Database Systems
Spring 2013

Relational Database Design
SL05

- Relational Database Design Goals
- Functional Dependencies
- 1NF, 2NF, 3NF, BCNF
- Dependency Preservation, Lossless Join Decomposition
- Multivalued Dependencies, 4NF

Literature and Acknowledgments

Reading List for SL05:

- Database Systems, Chapters 14 and 15, Sixth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Education, 2010.

These slides were developed by:

- Michael Böhlen, University of Zürich, Switzerland
- Johann Gamper, Free University of Bozen-Bolzano, Italy

The slides are based on the following text books and associated material:

- Fundamentals of Database Systems, Fourth Edition, Ramez Elmasri and Shamkant B. Navathe, Pearson Addison Wesley, 2004.
- A. Silberschatz, H. Korth, and S. Sudarshan: Database System Concepts, McGraw Hill, 2006.

Relational Database Design Guidelines

- Goals of Relational Database Design
- Update Anomalies

Relational Database Design Guidelines/1

- The goal of relational database design is to find a good collection of relation schemas.
- The main problem is to find a good grouping of the attributes into relation schemas.
- We have a good collection of relation schemas if we
  - Ensure a simple semantics of tuples and attributes
  - Avoid redundant data
  - Avoid update anomalies
  - Avoid null values as much as possible
  - Ensure that exactly the original data is recorded and (natural) joins do not generate spurious tuples

Relational Database Design Guidelines/2

- Consider the relation:
  - empproj(SSN, PNum, Hours, EName, PName, PLoc)
- Update Anomaly:
  - Changing the name of project location "Houston" to "Dallas" for an employee forces us to make this change for all other employees working on this project.
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it (except by using null values).
- Delete Anomaly:
  - When a project is deleted, it will result in deleting all the employees who work on that project.

Relational Database Design Guidelines/3

- Consider relation schema
  empproj(SSN, PNum, Hours, EName, PName, Ploc)
  with instance

empproj

SSN	PNum	Hours	EName	PName	PLoc
1234	1	32.5	Smith	ProductX	Bellaire
1234	2	7.5	Smith	ProductY	Sugarland
6688	3	40.5	Narayan	ProductZ	Houston
4567	1	20.0	English	ProductX	Bellaire
4567	2	20.0	English	ProductY	Sugarland
3334	2	10.0	Wong	ProductY	Sugarland
3334	3	10.0	Wong	ProductZ	Houston
3334	10	10.0	Wong	Computerization	Stafford
3334	20	10.0	Wong	Reorganization	Houston

- Relation schema empproj is not a good schema and suffers from update anomalies.

Relational Database Design Guidelines/4

- **Guideline 1**: Each tuple in a relation should only represent one entity or relationship instance.
- **Guideline 2**: Design a schema that does not suffer from insertion, deletion and update anomalies.
- **Guideline 3**: Relations should be designed such that their tuples will have as few NULL values as possible; attributes that are NULL shall be placed in separate relations (along with the primary key).
- **Guideline 4**: Relations should be designed such that no spurious (i.e., wrong) tuples are generated if we do a natural join of the relations.

Functional Dependencies

- Definition
- Armstrong's inference rules
- Soundness and completeness
- Closure and minimal cover

Keys (Refresher)

- A **superkey** of a relation schema $R = A_1, A_2, \ldots, A_n$ is a set of attributes $S \subseteq R$ with the property that no two tuples t1 and t2 in any legal relation state r of R will have $t_1[S] = t_2[S]$
- A **candidate key** K is a *superkey with the additional property* that removal of any attribute from K will cause the reduced K not to be a superkey any more.
- One of the candidate keys is *arbitrarily* chosen to be the **primary key**.
- Notation: We underline the primary key attributes: empproj(SSN, PNum, Hours, EName, PName, Ploc)

Functional Dependencies/1

- Functional dependencies (FDs) are used to specify *formal measures* of the goodness of relational designs.
- Functional dependencies and keys are used to define **normal forms** for relations.
- Functional dependencies are **constraints** that are derived from the *meaning* and *interrelationships* of the attributes.
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y.
- A functional dependency $X \rightarrow Y$ is **trivial** iff $Y \subseteq X$.

