Database Design & Normalization

BY: Prof. Prachi Shah

How we design a database

Designing a database is broadly categorized into four steps.

1. Purpose of database is defined.
2. Gather/ analyze data, design schemas and keys.
3. Create relationship among tables and apply constraints.
4. NORMALIZATION

Features of good relation database

- Relational database design requires that we find a "good" collection of relation schemas. A bad design may lead to
  - Repetition of Information
  - Inability to represent certain information.

- Design Goals
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of database integrity constraints.

Normalization

- Main objective in developing a logical data model for relational database systems is to create an accurate representation of the data, its relationships, and constraints.
- To achieve this objective, we must identify a suitable set of relations => The Normalization Technique
- Developed by E.F. Codd (1972).

Normalization

- Normalization: process of efficiently organizing data in the DB.
- Normalization presents a set of rules that tables and databases must follow to be well structured.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Maintaining integrity constraints.
  - Minimal loss of information.
  - Removes the anomalies.

Normalization

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies.
- Goal is to avoid (minimize) anomalies

– Insertion Anomaly: adding new rows forces user to create duplicate data

– Deletion Anomaly: deleting a row may cause loss of other data representing completely different facts

– Modification/Updation Anomaly: changing data in a row forces changes to other rows because of duplication

Database Anomalies

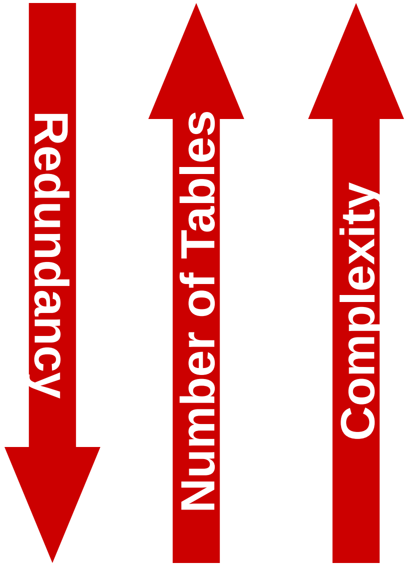
EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/201X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/201X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/201X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/201X
110	Chris Lucero	Info Systems	43,000	C++	4/22/201X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/201X
150	Susan Martin	Marketing	42,000	Java	8/12/201X

- What's the primary key? Answer – Composite: EmpID, CourseTitle
- Insertion – can't enter a new employee without having the employee take a class.
- Deletion – if we remove employee 140, we lose information about the existence of a Tax Acc class
- Modification – giving a salary increase to employee 100 forces us to update multiple records

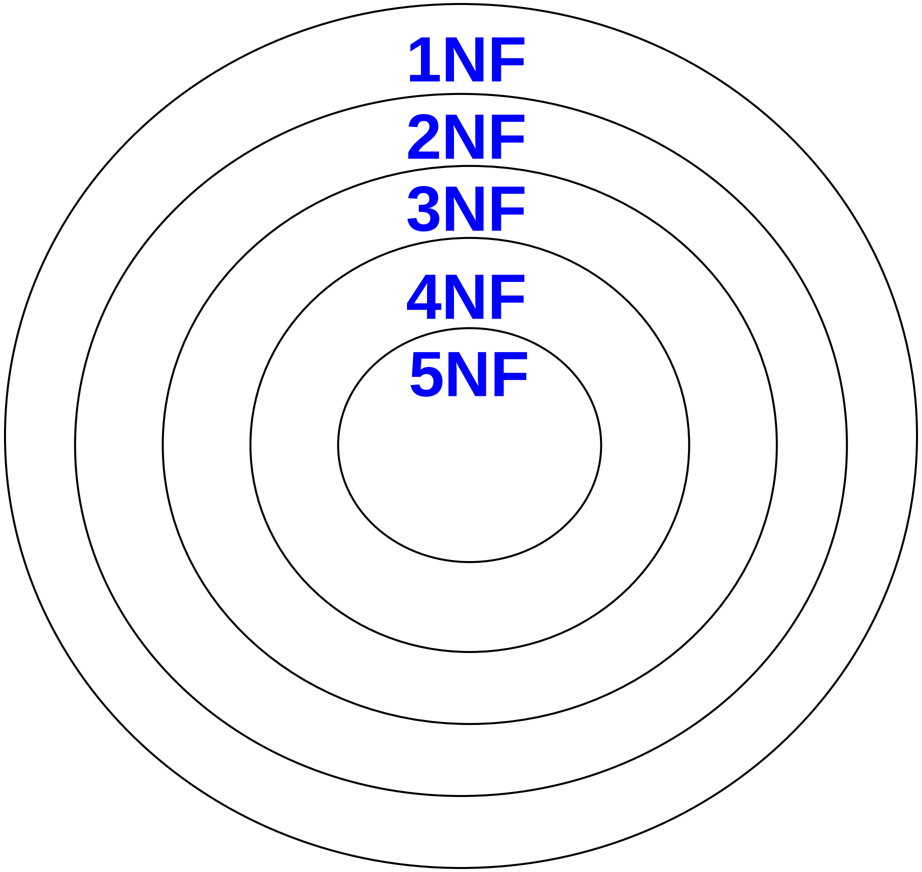
Levels of Normalization

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)



Most databases should be 3NF or BCNF in order to avoid the database anomalies.

