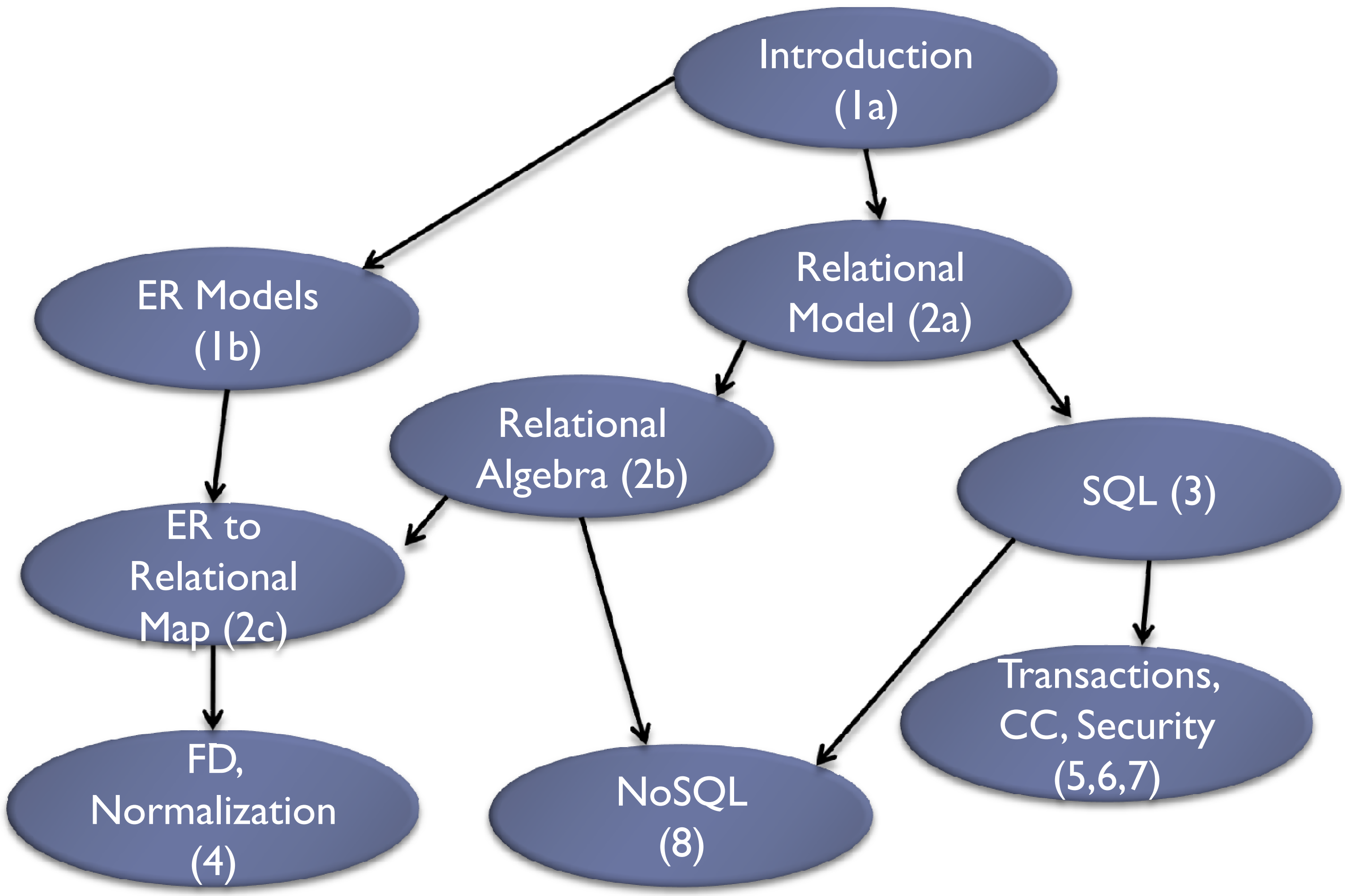
Database Management System (15ECSC208)

UNIT II: Chapter 4: Database Design

Functional Dependencies and Normalization for Relational Database

SG

Contextual Representation of DBMS Course



Informal Design Guidelines for Relation Schemas

Measures of quality

1. Making sure attribute semantics are clear
2. Reducing redundant information in tuples
3. Reducing NULL values in tuples
4. Disallowing possibility of generating spurious tuples

1. Semantics of Relation Attributes

- Semantics of a relation
  - Meaning resulting from interpretation of attribute values in a tuple.