Functional Dependencies/2

- $X \rightarrow Y$ denotes a functional dependency.
- $X \rightarrow Y$ means that X functionally determines Y.
- $X \rightarrow Y$ holds if whenever two tuples have the same value for X they have the same value for Y.
  - For any two tuples t_1 and t_2 in any relation instance $r(R)$: If $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$
- $X \rightarrow Y$ in R specifies a *constraint* on the schema, i.e., on all possible relation instances r(R).
- FDs are derived from the real-world constraints on the attributes.
- Notation: instead of $\{A, B\}$ we write AB (or A, B), e.g., $AB \rightarrow BCD$ (instead of $\{A, B\} \rightarrow \{B, C, D\}$)

Review 5.1

r

A	B	C
1	1	3
2	1	1
3	2	2
4	1	1

Consider the relation instance $r(R)$ and the statements

1. A is a primary key of R
2. $B \rightarrow C$ is a functional dependency that holds for R
3. $C \rightarrow B$ is a functional dependency that holds for R
4. $BC \rightarrow A$ is a functional dependency that holds for r

Which of these statements are true?

Functional Dependencies/3

Examples of FD constraints:

- Social security number determines employee name
  - $SSN \rightarrow ENAME$
- Project number determines project name and location
  - $PNUMBER \rightarrow PNAME, PLOCATION$
- Employee ssn and project number determines the hours per week that the employee works on the project
  - $SSN, PNUMBER \rightarrow HOURS$

Functional Dependencies/4

- A FD is a property of the semantics of the attributes.
- A FD constraint must hold on *every* relation instance r(R)
- If K is a candidate key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with $t_1[K] = t_2[K]$)
- Certain FDs can be ruled out based on a given state of the database:

teach

Teacher	Course	Textbook
Smith	Data Structures	Bertram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

The FD Textbook $\rightarrow$ Course is possible
The FD Teacher $\rightarrow$ Course does not hold

Functional Dependencies/5

- Given a set of FDs F, we can **infer** additional FDs that hold whenever the FDs in F hold
- Armstrong's inference rules (aka Armstrong's axioms):
  - Reflexivity: $Y \subseteq X \models X \rightarrow Y$
  - Augmentation: $X \rightarrow Y \models XZ \rightarrow YZ$
  - Transitivity: $X \rightarrow Y, Y \rightarrow Z \models X \rightarrow Z$
- Notation:
  - $A \models B$ means that from A we can infer B
  - XZ stands for $X \cup Z$
- Armstrong's inference rules are **sound** and **complete**
  - These rules hold (are correct) and all other rules that hold can be deduced from these

Review 5.2

Prove or disprove the following inferences:

1. $W \rightarrow Y, X \rightarrow Z \models WX \rightarrow Y$

2. $X \rightarrow Y, Z \subseteq Y \models X \rightarrow Z$

3. $X \rightarrow Y, X \rightarrow W, WY \rightarrow Z \models X \rightarrow Z$

Review 5.2

4. $XY \to Z, Y \to W \models XW \to Z$

5. $X \to Z, Y \to W \models X \to Y$

6. $X \to Y, XY \to Z \models X \to Y$

Functional Dependencies/6

- Additional inference rules that are useful:
  - Decomposition: $X \to YZ \models X \to Y, X \to Z$
  - Union: $X \to Y, X \to Z \models X \to YZ$
  - Pseudotransitivity: $X \to Y, WY \to Z \models WX \to Z$
- The last three inference rules, as well as any other inference rules, can be deduced from Armstrong's inference rules (because of the completeness property).

Functional Dependencies/7

- The **closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F.
- The **closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X.
- F^+ and X^+ can be calculated by repeatedly applying Armstrong's inference rules to F and X, respectively.

Functional Dependencies/8

- Two sets of FDs F and G are **equivalent** if:
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$
- Definition: F **covers** G if every FD in G can be inferred from F (i.e., if $G^+ \subseteq F^+$)
- F and G are equivalent if F covers G and G covers F

Review 5.3

Consider $F = \{A \to C, AC \to D, E \to AD, E \to H\}$ and $G = \{A \to CD, E \to AH\}$. Are F and G equivalent?

