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, 2018-19

Mon 12:00-13:15, Tue 9:00-10:15

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
- Normalization theory answers in the formal manner

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

Lossless Decomposition

- Must preserve losslessness of the corresponding 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

Lossless Decomposition

- Must preserve losslessness of the corresponding 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

- Must preserve losslessness of the corresponding 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

- 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 $\rightarrow$ name
- 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 $\rightarrow$ name
- 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
- 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 must be atomic

First Normal Form (1NF)

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

First Normal Form (1NF)

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

Id	Name	Phones
1	A	$\{3, 4\}$
2	B	$\{5\}$

should be

Id	Name	Phone
1	A	3
1	A	4
2	B	5

Nested Relations

- Nested relations should be decomposed

		Course	
Roll	Name	CourseId	Title
1	A	1	30
1	A	2	20
2	B	2	25
2	B	3	10

should be broken into

Nested Relations

- Nested relations should be decomposed

		Course	
Roll	Name	CourseId	Title
1	A	1	30
1	A	2	20
2	B	2	25
2	B	3	10

should be broken into

Roll	Name
1	A
2	B

and

Roll	CourseId	Title
1	1	30
1	2	20
2	2	25
2	3	10

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 $X \to Y$ is a full functional dependency if the FD does not hold when any attribute from X is removed
  - Example: (roll, courseid) $\to$ (grade)
- It is a partial functional dependency otherwise
  - (roll, gender) $\to$ (name)
- A FD $X \to Y$ is a transitive functional dependency if it can be derived from two FDs $X \to Z$ and $Z \to Y$
  - Example: (roll) $\to$ (title) since (roll) $\to$ (courseid) and (courseid) $\to$ (title) hold
- It is non-transitive otherwise
  - Example: (roll) $\to$ (name)

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 ($\underline{\text{Id}}$, $\underline{\text{ProjId}}$, Hrs, Name, ProjName) with FDs: (Id, ProjId) $\rightarrow$ (Hrs); (Id) $\rightarrow$ (Name); (ProjId) $\rightarrow$ (ProjName)

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 (Id, ProjId, Hrs, Name, ProjName) with FDs: (Id, ProjId) $\rightarrow$ (Hrs); (Id) $\rightarrow$ (Name); (ProjId) $\rightarrow$ (ProjName)
- It is not in 2NF since (Name) depends partially on (Id, ProjId)
- 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 (<u>Id</u>, <u>ProjId</u>, Hrs, Name, ProjName) with FDs: (Id, ProjId) $\rightarrow$ (Hrs); (Id) $\rightarrow$ (Name); (ProjId) $\rightarrow$ (ProjName)
- It is not in 2NF since (Name) depends partially on (Id, ProjId)
- After 2NF normalization,
  - (Id, ProjId, Hrs) with FD: (Id, ProjId) $\rightarrow$ (Hrs)
  - (Id, Name) with FD: (Id) $\rightarrow$ (Name)
  - (ProjId, ProjName) with FD: (ProjId) $\rightarrow$ (ProjName)

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, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- 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, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Consider (<u>Id</u>, Name, ProjId, ProjName) with FDs:
  (Id) $\rightarrow$ (Name, ProjId); (ProjId) $\rightarrow$ (ProjName)

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, for every FD $X \to Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Consider (<u>Id</u>, Name, ProjId, ProjName) with FDs: (Id) $\to$ (Name, ProjId); (ProjId) $\to$ (ProjName)
- It is not in 3NF since (ProjName) depends transitively on (Id) through (ProjId)
- 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, for every FD $X \rightarrow Y$, either
  - It is trivial, or
  - X is a superkey, or
  - Every attribute in $Y - X$ is prime
- Alternatively, every non-prime attribute should be
  - Fully functionally dependent on every key, and
  - Non-transitively dependent on every key
- Consider (<u>Id</u>, Name, ProjId, ProjName) with FDs: (Id) → (Name, ProjId); (ProjId) → (ProjName)
- It is not in 3NF since (ProjName) depends transitively on (Id) through (ProjId)
- After 3NF normalization,
  - (<u>Id</u>, Name, ProjId) with FD: (Id) → (Name, ProjId)
  - (<u>ProjId</u>, ProjName) with FD: (ProjId) → (ProjName)

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- Tests
  - 1NF: The relation should have no multivalued attributes or nested relations

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- 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

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- 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

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- 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
- Remedies
  - 1NF: Form new relations for each multi-valued attribute or nested relation

Normal forms

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- 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
- 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

- Informally
  - 1NF: All attributes depend on *the key*
  - 2NF: All attributes depend on *the whole key*
  - 3NF: All attributes depend on *nothing but the key*
- 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
- 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 nonkey attribute

Example

- 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)

Example

- 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

- 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

- 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)
- L is not in 2NF because (Rate) depends partially on (Dist)
- 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_2 = (<u>Dist</u>, Rate) with FD:
  - (Dist) $\rightarrow$ (Rate)
- L_1 is in 2NF but not 3NF because (Price) depends on (Id) through (Area)

Example

- 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 = (Id, 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 = (Id, 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} = (Id, Dist, Lot, Area) with FDs:
  - (Id) $\rightarrow$ (Dist, Lot, Area)
  - (Dist, Lot) $\rightarrow$ (Id, Area)
- L_{12} = (Area, Price) with FD:
  - (Area) $\rightarrow$ (Price)

Example (contd.)