Levels of Normalization



Each higher level is a subset of the lower level

First Normal Form (1NF)

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

Example (Not 1NF)

ISBN	Title	AuName	AuPhone	PubName	PubPhone	Price
0-321-32132-1	Balloon	Sleepy, Snoopy, Grumpy	321-321-1111, 232-234-1234, 665-235-6532	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Jones, Smith	123-333-3333, 654-223-3455	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Joyce	666-666-6666	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	Roman	444-444-4444	Big House	123-456-7890	$25.00

Author and AuPhone columns are not scalar

1NF - Decomposition

1. Place all items that appear in the repeating group in a new table
2. Designate a primary key for each new table produced.
3. Duplicate in the new table the primary key of the table from which the repeating group was extracted or vice versa.

Example (1NF)

ISBN	Title	PubName	PubPhone	Price
0-321-32132-1	Balloon	Small House	714-000-0000	$34.00
0-55-123456-9	Main Street	Small House	714-000-0000	$22.95
0-123-45678-0	Ulysses	Alpha Press	999-999-9999	$34.00
1-22-233700-0	Visual Basic	Big House	123-456-7890	$25.00

ISBN	AuName	AuPhone
0-321-32132-1	Sleepy	321-321-1111
0-321-32132-1	Snoopy	232-234-1234
0-321-32132-1	Grumpy	665-235-6532
0-55-123456-9	Jones	123-333-3333
0-55-123456-9	Smith	654-223-3455
0-123-45678-0	Joyce	666-666-6666
1-22-233700-0	Roman	444-444-4444

Functional Dependencies

1. If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

Example 1

ISBN	Title	Price
0-321-32132-1	Balloon	$34.00
0-55-123456-9	Main Street	$22.95
0-123-45678-0	Ulysses	$34.00
1-22-233700-0	Visual Basic	$25.00

Table Scheme: {ISBN, Title, Price}

Functional Dependencies: {ISBN} → {Title}

{ISBN} → {Price}

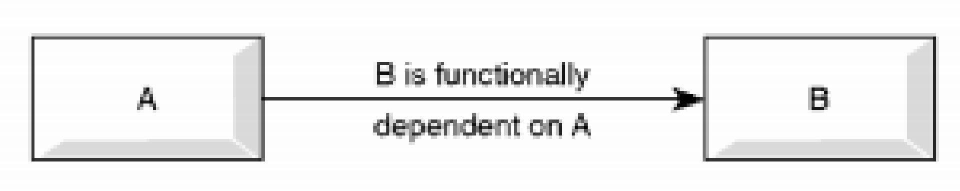
Functional Dependencies

Functional Dependency: The value of one attribute (the determinant) determines the value of another attribute.

A->B reads "Attribute B is functionally dependent on A"

A->B means if two rows have same value of A they necessarily have same value of B

Diagrammatic representation:



Id	Name	Gender	Age
1	Orlando	Male	35
2	John	Male	35
3	Jane	Female	31
4	Jane	Female	30

- **Id ->Name?**
- **Age -> Gender?**
- **Name -> Id?**
- **Name, Age -> Id?**

Prachi Shah

Functional Dependencies

Example 2

PubID	PubName	PubPhone
1	Big House	999-999-9999
2	Small House	123-456-7890
3	Alpha Press	111-111-1111

Table Scheme: {PubID, PubName, PubPhone}

Functional Dependencies: {PubId} → {PubPhone}

{PubId} → {PubName}

{PubName, PubPhone} → {PubID}

Example 3

AuID	AuName	AuPhone
1	Sleepy	321-321-1111
2	Snoopy	232-234-1234
3	Grumpy	665-235-6532
4	Jones	123-333-3333
5	Smith	654-223-3455
6	Joyce	666-666-6666
7	Roman	444-444-4444

Table Scheme: {AuID, AuName, AuPhone}

Functional Dependencies: {AuId} → {AuPhone}

{AuId} → {AuName}

{AuName, AuPhone} → {AuID}

FD – Example

A database to track reviews of papers submitted to an academic conference. Prospective authors submit papers for review and possible acceptance in the published conference proceedings. Details of the entities

- Author information includes a unique author number, a name, a mailing address, and a unique (optional) email address.
- Paper information includes the primary author, the paper number, the title, the abstract, and review status (pending, accepted,rejected)
- Reviewer information includes the reviewer number, the name, the mailing address, and a unique (optional) email address
- A completed review includes the reviewer number, the date, the paper number, comments to the authors, comments to the program chairperson, and ratings (overall, originality, correctness, style, clarity)

FD – Example

Functional Dependencies

- AuthNo → AuthName, AuthEmail, AuthAddress
- AuthEmail → AuthNo, aname
- PaperNo → Primary-AuthNo, Title, Abstract, Status
- RevNo → RevName, RevEmail, RevAddress
- RevEmail → RevNo
- RevNo, PaperNo → AuthComm, Prog-Comm, Date, Rating1, Rating2, Rating3, Rating4, Rating5

FD – Example

A	B	C
1	2	3
4	2	3
5	3	3

Q: Which of the following FD is not correct?