Functional Dependencies/9

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. No pair of FDs has the same left-hand side.
  2. We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
  3. We cannot replace any dependency $X \to A$ in F with a dependency $Y \to A$, where $Y \subset X$ and still have a set of dependencies that is equivalent to F.
- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- The first condition can also be changed to "every FD has a single attribute for its right-hand side" (Elmasri and Navathe does this). Note: $X \to YZ \equiv X \to Y, X \to Z$

Review 5.4

Consider $R(A, B, C)$ and $F = \{A \to C, A \to B, B \to A\}$. Determine the minimal cover.

Normal Forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)

Normalization/1

- **Normalization:** The process of decomposing bad relations by breaking up their attributes into smaller relations that fulfill the normal forms.
- The normalization process was proposed by Codd in 1972.
- The normalization process applies a series of tests to a relation schema to verify that the schema qualifies for some normal form.
- A normalized database consists of a good collection of relation schemas.

Normalization/2

- 1NF
  - attribute values must be atomic
- 2NF, 3NF, BCNF
  - based on candidate keys and FDs of a relation schema
- 4NF
  - based on candidate keys, multi-valued dependencies (MVDs)
- 5NF
  - based on candidate keys, join dependencies (JDs)
- Additional properties may be needed to ensure a good relational design:
  - Losslessness of the corresponding join (very important and cannot be sacrificed)
  - Preservation of the functional dependencies (less stringent and may be sacrificed)

Normalization/3

- In practice **normalization** is carried out to guarantee that the resulting schemas are of high quality
- The normalization process provides a deep understanding of relations and attributes.
- The database designers *need not* normalize to the highest possible normal form
  - usually they choose 3NF, BCNF or 4NF
  - controlled redundancy is OK/good
- **Denormalization:**
  - The process of storing the join of higher normal form relations as a base relation (which is in a lower normal form since the join destroys the normal form)

First Normal Form (1NF)/1

- Disallows
  - composite attributes
  - multivalued attributes
  - nested relations: attributes whose values for an *individual tuple* are relations
- Often 1NF is considered to be part of the definition of a relation
- The following instance of schema *department*(*DName*, <u>*DNum*</u>, *DMgrSSN*, *DLoc*) is not in 1NF:

department

DName	DNum	DMgrSSN	DLoc
Research	5	334455	{Bellaire, Sugarland, Houston }
Administration	4	987654	{ Stafford }
Headquarters	1	888666	{ Houston }

First Normal Form (1NF)/2

- Remedy to get 1NF: Form new relations for each multivalued attribute or nested relation
- The following instance is the equivalent instance in 1NF:

department

DName	DNum	DMgrSSN	DLoc
Research	5	334455	Bellaire
Research	5	334455	Sugarland
Research	5	334455	Houston
Administration	4	987654	Stafford
Headquarters	1	888666	Houston

Second Normal Form (2NF)/1

- A relation schema R is in **second normal form (2NF)** iff each attribute not contained in a candidate key is not partially functional dependent on a candidate key of R.
- An attribute is *partially functional dependent* on a candidate key if it is functionally dependent on a proper subset of the candidate key.
- The following relation is not in 2NF:

empproj

SSN	PNum	Hours	EName	PName	PLoc
1234	1	32.5	Smith	ProductX	Bellaire
1234	2	7.5	Smith	ProductY	Sugarland
6688	3	40.5	Narayan	ProductZ	Houston
4567	1	20.0	English	ProductX	Bellaire
4567	2	20.0	English	ProductY	Sugarland
3334	2	10.0	Wong	ProductY	Sugarland
3334	3	10.0	Wong	ProductZ	Houston
3334	10	10.0	Wong	Computerization	Stafford
3334	20	10.0	Wong	Reorganization	Houston

Second Normal Form (2NF)/2

- Remedy to get 2NF: Decompose and set up a new relation for each partial key with its dependent attributes. Keep a relation with the original key and any attributes that are functionally dependent on it.
- Consider empproj(SSN, PNum, Hours, EName, PName, PLoc)
  - Candidate key is SSN and PNum which funtionally determine Hours
  - SSN is a partial key with dependent attributes EName
  - PNum is a partial key with dependent attributes PName and PLoc
- 2NF normalization
  - empproj1(SSN, EName)
  - empproj2(PNum, PName, PLoc)
  - empproj3(SSN, PNum, Hours)

Review 5.5

Consider $R(A, B, C)$ and $F = \{A \rightarrow BC, B \rightarrow C\}$. Is R in 2NF? Is R a good schema?