- L_1 = ($\underline{\text{Id}}$, 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} = ($\underline{\text{Id}}$, Dist, Lot, Area) with FDs:
  - (Id) $\rightarrow$ (Dist, Lot, Area)
  - (Dist, Lot) $\rightarrow$ (Id, Area)
- L_{12} = ($\underline{\text{Area}}$, Price) with FD:
  - (Area) $\rightarrow$ (Price)
- L_{11} and L_{12} are in 3NF

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
- BCNF 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
- BCNF can *lose* FDs
- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- Consider (<u>Id</u>, 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
- BCNF can *lose* FDs
- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- 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,

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
- BCNF can *lose* FDs
- Every BCNF relation is in 3NF
- Good design ensures that every relation is at least in 3NF (if not BCNF)
- 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 (Dist, Lot) → (Id, Area)

Normal forms

- Informally
  - BCNF: Every attribute depends on *only the key*

Normal forms

- Informally
  - BCNF: Every attribute depends on *only the key*
- Test
  - BCNF: The relation should not have an attribute functionally determined by a set of nonkey attributes

Normal forms

- Informally
  - BCNF: Every attribute depends on *only the key*
- Test
  - BCNF: The relation should not have an attribute functionally determined by a set of nonkey attributes
- Remedy
  - BCNF: Decompose and set up a relation for each nonkey attribute with attributes functionally dependent on it

Lossless decomposition

- BCNF decomposition is not always possible
- (town, state, dist) with FDs:
  (town, state) $\rightarrow$ (dist); (dist) $\rightarrow$ (state)

town	state	dist
iit	up	east
iit	wb	mdp
prayag	up	east
prayag	wb	dinaj
kanpur	up	center
lucknow	up	west

Lossless decomposition

- BCNF decomposition is not always possible
- (<u>town</u>, <u>state</u>, dist) with FDs:
  (town, state) $\rightarrow$ (dist); (dist) $\rightarrow$ (state)

town	state	dist
iit	up	east
iit	wb	mdp
prayag	up	east
prayag	wb	dinaj
kanpur	up	center
lucknow	up	west

- According to rule, decomposed into (state, <u>dist</u>) and (<u>town</u>, <u>state</u>)

Lossless decomposition

- BCNF decomposition is not always possible
- (town, state, dist) with FDs:
  (town, state) $\rightarrow$ (dist); (dist) $\rightarrow$ (state)

town	state	dist
iit	up	east
iit	wb	mdp
prayag	up	east
prayag	wb	dinaj
kanpur	up	center
lucknow	up	west

- According to rule, decomposed into (state, dist) and (town, state)
- However, the decomposition is *not* lossless

Lossless decomposition

- BCNF decomposition is not always possible
- (<u>town</u>, <u>state</u>, dist) with FDs:
  (town, state) $\rightarrow$ (dist); (dist) $\rightarrow$ (state)

town	state	dist
iit	up	east
iit	wb	mdp
prayag	up	east
prayag	wb	dinaj
kanpur	up	center
lucknow	up	west

- According to rule, decomposed into (state, <u>dist</u>) and (<u>town</u>, <u>state</u>)
- However, the decomposition is *not* lossless
- Also, (<u>town</u>, <u>state</u>) and (<u>town</u>, <u>dist</u>) is lossy
- Only (<u>town</u>, <u>dist</u>) and (state, <u>dist</u>) is lossless
- Losslessness *must* be preserved

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
db	ab	fdb
db	ab	dbm
db	sg	fdb
db	sg	dbm
nt	rm	ntb
nt	rm	usc
nt	ab	ntb
nt	ab	usc

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
db	ab	fdb
db	ab	dbm
db	sg	fdb
db	sg	dbm
nt	rm	ntb
nt	rm	usc
nt	ab	ntb
nt	ab	usc

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

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
db	ab	fdb
db	ab	dbm
db	sg	fdb
db	sg	dbm
nt	rm	ntb
nt	rm	usc
nt	ab	ntb
nt	ab	usc

- Modification anomalies are still there
  - Inserting a new teacher for db 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 (db, ab, fdb) and (db, sg, dbm) exist, then (db, ab, dbm) and (db, sg, fdb) must exist
  - Otherwise, ab has something to do with fdb

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 = ($\underline{X}$, $\underline{Y}$, $\underline{Z}$)
- X $\twoheadrightarrow$ Y, and by symmetry, X $\twoheadrightarrow$ 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 = (<u>X</u>, <u>Y</u>, <u>Z</u>)
- X $\twoheadrightarrow$ Y, and by symmetry, X $\twoheadrightarrow$ 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 = (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
- Closure of a set of MVDs is the set of all MVDs that can be inferred using the following rules

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 (course, teacher, book) 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 (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)

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)
- Good design ensures that every relation is in 3NF or BCNF

Join dependency (JD)

- General way of decomposing a relation into multi-way joins
- A join dependency (JD) $(R_1, \ldots, R_n)$ holds for a relation 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	R	P
W	R	V
W	R	B
W	A	V
W	A	B

- Suppose, the following rule holds: 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) $(R_1, \ldots, R_n)$ holds for a relation 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	R	P
W	R	V
W	R	B
W	A	V
W	A	B

- Suppose, the following rule holds: 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
  - 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
  - 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
  - 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	P
R	V
R	B

Salesman	Brand
J	A
W	R
W	A

Product	Salesman
V	J
B	J
P	W
V	W
B	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
- Mostly theoretical
- Once a relation is in DKNF, there is no anomaly and FDs and MVDs need not be checked any more