Guideline 1:

- Design relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types and relationship types into a single relation.

2. Redundant Information in Tuples and Update Anomalies

- **Problems with bad schema**

- **Update anomalies**

2. Redundant Information in Tuples and Update Anomalies

- **Problems with bad schema**
  - Redundant storage of data leading into wastage of storage space.
  - Update of data, must change at several spaces leads into more running time and error prone.

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

2. Redundant Information in Tuples and Update Anomalies

- **Types of update anomalies:**
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Redundancy and Update Anomalies

A poorly designed database causes ***anomalies***:

Student	Course	Room
Amar	CSC204	LHC203
Akbar	CSC204	LHC203
Anthony	CSC204	LHC203
..	..	..

If every course is in only one room, contains ***redundant*** information!

Redundancy and Update Anomalies

A poorly designed database causes
update anomaly:

Student	**Course**	**Room**
Amar	CSC204	LHC203
Akbar	CSC204	CL4
Anthony	CSC204	LHC203
..	..	..

If we update the room number for one tuple, we get inconsistent data = an ***update anomaly***

Redundancy and Update Anomalies

A poorly designed database causes
delete anomaly:

Student	Course	Room
..	..	..

If everyone drops the class, we lose what room the class is in! = a ***<u>delete anomaly</u>***

Redundancy and Update Anomalies

A poorly designed database causes
insert anomaly:

Student	Course	Room
Amar	CSC204	LHC203
Akbar	CSC204	LHC203
Anthony	CSC204	LHC203
..	..	..

...	CSC208	CL4

Similarly, we can't reserve a room without students = an ***insert anomaly***

2. Redundant Information in Tuples and Update Anomalies

Guideline 2

- Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
- If any anomalies are present:
  - Note them clearly and make sure that the programs which update the database will operate correctly

Redundancy and Update Anomalies

Student	Course
Amar	CSC204
Akbar	CSC204
Anthony	CSC204
..	..

Course	Room
CSC204	LHC203
CSC208	CL4

Is this form better?

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

title	*year*	*length*	*genre*	*studioName*	*starName*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone With the Wind	1939	231	drama	MGM	Vivien Leigh
Wayne's World	1992	95	comedy	Paramount	Dana Carvey
Wayne's World	1992	95	comedy	Paramount	Mike Meyers

title	*year*	*length*	*genre*	*studioName*
Star Wars	1977	124	sciFi	Fox
Gone With the Wind	1939	231	drama	MGM
Wayne's World	1992	95	comedy	Paramount

title	*year*	*starName*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

3. NULL Values in Tuples

- Problems with NULLs
  - Wasted storage space
  - Problem understanding meaning

Guideline 3

- Avoid placing attributes in a base relation whose values may frequently be NULL.
- If NULLs are unavoidable:
  - Make sure that they apply in exceptional cases only, not to a majority of tuples.

3. NULL Values in Tuples

Multiple interpretations of NULLs

- The attribute does not apply to this tuple.
- The attribute value for this tuple is unknown.
- The value is known but absent; that is, it has not been recorded yet.

4. Generation of Spurious Tuples

- When NATURAL JOIN is applied between relations and if resulting relation produces many more tuples than the original set of tuples, these additional tuples are called **spurious tuples.**
  - Represents spurious information that is not valid

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

EMP_LOCS

ENAME	PLOCATION
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION
123456789	1	32.5	Product X	Bellaire
123456789	2	7.5	Product Y	Sugarland
666884444	3	40.0	Product Z	Houston
453453453	1	20.0	Product X	Bellaire
453453453	2	20.0	Product Y	Sugarland
333445555	2	10.0	Product Y	Sugarland
333445555	3	10.0	Product Z	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston

EMP_PROJ1 * EMP_LOCS

	SSN	PNUMBER	HOURS	PNAME	PLOCATION	ENAME
	123456789	1	32.5	ProductX	Bellaire	Smith,John B.
*	123456789	1	32.5	ProductX	Bellaire	English,Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith,John B.
*	123456789	2	7.5	ProductY	Sugarland	English,Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong,Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan,Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong,Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith,John B.
	453453453	1	20.0	ProductX	Bellaire	English,Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith,John B.
	453453453	2	20.0	ProductY	Sugarland	English,Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong,Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith,John B.
*	333445555	2	10.0	ProductY	Sugarland	English,Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong,Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan,Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong,Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong,Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan,Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong,Franklin T.