Third Normal Form (3NF)/1

- A relation schema R is in **third normal form (3NF)** iff for all $X \to A \in F^+$ at least one of the following holds:
  - $X \to A$ is trivial
  - X is a superkey for R
  - A is contained in a candidate key of R
- Intuition: "Each non-key attribute must describe the key, the whole key, and nothing but the key." [Bill Kent, CACM 1983]
- A relation that is in 3NF is also in 2NF.

Third Normal Form (3NF)/2

- The following relation with the functional dependencies $SC \to T$ and $T \to C$ is in 3NF:

R

Student	Course	Textbook
Smith	Data Structures	Bertram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

- $SC \to T$ is OK since SC is a candidate key.
- $T \to C$ is OK since C is contained in a candidate key.
- This relation is in 3NF but permits redundant information, which can lead to update anomalies.

Third Normal Form (3NF)/3

- Consider adding a tuple to the above relation:

R

Student	Course	Textbook
Smith	Data Structures	Bertram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz
Jones	Data Structures	Bertram

- The fact that Bertram is a textbook for the Data Structures class is stored twice
- $SC \to T$ and $T \to C$ can be checked by looking at relation R only (dependency preservation, will be discussed later)

Boyce-Codd Normal Form (BCNF)/1

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** iff for all $X \to A \in F^+$ at least one of the following holds:
  - $X \to A$ is trivial
  - X is a superkey for R
- Intuition: "Each attribute must describe the key, the whole key, and nothing but the key." [Chris Date, adaption of Bill Kent for 3NF]
- A relation that is in BCNF is also in 3NF.
- There exist relations that are in 3NF but not in BCNF

Boyce-Codd Normal Form (BCNF)/2

- The following relations with the functional dependencies $SC \rightarrow T$ and $T \rightarrow C$ are in BCNF:

R1

Course	Textbook
Data Structures	Bertram
Data Management	Martin
Compilers	Hoffman
Data Structures	Horowitz

R2

Student	Textbook
Smith	Bertram
Smith	Martin
Hall	Hoffman
Brown	Horowitz
Jones	Horowitz

- $T \rightarrow C$ is OK since T is a candidate key.
- $SC \rightarrow T$ is not considered since it uses attributes from different relations (a functional dependency is a constraint between two sets of attributes in a single relation).

Boyce-Codd Normal Form (BCNF)/3

- With BCNF less redundancy exists but it is no longer possible to check all functional dependencies by looking at one relation only.

R1

Course	Textbook
Data Structures	Bertram
Data Management	Martin
Compilers	Hoffman
Data Structures	Horowitz

R2

Student	Textbook
Smith	Bertram
Smith	Martin
Hall	Hoffman
Brown	Horowitz
Jones	Horowitz
Jones	Bertram

- No information is stored redundantly
- The fact that Jones uses two textbooks for the Data Structures class and therefore $SC \rightarrow T$ does not hold cannot be checked without joining the relations

Review 5.6

Relation R given below is in BCNF:

r

A	B	C
a1	b1	c1
a1	b2	

Assume we know that the functional dependency $A \rightarrow C$ holds. What value can we infer for the value that is missing?

Properties of Decompositions and Normalization Algorithm

- Dependency Preservation
- Lossless Join Decomposition
- BCNF Normalization Algorithm

Multiple Relations

- Relational database design by decomposition:
  - Universal Relation Schema: A relation schema $R = A_1, A_2, \ldots, A_n$ that includes all attributes of the database.
  - Decomposition: decompose the universal relation schema R into a set of relation schemas $D = R1, R2, \ldots, Rm$ that will become the relational database schema by using the functional dependencies.
  - Additional conditions:
    - Each attribute in R will appear in at least one relation schema Ri in the decomposition so that no attributes are lost.
    - Have each individual relation Ri in the decomposition D in 3NF (or higher).
    - **Lossless join decomposition**: ensures that the decomposition does not introduce wrong tuples when relations are joined together.
    - **Dependency preservation**: ensures that all functional dependency can be checked by considering individual relations Ri only.

Dependency Preservation/1

- Given a set of dependencies F on R, the **projection** of F on Ri, denoted by $F|Ri$ where Ri is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ are all contained in R_i.
- Hence, the projection of F on each relation schema Ri in the decomposition D is the set of functional dependencies in F^+, the closure of F, such that all their left- and right-hand side attributes are in Ri.

