dist) holds
- After BCNF normalization,

Boyce-Codd Normal Form (BCNF)

- A relation is in BCNF
  - If $X \rightarrow Y$ is a non-trivial FD, then X is a superkey of R
- Alternatively, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Also called 3.5NF
- BCNF decomposition can *lose* FDs
- Consider (<u>Id</u>, Dist, Lot, Area) with FDs:
  (Id) → (Dist, Lot, Area); (Dist, Lot) → (Id, Area); (Area) → (Dist)
- It is not in BCNF since (Area) is not a superkey although (Area) → (Dist) holds
- After BCNF normalization,
  - (<u>Id</u>, Lot, Area) with FD: (Id) → (Dist, Lot, Area)
  - (Dist, <u>Area</u>) with FD: (Area) → (Dist)
- Loses the FD (Dist, Lot) → (Id, Area)

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF
- Consider (city, showroom, mall) with FDs:
  (mall) $\rightarrow$ (city); (city, showroom) $\rightarrow$ (mall)

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF
- Consider (<u>city</u>, <u>showroom</u>, mall) with FDs: (mall) $\rightarrow$ ($\overline{\text{city}}$); (city, showroom) $\rightarrow$ (mall)
- Candidate keys are (city, showroom) and (mall, showroom) but not (city, mall)

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF
- Consider (city, showroom, mall) with FDs:
  (mall) → (city); (city, showroom) → (mall)
- Candidate keys are (city, showroom) and (mall, showroom) but not (city, mall)
- Not in BCNF as (mall) → (city) violates with (mall) not being a superkey

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF
- Consider (city, showroom, mall) with FDs:
  (mall) → (city); (city, showroom) → (mall)
- Candidate keys are (city, showroom) and (mall, showroom) but not (city, mall)
- Not in BCNF as (mall) → (city) violates with (mall) not being a superkey
- BCNF decomposition is *not possible* while guaranting both losslessness and dependency preservation

BCNF versus 3NF

- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Difference lies in one condition
  - $X \rightarrow Y$ where Y is prime is allowed in 3NF
- Consider (city, showroom, mall) with FDs:
  (mall) → (city); (city, showroom) → (mall)
- Candidate keys are (city, showroom) and (mall, showroom) but not (city, mall)
- Not in BCNF as (mall) → (city) violates with (mall) not being a superkey
- BCNF decomposition is *not possible* while guaranting both losslessness and dependency preservation
- Therefore, "good" design ensures either BCNF or its relaxation, i.e., 3NF

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- According to rule, (<u>mall</u>, city) and (<u>showroom</u>, <u>city</u>)

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- According to rule, (<u>mall</u>, city) and (<u>showroom</u>, <u>city</u>)

mall	city
iit	kanpur
zsq	kanpur
quest	kolkata

showroom	city
tata	kanpur
maruti	kanpur
tata	kolkata

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- According to rule, (mall, city) and (showroom, city)

mall	city
iit	kanpur
zsq	kanpur
quest	kolkata

showroom	city
tata	kanpur
maruti	kanpur
tata	kolkata

Joining produces

showroom	city	mall
tata	kanpur	iit
tata	kanpur	zsq
maruti	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (mall, showroom) and (showroom, city)

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (mall, showroom) and (showroom, city)

mall	showroom
iit	tata
quest	tata
zsq	maruti

showroom	city
tata	kanpur
maruti	kanpur
tata	kolkata

Lossy Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (<u>mall</u>, <u>showroom</u>) and (<u>showroom</u>, <u>city</u>)

mall	showroom
iit	tata
quest	tata
zsq	maruti

showroom	city
tata	kanpur
maruti	kanpur
tata	kolkata

Joining produces

showroom	city	mall
tata	kanpur	iit
tata	kanpur	quest
maruti	kanpur	zsq
tata	kolkata	iit
tata	kolkata	quest

Lossless Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

Lossless Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (<u>mall</u>, <u>showroom</u>) and (<u>mall</u>, <u>city</u>)

Lossless Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (mall, showroom) and (mall, city)

mall	showroom
iit	tata
quest	tata
zsq	maruti

mall	city
iit	kanpur
zsq	kanpur
quest	kolkata

Lossless Decomposition

showroom	city	mall
tata	kanpur	iit
maruti	kanpur	zsq
tata	kolkata	quest

- If (<u>mall</u>, <u>showroom</u>) and (<u>mall</u>, <u>city</u>)

mall	showroom
iit	tata
quest	tata
zsq	maruti

mall	city
iit	kanpur
zsq	kanpur
quest	kolkata

Joining *correctly* produces

showroom	city	mall
tata	kanpur	iit
tata	kolkata	quest
maruti	kanpur	zsq

Loss of Functional Dependency

- Possible decomposition is
  - (mall, city) and (mall, showroom)