* are spurious tuples

4. Generation of Spurious Tuples

Guideline 4:

- Design relation schemas so that they can be joined with equality conditions on attributes that are **either primary keys or foreign keys** in a way that guarantees that **no spurious tuples** are generated.

- **Avoid** relations that contain matching attributes that are **not (foreign key, primary key) combinations**, because joining on such attributes may produce spurious tuples.

Database Design - Overview

- **Relational Design by decomposition**
  - Start with "mega" relations containing everything
  - Decompose into smaller, better relations with same info.
  - Can do decomposition automatically
- Automatic decomposition
  - "Mega" relations + properties of the data
  - System decomposes based on properties
  - Final set of relations satisfies normal form
    - No anomalies, no lost information
- **Properties and Normal Forms**
  - Functional dependencies ⇒ Boyce-Codd Normal Form
  - + Multi-valued dependences ⇒ Fourth Normal Form

Functional Dependencies (FDs)

- Framework for systematic design and optimization of relational schemas
- Generalization over the notion of keys
- Crucial in obtaining correct normalized schemas

- **Functional dependencies are generally useful concept**
  - Data storage – compression
  - Reasoning about queries – optimization

FD Definition

- In any relation R, if there exists a set of attributes (A1, A2, … An) and an attribute B such that if any two tuples have the same value for (A1, A2, … An) then they also have the same value for B.
- A functional dependency (FD) of the above form is written as:

$$A1, A2, \ldots An \rightarrow B$$

- Functional dependencies define properties of the ***schema*** *and not* of any particular instance (tuple). The dependency must hold for all tuples in the schema.

FD Definition

- If (A1, A2, … An) can uniquely determine many attributes, they can all be clubbed together in one expression.

  A1, A2, … An $\rightarrow$ B1

  A1, A2, … An $\rightarrow$ B2

  A1, A2, … An $\rightarrow$ B3

  …

  A1, A2, … An $\rightarrow$ Bm

  $\longrightarrow$

  A1, A2, … An $\rightarrow$ B1B2B3…Bm

FDs with Keys

- If a subset of attributes can uniquely determine the entire tuple, then they are called ***keys***.

- Consider Relation with no duplicates
- And Suppose A $\rightarrow$ all attributes
- Then **A is the key.**

FD Example

- **Ex1:**

  ZIPCodes(ZIP Code, City, County, State Abbreviation, State Name)

  ZIP Code → {City, County, State Abbreviation}

  State Abbreviation → State Name

- **Ex2:**

  EMPID → ENAME

  PNUMBER → {PNAME, PLOCATION}

  {EMPID, PNUMBER} → HOURS

Functional Dependencies - Summary

- A functional dependency (FD) $X \rightarrow Y$(read as X determines Y) ($X \subseteq R$, $Y \subseteq R$) is said to hold on a schema R if in any instance r on R, if two tuples t1, t2 ($t1 \neq t2$, $t1 \in r$, $t2 \in r$) agree on X i.e. $t1[X] = t2[X]$ then they also agree on Y i.e. $t1[Y] = t2[Y]$

- Note: If $K \subset R$ is a key for R then for any $A \in R$,

  $K \rightarrow A$ holds

FD Example

- **Ex3:**

  Movies (title, year, length, filmType, studio, star)

  Some FDs are as follows:

  title, year $\rightarrow$ length

  title, year $\rightarrow$ filmType

  title, year $\rightarrow$ star

Find FDs ?