Dependency Preservation/2

- Dependency Preservation:
  - A decomposition $D = R1, R2, \ldots, Rm$ of R is **dependency-preserving** with respect to F if the union of the projections of F on each Ri in D is equivalent to F; that is

$$(F|R1 \cup \ldots \cup F|Rm)^+ = F^+$$

- It is always possible to find a dependency-preserving decomposition such that each relation is in 3NF.
- It is not always possible to find a dependency-preserving decomposition such that each relation is in BCNF.

Review 5.7

Consider $R(A, B, C, D)$ and $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A, A \rightarrow D\}$. Is the decomposition $R1(A, B)$, $R2(B, C)$, and $R3(C, D)$ dependency preserving?

Lossless Join Decomposition

- A decomposition $D = R1, R2, \ldots, Rm$ of R is a **lossless join decomposition** with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds:

$$\pi_{R1}(r) \bowtie \ldots \bowtie \pi_{Rm}(r) = r$$

- Note: The word loss in lossless refers to loss of information, not to loss of tuples. If a join decomposition is not lossless then new spurious tuples are present in the result of the join.
- R1 and R2 form a lossless join decomposition of R with respect to a set of functional dependencies F iff
  - $(R1 \cap R2) \rightarrow (R1 - R2)$ is in F^+ or
  - $(R1 \cap R2) \rightarrow (R2 - R1)$ is in F^+

Review 5.8

Consider $R(A, B, C)$, $F = \{AB \rightarrow C, C \rightarrow B\}$, $R1(A, C)$, $R2(B, C)$.

1. Is $\{R1, R2\}$ a lossless decomposition of R?
2. Illustrate your answer for $r = \{(\alpha, 0, a), (\beta, 2, b), (\gamma, 1, c), (\alpha, 2, c)\}$.
3. Discuss what happens if we replace tuple $(\alpha, 2, c)$ by $(\alpha, 2, b)$.

Algorithm for BCNF Normalization/1

teach

Student	Course	Textbook
Smith	Data Structures	Bertram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

- Three possible decompositions for relation *teach*
  - {Student, Textbook} and {Student, Course}
  - {Course, Textbook} and {Course, Student}
  - {Textbook, Course} and {Textbook, Student}
- All three decompositions will loose fd1.
  - We have to settle for sacrificing the functional dependency preservation. We cannot sacrifice the lossless join decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join (and, thus, is lossless).

Algorithm for BCNF Normalization/2

1. Set D := {R};
2. **while** a relation schema Q in D is not in BCNF **do**
   find a functional dependency $X \rightarrow Y$ in Q that violates BCNF;
   replace Q in D by two relation schemas $(Q - Y)$ and $(X \cup Y)$;

Assumption: No null values are allowed for the join attributes.

- The result is a lossless join decomposition of R.
- The resulting schemas do not necessarily preserve all dependencies.

Review 5.9

Consider $R(Course, Teacher, Hour, Room, Student, Grade)$ and the following functional dependencies:

- $C \rightarrow T$ each course has only one teacher
- $HR \rightarrow C$ one course in one room at one time
- $HT \rightarrow R$ a teacher can only teach in one room at one time
- $CS \rightarrow G$ students get one grade in one course
- $HS \rightarrow R$ students can be in one room at one time

Decompose the schema into a lossless BCNF.

Discussion of BCNF Normalization

- It is valuable to construct a good schema that is in BCNF.
- The normalization process gives important insights into the properties of the data.
- A potential difficulty is that the database designer must first specify *all* the relevant functional dependencies among the database attributes.
- The normalization algorithms are **not deterministic** in general (e.g., not a unique minimal cover).
- It is not always possible to find a decomposition into relation schemas that preserves dependencies and allows each relation schema in the decomposition to be in BCNF.

3NF versus BCNF

- It is possible to construct a decomposition that is in BCNF and is lossless
- It is possible to construct a decomposition that is in 3NF, is lossless, and preserves dependency.
- It is not always possible to construct a decomposition that is in BCNF, is lossless, and is dependency preserving.

- 3NF allows redundancies that BCNF does not allow.
- BCNF cannot be checked efficiently since multiple relations must be considered for the check.
- The application needs to determine if a BCNF or 3NF decomposition should be chosen.