Loss of Functional Dependency

- Possible decomposition is
  - (<u>mall</u>, city) and (<u>mall</u>, <u>showroom</u>)
- FD (city, showroom) → (mall) is lost

Loss of Functional Dependency

- Possible decomposition is
  - (<u>mall</u>, city) and (<u>mall</u>, <u>showroom</u>)
- FD (city, showroom) → (mall) is lost
- This is *allowed*

mall	city
iit	kanpur
zsq	kanpur

mall	showroom
iit	tata
zsq	tata

Loss of Functional Dependency

- Possible decomposition is
  - (mall, city) and (mall, showroom)
- FD (city, showroom) → (mall) is lost
- This is *allowed*

mall	city
iit	kanpur
zsq	kanpur

mall	showroom
iit	tata
zsq	tata

although joining them produces

mall	city	showroom
iit	kanpur	tata
zsq	kanpur	tata

that *violates* the FD (city, showroom) → (mall)

Example of Normalization

- L = (<u>Id</u>, Dist, Lot, Area, Price, Rate) with FDs:
  - (Id) $\rightarrow$ (Dist, Lot, Area, Price, Rate)
  - (Dist, Lot) $\rightarrow$ (Id, Area, Price, Rate)
  - (Dist) $\rightarrow$ (Rate)
  - (Area) $\rightarrow$ (Price)

Example of Normalization

- L = (<u>Id</u>, Dist, Lot, Area, Price, Rate) with FDs:
  - (Id) → (Dist, Lot, Area, Price, Rate)
  - (Dist, Lot) → (Id, Area, Price, Rate)
  - (Dist) → (Rate)
  - (Area) → (Price)
- L is not in 2NF because (Rate) depends partially on (Dist)

Example of Normalization

- L = (<u>Id</u>, Dist, Lot, Area, Price, Rate) with FDs:
  - (Id) → (Dist, Lot, Area, Price, Rate)
  - (Dist, Lot) → (Id, Area, Price, Rate)
  - (Dist) → (Rate)
  - (Area) → (Price)
- L is not in 2NF because (Rate) depends partially on (Dist)
- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) → (Dist, Lot, Area, Price)
  - (Dist, Lot) → (Id, Area, Price)
  - (Area) → (Price)
- L_2 = (<u>Dist</u>, Rate) with FD:
  - (Dist) → (Rate)

Example of Normalization

- L = (<u>Id</u>, Dist, Lot, Area, Price, Rate) with FDs:
  - (Id) → (Dist, Lot, Area, Price, Rate)
  - (Dist, Lot) → (Id, Area, Price, Rate)
  - (Dist) → (Rate)
  - (Area) → (Price)
- L is not in 2NF because (Rate) depends partially on (Dist)
- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) → (Dist, Lot, Area, Price)
  - (Dist, Lot) → (Id, Area, Price)
  - (Area) → (Price)
- L_2 = (<u>Dist</u>, Rate) with FD:
  - (Dist) → (Rate)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)

Example of Normalization

- L = (<u>Id</u>, Dist, Lot, Area, Price, Rate) with FDs:
  - (Id) → (Dist, Lot, Area, Price, Rate)
  - (Dist, Lot) → (Id, Area, Price, Rate)
  - (Dist) → (Rate)
  - (Area) → (Price)
- L is not in 2NF because (Rate) depends partially on (Dist)
- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) → (Dist, Lot, Area, Price)
  - (Dist, Lot) → (Id, Area, Price)
  - (Area) → (Price)
- L_2 = (<u>Dist</u>, Rate) with FD:
  - (Dist) → (Rate)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)
- L_2 is in 2NF and in 3NF

Example (contd.)

- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) → (Dist, Lot, Area, Price)
  - (Dist, Lot) → (Id, Area, Price)
  - (Area) → (Price)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)

Example (contd.)

- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) → (Dist, Lot, Area, Price)
  - (Dist, Lot) → (Id, Area, Price)
  - (Area) → (Price)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)
- L_{11} = (<u>Id</u>, Dist, Lot, Area) with FDs:
  - (Id) → (Dist, Lot, Area)
  - (Dist, Lot) → (Id, Area)
- L_{12} = (<u>Area</u>, Price) with FD:
  - (Area) → (Price)

Example (contd.)