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Teacher → Course

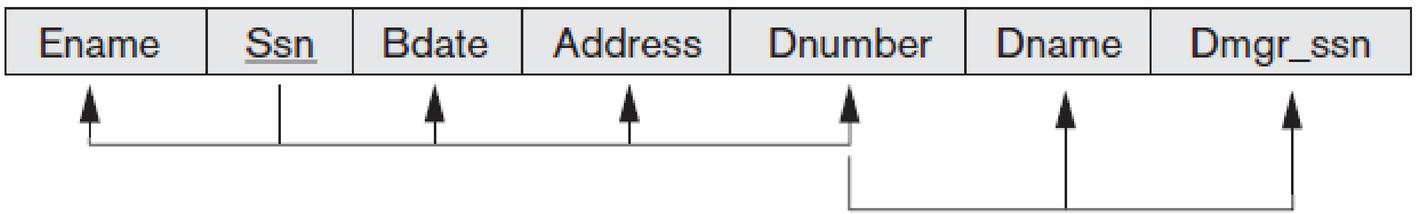
Text → Course

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

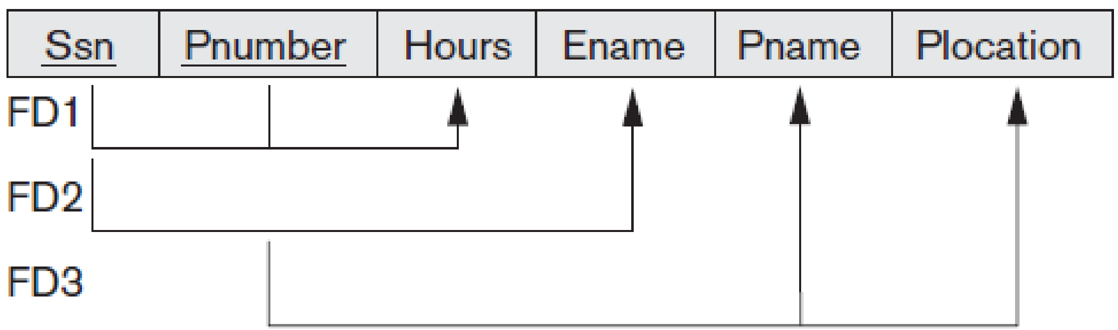
A → B
B → A
B → C
C → B
D → C
AB → D
CD → B

Diagrammatic representation of FDs

EMP_DEPT



EMP_PROJ



Functional Dependencies –Examples

Consider the schema:

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

- Since rollNo is a key,

  rollNo→{studName, gender, dept, hostelName, roomNo}

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

  {hostelName, roomNo}→rollNo

- Suppose boys and girls are accommodated in separate hostels, then

  hostelName→gender

Normalization

- Formal technique for analyzing a relation based on
  - its primary key and
  - the FDs between the attributes of that relation.

- A relation can be normalized to a specific form to prevent the possible occurrence of update anomalies.

Process of Normalization

- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

- Most commonly used normal forms are:
  - **1NF**
  - **2NF**
  - **3NF**
  - **BCNF**
  - 4NF
  - 5NF

First Normal Form

- **Def:** A relation schema R is in 1NF if every attribute of R takes only single and atomic values.

- **Ex:** Student_Apply

RollNo	Name	Age	DeptName
201	Amar	30	{ee, cse}
205	Akbar	32	{ece}
203	Anthony	31	{auto, robo}

- Violates 1NF, because of DeptName.

First Normal Form

- Solution1:
  - Use multiple tuples – one per value.
  - Student_Apply(RollNo, DeptName, Name, Age)

- Solution2:
  - Split the Student table into two different tables.
  - Student(RollNo, Name, Age)
  - Apply(RollNo,DepName)

Fully Functional Dependency

- The FD A→ B in R is a full FD, if removal of any attribute X from A makes the dependency does not hold any more on R.

  Ex: {ProfID,ProjNo} → Hours

- **Partial Functional Dependency:** When an attribute is removed from a relation and if the dependency still holds good, then such a FD is said to be partial.

  Ex: {ProfID,ProjNo} → Name

Second Normal Form

- **Def:** A relation schema R is in 2NF if and only if it is in 1NF and every non-prime attribute in R is fully functionally dependent on key.