Multivalued Dependencies

- Definition
- Fourth Normal Form (4NF)

Multivalued Dependencies/1

Definition:

- A **multivalued dependency (MVD)** $X \twoheadrightarrow Y$ on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R:

  If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties, where we use Z to denote $(R - (X \cup Y))$:
  - $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
  - $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
  - $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.

- A MVD $X \twoheadrightarrow Y$ in R is called a **trivial MVD** if $Y \subset X$ or $X \cup Y = R$.

Review 5.10

Consider schema $R(Brand, Product, Country)$. Show an instance that represents the following facts:

- Nike produces shoes and socks
- Nike produces in Taiwan and China
- Ecco produces shoes
- Ecco produces in Denmark and China

Determine the multivalued dependencies on the resulting instance. How must the instance be changed so that the multivalued dependency no longer holds?

Multivalued Dependencies/2

Inference Rules for Functional and Multivalued Dependencies:

- reflexivity FDs: $X \supseteq Y \models X \rightarrow Y$.
- augmentation FDs: $X \rightarrow Y \models XZ \rightarrow YZ$.
- transitivity FDs: $X \rightarrow Y, Y \rightarrow Z \models X \rightarrow Z$.
- complementation: $X \twoheadrightarrow Y \models X \twoheadrightarrow (R - (X \cup Y))$.
- augmentation MVDs: $X \twoheadrightarrow Y, W \supseteq Z \models WX \twoheadrightarrow YZ$.
- transitivity MVDs: $X \twoheadrightarrow Y, Y \twoheadrightarrow Z \models X \twoheadrightarrow (Z - Y)$.
- replication: $X \rightarrow Y \models X \twoheadrightarrow Y$.
- coalescing: $X \twoheadrightarrow Y, \exists W(W \cap Y = \emptyset, W \rightarrow Z, Y \supseteq Z) \models X \rightarrow Z$.

Fourth Normal Form (4NF)/1

Definition:

- A relation schema R with a set of functional and multivalued dependencies F is in **4NF** iff, for every multivalued dependency $X \twoheadrightarrow Y$ in F^+ at least one of the following holds:
  - $X \twoheadrightarrow Y$ is trivial
  - X is a superkey for R
- F^+ is called the **closure** of F and is the complete set of all dependencies (functional or multivalued) that will hold in every relation state r of R that satisfies F.

Fourth Normal Form (4NF)/2

Example of decomposing a relation that is not in 4NF:

1. Relation emp is not in 4NF.
2. Relations *emp_projects* and *emp_dependents* are in 4NF.

emp

EName	PName	DName
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John
Brown	W	Jim
Brown	X	Jim
Brown	Y	Jim
Brown	Z	Jim
Brown	W	Joan
Brown	X	Joan
Brown	Y	Joan
Brown	Z	Joan
Brown	W	Bob
Brown	X	Bob
Brown	Y	Bob
Brown	Z	Bob

emp_projects

EName	PName
Smith	X
Smith	Y
Brown	W
Brown	X
Brown	Y
Brown	Z

emp_dependents

EName	DName
Smith	John
Smith	Anna
Brown	Jim
Brown	Joan
Brown	Bob

Fourth Normal Form (4NF)/3

- If a relation is not in 4NF because of the MVD $X \twoheadrightarrow Y$ we decompose R into $R1(X \cup Y)$ and $R2(R - Y)$.
- Such a decomposition is lossless.
- R1 and R2 form a lossless join decomposition of R with respect to a set of functional and multivalued dependencies iff
  - $(R1 \cap R2) \twoheadrightarrow (R1 - R2)$ or
  - $(R1 \cap R2) \twoheadrightarrow (R2 - R1)$

Summary/1

- Relational database design goal: eliminate redundancy
- Main concept: functional dependencies
- Functional Dependencies (FDs)
  - Definition
  - Armstrong's inference rules: reflexivity, augmentation, transitivity
  - equivalence of sets of FDs
  - minimal sets of FDs
- Approach: Start with all attributes in a single relation and decompose it vertically until all functional dependencies are acceptable

Summary/2

- Normal forms based on candidate keys and FD
  - 1NF, 2NF, 3NF, BCNF
- BCNF normalization algorithm
- Dependency Preservation
  - always possible for 3NF; not always possible for BCNF
- Lossless Join Decomposition
  - always required
- Multivalued dependencies, 4NF