- L_1 = (<u>Id</u>, Dist, Lot, Area, Price) with FDs:
  - (Id) $\rightarrow$ (Dist, Lot, Area, Price)
  - (Dist, Lot) $\rightarrow$ (Id, Area, Price)
  - (Area) $\rightarrow$ (Price)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)
- L_{11} = (<u>Id</u>, Dist, Lot, Area) with FDs:
  - (Id) $\rightarrow$ (Dist, Lot, Area)
  - (Dist, Lot) $\rightarrow$ (Id, Area)
- L_{12} = (<u>Area</u>, Price) with FD:
  - (Area) $\rightarrow$ (Price)
- L_{11} and L_{12} are in 3NF

Normal Forms: Tests

- 1NF: The relation should have no multivalued attributes or nested relations

Normal Forms: Tests

- 1NF: The relation should have no multivalued attributes or nested relations
- 2NF: For a relation where candidate key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the candidate key

Normal Forms: Tests

- 1NF: The relation should have no multivalued attributes or nested relations
- 2NF: For a relation where candidate key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the candidate key
- 3NF: The relation should not have a nonkey attribute functionally determined by a set of nonkey attributes

Normal Forms: Tests

- 1NF: The relation should have no multivalued attributes or nested relations
- 2NF: For a relation where candidate key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the candidate key
- 3NF: The relation should not have a nonkey attribute functionally determined by a set of nonkey attributes
- BCNF: The relation should not have an attribute functionally determined by a set of nonkey attributes

Normal Forms: Remedies

- 1NF: Form new relations for each multi-valued attribute or nested relation

Normal Forms: Remedies

- 1NF: Form new relations for each multi-valued attribute or nested relation
- 2NF: Decompose and set up a relation for each partial key with its dependent(s); retain the primary key and attributes fully dependent on it

Normal Forms: Remedies

- 1NF: Form new relations for each multi-valued attribute or nested relation
- 2NF: Decompose and set up a relation for each partial key with its dependent(s); retain the primary key and attributes fully dependent on it
- 3NF: Decompose and set up a relation for each transitively dependent nonkey attribute with nonkey attributes that it functionally depends upon

Normal Forms: Remedies

- 1NF: Form new relations for each multi-valued attribute or nested relation
- 2NF: Decompose and set up a relation for each partial key with its dependent(s); retain the primary key and attributes fully dependent on it
- 3NF: Decompose and set up a relation for each transitively dependent nonkey attribute with nonkey attributes that it functionally depends upon
- BCNF: Decompose and set up a relation for each nonkey attribute with attributes functionally dependent on it

Anomalies with BCNF

- Consider (<u>course</u>, <u>teacher</u>, <u>book</u>)
  - (c, t, b): t can teach c, and b is a textbook for c
- No other FD
- Therefore, relation is in BCNF

Anomalies with BCNF

- Consider (<u>course</u>, <u>teacher</u>, <u>book</u>)
  - (c, t, b): t can teach c, and b is a textbook for c
- No other FD
- Therefore, relation is in BCNF

course	teacher	book
C1	AB	B1
C1	AB	B2
C1	CD	B1
C1	CD	B2
C2	EF	B3
C2	EF	B4
C2	AB	B3
C2	AB	B4

Anomalies with BCNF

- Consider (course, teacher, book)
  - (c, t, b): t can teach c, and b is a textbook for c
- No other FD
- Therefore, relation is in BCNF

course	teacher	book
C1	AB	B1
C1	AB	B2
C1	CD	B1
C1	CD	B2
C2	EF	B3
C2	EF	B4
C2	AB	B3
C2	AB	B4

- Modification anomalies are still there
  - Inserting a new teacher for C1 requires two tuples

Anomalies with BCNF

- Consider (course, teacher, book)
  - (c, t, b): t can teach c, and b is a textbook for c
- No other FD
- Therefore, relation is in BCNF

course	teacher	book
C1	AB	B1
C1	AB	B2
C1	CD	B1
C1	CD	B2
C2	EF	B3
C2	EF	B4
C2	AB	B3
C2	AB	B4

- Modification anomalies are still there
  - Inserting a new teacher for C1 requires two tuples
- Better design if (course, teacher) and (course, book)

Multi-Valued Dependency (MVD)