Note:

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

2NF - Example

Consider:

- Student (USN, Name, Dept, Gender, HostelName, RoomNo)
- **Assumption:** Boys & girls are accommodated in separate hostels
- **Keys:** USN, {HostelName, RoomNo}
- **Q:** Is this relation in 2NF ?

- **A:** Not in 2NF as HostelName→Gender
- **Converting to 2NF: [Both the relations are in 2NF]**
- Student (USN, Name, Dept, HostelName, RoomNo)
  HostelDetail(HostelName, Gender)

Transitivity FDs

- In any relation R, if A→ B and B → C, then the FD A → C also holds for R.

- **Note:** B is not a subset of any key of R

- **Example:**

  If Employee_Number → Job and

  Job → Salary, then

  Employee_Number → Salary

Transitivity FDs - Example

Consider

- Student (USN, Name, Dept, HostelName, RoomNo, HeadDept)
- **Keys:** USN, {HostelName, RoomNo}
- **FDs:** USN→Dept; Dept →HeadDept hold.

- Thus, USN→HeadDept a transitive dependency.
- **Note:** Head of the dept of Dept is stored redundantly in every tuple where Dept appears.
- **Q:** Is the relation in 2NF ?
- **A:** Relation is in 2NF but redundancy still exists.

Third Normal Form [Example]

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

Eg:

Student (USN, Name, Dept, HostelName, RoomNo, HeadDept)

Is it in 3NF? **[No]**

Decompose:

Student (USN, Name, Dept, HostelName, RoomNo)

DeptInfo(Dept, HeadDept) both in 3NF

Redundancy in data storage is removed.

Trivial FDs

- Consider Movies relation:

  title, year → title

- An FD where the right hand side is contained within the left hand side is called a **trivial FD**.

- If there is at least one element on the RHS that is not contained in the LHS, it is called **non-trivial**, and if none of the elements of the RHS are contained in the LHS, it is called **completely non-trivial FD**.

Trivial / Non-Trivial FDs

- An FD $A \rightarrow B$ where $B \subseteq A$

  -called a trivial FD

- An FD $A \rightarrow B$ where $B \nsubseteq A$

  -non-trivial FD

- An FD $A \rightarrow B$ where $A \cap B = \emptyset$

  -completely non-trivial FD

Closure of FDs

- In any relation R, let A be a set of attributes of R.

- The **closure of FDs** defined by A, is the set of all attributes that are "eventually" defined by A.

- Let:

  $A \rightarrow B;$

  $B \rightarrow C, D;$

  $B \cup D \rightarrow E;$

  Then, **closure(A) = A ∪ B ∪ C ∪ D ∪ E**

Closure of FDs

- Adding attributes to closure(A):

  Let

  $A' \subseteq$ closure(A) and

  $A' \rightarrow F$, then

  closure(A) = closure(A) ∪ F

Computing closure of FDs

- Given a relation R and a set of attributes A, closure(A) is computed by the following algorithm:

  1. Initially *closure(A) = A*

  2. For every $A' \subseteq A$, *if there exists an FD of the form* $A' \rightarrow B$ and $B \not\subseteq A$, *then* $closure(A) = closure(A) \cup B$

  3. Repeat step 2 until no more attributes can be added to closure(A)

- The closure of a set of attributes A is denoted by A^+. Note that if A^+ is the set of all attributes of R, then A is a super-key of R.

Closure of a set of Attributes

Example: F =

$\{name\} \rightarrow \{color\}$
$\{category\} \rightarrow \{department\}$
$\{color, category\} \rightarrow \{price\}$

Example Closures:

$\{name\}^{+} = \{name, color\}$
$\{name, category\}^{+} = \{name, category, color, dept, price\}$
$\{color\}^{+} = \{color\}$

Closure - Example

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\mathbf{\{A,B\}}^+ = \{A, B, C, D, E\}$

Compute $\mathbf{\{A, F\}}^+ = \{A, B, C, D, E, F\}$

Closure and Keys

- How can we find all keys given a set of FDs ?

- Is A a key of R?
- **Sol:** Compute A^+ (if = all attributes) then A is a key.

Specifying FDs for a relation

- **Want:** Minimal set of completely nontrivial FDs such that all FDs that hold on the relation follow from the dependencies in this set.

Inferred FDs

- In a relation R, suppose A, B, C and D be sets of attributes of R such that:

  $$A \rightarrow B;\ B \rightarrow C;\ \text{and } C \rightarrow D$$

  Also let $D_A \subset D$ such that $D_A \subset A$ and let

  $$D' = D - D_A.$$

  Given this, we can infer a *non-trivial FD:*

  $$A \rightarrow D'$$

- FDs which are specified are called ***stated FDs****, and FDs which are* derived are called ***inferred FDs***.

Finding Functional Dependencies

Example: **Inferred FDs:**

Inferred FD	**Rule used**
4. {Name, Category} -> {Name}	?
5. {Name, Category} -> {Color}	?
6. {Name, Category} -> {Category}	?
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Example: **Inferred FDs:**

Inferred FD	**Rule used**
4. {Name, Category} -> {Name}	**Trivial**
5. {Name, Category} -> {Color}	**Transitive (4 -> 1)**
6. {Name, Category} -> {Category}	**Trivial**
7. {Name, Category -> {Color, Category}	Split/**Combine (5 + 6)**
8. {Name, Category} -> {Price}	**Transitive (7 -> 3)**

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Can we find an algorithmic way to do this?

Armstrong's Axioms (Inference Rules)

- For computing the set of FDs that follow a given FD, the following rules called ***Armstrong's axioms*** *are useful:*

1. **Reflexivity:** If $B \subseteq A$, then $A \rightarrow B$

2. **Augmentation:** If $A \rightarrow B$, then $A \cup C \rightarrow B \cup C$
[Note also that if $A \rightarrow B$, then $A \cup C \rightarrow B$ for any set of attributes C.]

3. **Transitivity:** If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

Additional Inference Rules

4. (**Decomposition**) – split rule

If A -> BC, then A -> B and A -> C

5. (**Union**) – combine rule

If A -> B and A -> C, then A -> BC

6. (**Psuedo-transitivity**)

If A -> B and DB -> C, then DA -> C

Projecting FDs

- Let R be a relation and F(R) be the set of all FDs in R.

- Suppose relation S is projected from R, by removing some attributes. How can we infer F(S)?

- FDs that belong to F(S) are those which:

  1. Follow from F(R)
  2. Involve only attributes of S

Projecting FDs

- Given a relation R (A,B,C,D) and F(R) = {A→B, B→C, C→D}.

- Suppose S is projected from R as S(A,C,D). What is F(S).

  To compute F(S), start by computing the closures of all attributes in S.

  In R, A^+ = {A→B, A→C, A→D}

  In S, A^+ = {A→C, A→D}

  C^+ = {C→D} and

  D^+ = {D}

- Since A^+ contains all attributes of S, it is not required to compute $(AC)^+$, $(AD)^+$ or $(ACD)^+$.

Decomposition of relational schema

- Good decomposition – **lossless join property** (reassembling produces original relation data)
- Into good relations – **BCNF**

- **BCNF:**
- Relation R with FDs is in BCNF if,

  For each A → B, A is a key

Decomposition & BCNF

- Anomalies are removed from a relation R(A), by *decomposing* it into other relations S(B) and T(C) where B, C ⊂ A, such that there are no anomalies in S and T.

- A decomposition that does not contain any anomalies is said to be in **Boyce-Codd Normal Form (BCNF)**.

- **Def:** A relation R(A) is said to be in BCNF, if any *nontrivial* FD of the form A’ → A” exists in R(A), it means A’ is a super-key for R.

Decomposition & BCNF

- In a given relation R(A), let there be a functional dependency of the form A’ → A” which violates BCNF.

- In order to bring R into BCNF, decompose R as follows:
- Let B be the set of all attributes which lie in the RHS of any FD that has A’ in the LHS.
- Remove the set of all attributes A’ ∪ B and form a separate relation.
- Retain A’ along with A – {A’ ∪ B} to form the other decomposed part of the relation R.