- A multi-valued dependency (MVD) $X \twoheadrightarrow Y$ holds for a relation schema R if for all *legal* relations $r(R)$, if for a pair of tuples t_1 and t_2, $t_1.X = t_2.X$, then there exists another pair of tuples t_3 and t_4
  - $t_1.X = t_2.X = t_3.X = t_4.X$
  - $t_3.Y = t_1.Y$
  - $t_3.R - Y - X = t_2.R - Y - X$
  - $t_4.Y = t_2.Y$
  - $t_4.R - Y - X = t_1.R - Y - X$

	X	Y	R - Y - X
t_1	a	b	c
t_2	a	d	e
t_3	a	b	e
t_4	a	d	c

- Example: (course) $\twoheadrightarrow$ (teacher) in (course, teacher, book)
  - If (C1, AB, B1) and (C1, CD, B2) exist, then (C1, AB, B2) and (C1, CD, B1) must exist
  - Otherwise, AB (resp. CD) has something special to do with B1 (resp. B2)

MVD and Lossless Join

- $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow R - Y - X$

MVD and Lossless Join

- $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow R - Y - X$
- R = (<u>X</u>, <u>Y</u>, <u>Z</u>)
- X ↠ Y, and by symmetry, X ↠ Z
- Then, decomposition into (X, Y) and (X, Z) will be lossless
- For any relation $r = \Pi_{X,Y}(r) \bowtie \Pi_{X,Z}(r)$

MVD and Lossless Join

- $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow R - Y - X$
- R = (X, Y, Z)
- X ↠ Y, and by symmetry, X ↠ Z
- Then, decomposition into (X, Y) and (X, Z) will be lossless
- For any relation $r = \Pi_{X,Y}(r) \bowtie \Pi_{X,Z}(r)$
- A MVD $X \twoheadrightarrow Y$ on R is trivial if either $Y \subseteq X$ or $R = X \cup Y$
- It is non-trivial otherwise

MVD and Lossless Join

- $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow R - Y - X$
- R = ($\underline{X}$, $\underline{Y}$, $\underline{Z}$)
- X ↠ Y, and by symmetry, X ↠ Z
- Then, decomposition into (X, Y) and (X, Z) will be lossless
- For any relation $r = \Pi_{X,Y}(r) \bowtie \Pi_{X,Z}(r)$
- A MVD $X \twoheadrightarrow Y$ on R is trivial if either $Y \subseteq X$ or $R = X \cup Y$
- It is non-trivial otherwise
- Closure of a set of MVDs is the set of all MVDs that can be inferred

Fourth Normal Form (4NF)

- A relation is in 4NF
  - If $X \twoheadrightarrow Y$ is a non-trivial MVD, then X is a superkey of R
- Alternatively, for every MVD $X \twoheadrightarrow Y$, either
  - It is trivial, or
  - X is a superkey

Fourth Normal Form (4NF)

- A relation is in 4NF
  - If $X \twoheadrightarrow Y$ is a non-trivial MVD, then X is a superkey of R
- Alternatively, for every MVD $X \twoheadrightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Every 4NF relation is in BCNF
- Consider (<u>course</u>, <u>teacher</u>, <u>book</u>) with MVD: course $\twoheadrightarrow$ book

Fourth Normal Form (4NF)

- A relation is in 4NF
  - If $X \twoheadrightarrow Y$ is a non-trivial MVD, then X is a superkey of R
- Alternatively, for every MVD $X \twoheadrightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Every 4NF relation is in BCNF
- Consider (<u>course</u>, <u>teacher</u>, <u>book</u>) with MVD: course $\twoheadrightarrow$ book
- It is not in 4NF since (course) is not a superkey
- After 4NF normalization,

Fourth Normal Form (4NF)

- A relation is in 4NF
  - If $X \twoheadrightarrow Y$ is a non-trivial MVD, then X is a superkey of R
- Alternatively, for every MVD $X \twoheadrightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Every 4NF relation is in BCNF
- Consider (<u>course</u>, <u>teacher</u>, <u>book</u>) with MVD: course $\twoheadrightarrow$ book
- It is not in 4NF since (course) is not a superkey
- After 4NF normalization,
  - (<u>course</u>, <u>book</u>) with trivial MVD: (course) $\twoheadrightarrow$ (book)
  - (<u>course</u>, <u>teacher</u>) with trivial MVD: (course) $\twoheadrightarrow$ (teacher)
- Decompose R with X $\twoheadrightarrow$ Y into (X,Y) and (X,R-Y-X)

Fourth Normal Form (4NF)