BCNF decomposition algorithm

Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"

Compute keys for R

Repeat until all relations are in BCNF:

- Pick any R' with $A \rightarrow B$ that violates BCNF
- Decompose R' into $R_1(A, B)$ and $R_2(A, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2

2-attribute Relations BCNF

- Any 2-attribute relation of the form R(A,B) is always in BCNF.
- To **prove,** consider the following cases:

1. There are no FDs between A and B, in which case only trivial FDs exist and R is in BCNF
2. A → B, but there is no FD of the form B → A. In this case, A is the key and R is in BCNF.
3. B → A, but there is no FD of the form A → B. This is symmetric to the case above, here, B is the key.
4. A → B and B → A. Both A and B are keys, this does not violate the BCNF condition.

Third Normal Form (3NF) - Alternate

- Sometimes, some BCNF violating FDs cannot be removed from relations without losing information.
- Consider the relation **Drama (title, city, theater)** having the following FDs:

  FD1: title, city → theater

  FD2: theater → city

  (each drama theater has a unique name across cities)

Candidate Keys are: ?

- {title, city}
- {theater, title}

Third Normal Form (3NF) - Alternate

- FD2 violates BCNF since {theater} is not a key to Drama.

- Based on FD2, if we decompose Drama into the relations
  Drama1 (title, theater) and
  Drama2 (theater, city) it will be incorrect! **Why ?**

- This is because in the join of the relation Drama1 and Drama2, (title, city) will no longer be the key!

Third Normal Form (3NF) - Alternate

- Consider the example tables:

Drama1

Title	Theater
SOP	Ratna mandir
SOP	Gujarat mandir

Drama2

Theater	City
Ratna mandir	Hubli
Gujarat mandir	Hubli

Third Normal Form (3NF) - Alternate

- A Join between Drama1 and Drama2 gives the table:

Title	Theater	City
SOP	Ratna mandir	Hubli
SOP	Gujarat mandir	Hubli

- **Note that above relation violates the FD1**

title, city → theater

Third Normal Form (3NF) - Alternate

- Discrepancies in the previous example occurred because of the FD theater → city where theater is not part of a key, but city is!
- In accommodating such cases, the 3NF decomposition is used which relaxes BCNF as follows:

- **Def:** Any relation R is said to be in 3NF, if for any non-trivial FD of the form A → B, either A is the super-key **or B is a prime attribute**.

> **BCNF Def:** A relation R is said to be in BCNF, if for any nontrivial FD of the form A→ B exists in R, where A is the super-key for R.

3NF - Example

- **gradeInfo(rollNo, studName, course, grade)**
- Suppose the following FDs hold:

1) rollNo, course →grade
2) studName, course →grade
3) rollNo→studName
4) studName→rollNo

Keys:
{rollNo, course}
{studName, course}

- For 1,2 LHS is a key. For 3,4 RHS is prime, So gradeInfo is in 3NF
- But studName is stored redundantly along with every course being done by the student

3NF - Example cont…

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

- Decompose:

  gradeInfo(rollNo, course, grade)

  studInfo(rollNo, studName)

- Redundancy allowed by 3NF is disallowed by BCNF

  BCNF is stricter than 3NF

  3NF is stricter than 2NF

Exercise

Consider relation schema R

(<u>title,year</u>,studioName,president,presAddr)

with three given functional dependencies F:

title, year → studioName

studioName → president

president → presAddr

Is R in BCNF? If yes state why, if not decompose into one or more relations which are in BCNF.

Solution

Consider relation schema R

(<u>title,year</u>,studioName,president,presAddr)

with three given functional dependencies F:

title, year → studioName president presAddr

studioName → president

president → presAddr

1) Pick violating FD: studioName → presAddr

Solution cont…

2) Compute $\{studioName\}^+$
studioName → president, presAddr

Decomposition:
R1 (studioName, president, presAddr)
R2 (studioName, title, year)

Candidate Keys:
R1: {studioName}
R2: {title, year}

Done?
R2 in BCNF but R1 is not.

Solution cont…

Repeat Step 2 for R1:

1, 2) Pick violating FD and compute closure:

president $\rightarrow$ presAddr

Decomposition:

R11 (president, presAddr)

R12 (president, studioName)

Keys:

R11: {president}

R12: {studioName}

Done?

Yes, all three relations in BCNF

Resources

- Chapter 15 (Sec. 15.1 to Sec. 15.5) of Fundamentals of Database Systems (FODS), 6^{th} Edition.

- Internet Surf