- A relation is in 4NF
  - If $X \twoheadrightarrow Y$ is a non-trivial MVD, then X is a superkey of R
- Alternatively, for every MVD $X \twoheadrightarrow Y$, either
  - It is trivial, or
  - X is a superkey
- Every 4NF relation is in BCNF
- Consider (course, teacher, book) with MVD: course $\twoheadrightarrow$ book
- It is not in 4NF since (course) is not a superkey
- After 4NF normalization,
  - (course, book) with trivial MVD: (course) $\twoheadrightarrow$ (book)
  - (course, teacher) with trivial MVD: (course) $\twoheadrightarrow$ (teacher)
- Decompose R with X $\twoheadrightarrow$ Y into (X,Y) and (X,R-Y-X)

course	teacher
C1	AB
C1	CD
C2	EF
C2	AB

course	book
C1	B1
C1	B2
C2	B3
C2	B4

Join Dependency (JD)

- General way of decomposing a relation into multi-way joins
- A join dependency (JD) $(R_1 \subseteq R, \ldots, R_n \subseteq R)$ holds for a schema R if for all *legal* relations $r(R)$,

$$\bowtie_{i=1}^{n} (\Pi_{R_i}(r)) = r$$

Join Dependency (JD)

- General way of decomposing a relation into multi-way joins
- A join dependency (JD) $(R_1 \subseteq R, \ldots, R_n \subseteq R)$ holds for a schema R if for all *legal* relations $r(R)$,

$$\bowtie_{i=1}^{n} (\Pi_{R_i}(r)) = r$$

- A JD is trivial if one of R_i is R itself

Join Dependency (JD)

- General way of decomposing a relation into multi-way joins
- A join dependency (JD) $(R_1 \subseteq R, \ldots, R_n \subseteq R)$ holds for a schema R if for all *legal* relations $r(R)$,

$$\bowtie_{i=1}^{n} (\Pi_{R_i}(r)) = r$$

- A JD is trivial if one of R_i is R itself

Salesman	Brand	Product
J	A	V
J	A	B
W	A	V
W	A	B
W	R	V
W	R	P

- If S sells products of brand B and if S sells product type P, then S *must* sell product type P of brand B (assuming B makes P)
- This means that (S,B) $\bowtie$ (B,P) $\bowtie$ (P,S) is equal to (S,B,P)

Join Dependency (JD)

- General way of decomposing a relation into multi-way joins
- A join dependency (JD) ($R_1 \subseteq R, \ldots, R_n \subseteq R$) holds for a schema R if for all *legal* relations $r(R)$,

$$\bowtie_{i=1}^{n} (\Pi_{R_i}(r)) = r$$

- A JD is trivial if one of R_i is R itself

Salesman	Brand	Product
J	A	V
J	A	B
W	A	V
W	A	B
W	R	V
W	R	P

- If S sells products of brand B and if S sells product type P, then S *must* sell product type P of brand B (assuming B makes P)
- This means that (S,B) $\bowtie$ (B,P) $\bowtie$ (P,S) is equal to (S,B,P)
- A MVD is a special case of JD with $n = 2$

Fifth Normal Form (5NF) or Project-Join Normal Form (PJNF)

- A relation is in 5NF or PJNF
  - If $(R_1, \ldots, R_n)$ is a non-trivial JD, then *every* R_i is a superkey of R

Fifth Normal Form (5NF) or Project-Join Normal Form (PJNF)

- A relation is in 5NF or PJNF
  - If $(R_1, \ldots, R_n)$ is a non-trivial JD, then *every* R_i is a superkey of R
- Consider that J starts selling brand R's products
- Insertion anomaly since multiple tuples need to be inserted

Fifth Normal Form (5NF) or Project-Join Normal Form (PJNF)

- A relation is in 5NF or PJNF
  - If $(R_1, \ldots, R_n)$ is a non-trivial JD, then *every* R_i is a superkey of R
- Consider that J starts selling brand R's products
- Insertion anomaly since multiple tuples need to be inserted
- Better design if broken into three relations (B,P), (S,B), and (P,S)

Brand	Product
A	V
A	B
R	V
R	P

Salesman	Brand
J	A
W	A
W	R

Product	Salesman
V	J
B	J
V	W
B	W
P	W

- Now, insertion requires only one tuple (J, R) in (Salesman, Brand)

Domain-Key Normal Form (DKNF)

- A relation schema is in domain-key normal form (DKNF) if all constraints and relations that should hold can be enforced simply by domain constraints and key constraints
- *Ideal* normal form
- Once a relation is in DKNF, there is no anomaly
- Mostly theoretical