$A \to B$ ✓

$B \to C$ ✓

$BC \to A$ ✗

$AC \to B$ ✓

X	Y	Z
1	4	3
1	5	3
4	6	3
3	2	2

Q: Which of the following FDs are satisfied.

a) $XY \to Z$, $Z \to Y$ ✗

b) $XZ \to X$, $Y \to Z$ ✓

c) $YZ \to X$, $Z \to X$ ✗

d) $XZ \to Y$, $Y \to Z$ ✗

FDs are two types:

1) Trivial FD ($X \to Y$ and $Y \subseteq X$)

2) Non-Trivial FD ($X \to Y$ and $Y \not\subseteq X$)

Armstrong inference rules

- ***Axioms:***
  1. Reflexivity: if $Y \subseteq X$, then $X \rightarrow Y$

     ename, eage, egender -> ename,eage

  2. Augmentation: if $X \rightarrow Y$, then $WX \rightarrow WY$

     rollno -> name

     rollno, age -> name, age

  3. Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

     rollno -> branch and branch -> institute

     so, rollno -> institute

Armstrong inference rules

- ***Derived Rules:***
  1. Union: if $X \rightarrow Y$ and $X \rightarrow Z$, the $X \rightarrow YZ$

     rollno -> name   and   rollno -> age

     rollno -> name, age

  2. Decomposition: if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

  2. Pseudo transitivity: if $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

     rollno -> branch   and   branch, year -> proctor

     rollno, year -> proctor

Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements

RULE 1: The database is in first normal form

RULE 2: No Partial functional dependency is there.

All **nonkey** attributes in the table must be functionally dependent on the entire primary key, not on part of primary key.

Note: *Remember that we are dealing with non-key attributes*

Example 1 (Not 2NF)

Scheme → {Title, PubId, AuId, Price, AuAddress}

1. **Key → {Title, PubId, AuId}**
2. **{Title, PubId, AuID} → {Price}**
3. **{Title, PubId, AuID} → {AuAddress}**
4. **{AuID} → {AuAddress} // PARTIAL**
5. **AuAddress does not belong to a key**
6. **AuAddress functionally depends on AuId which is a subset of a key**

- PRIMARY KEY ROLLNO
- SNAME, EMAILID -- COMPOSITE

Second Normal Form (2NF)

Example 2 (Not 2NF)

Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

1. **key → {City, Street, HouseNumber}**
2. **{City, Street, HouseNumber} → {HouseColor}**
3. **{City} → {CityPopulation} ///new table**
4. **CityPopulation does not belong to any key.**
5. **CityPopulation is functionally dependent on the City which is a proper subset of the key**

Example 3 (Not 2NF)

Scheme → {studio, movie, budget, studio_city}

1. **Key → {studio, movie}**
2. **{studio, movie} → {budget}**
3. **{studio} → {studio_city}**
4. **studio_city is not a part of a key**
5. **studio_city functionally depends on studio which is a proper subset of the key**

2NF - Decomposition

1. If a data item is fully functionally dependent on only a part of the primary key, move that data item and that part of the primary key to a new table.
2. If other data items are functionally dependent on the same part of the key, place them in the new table also
3. Make the partial primary key copied from the original table the primary key for the new table. Place all items that appear in the repeating group in a new table

Example 1 (Convert to 2NF)

Old Scheme → {<u>Title, PubId, AuId</u>, Price, AuAddress}

New Scheme → {<u>Title, PubId, AuId</u>, Price}

New Scheme → {<u>AuId</u>, AuAddress}

2NF - Decomposition

Example 2 (Convert to 2NF)

Old Scheme → {City, Street, HouseNumber, HouseColor, CityPopulation}

New Scheme → {City, Street, HouseNumber, HouseColor}

New Scheme → {City, CityPopulation}

Example 3(Convert to 2NF)

Old Scheme → {Studio, Movie, Budget, StudioCity}

New Scheme → {Movie, Studio, Budget}

New Scheme → {Studio, City}

Third Normal Form (3NF)

This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes.

For a table to be in 3NF, there are two requirements

RULE 1: The table should be second normal form.

RULE 2: No transitive dependency should be there.

Example (Not in 3NF)

Scheme → {Title, PubID, PageCount, Price }

1. **Key → {Title, PubId}**
2. **{Title, PubId} → {PageCount}**
3. **{PageCount} → {Price}**
4. **Both Price and PageCount depend on a key hence 2NF**
5. **Transitively {Title, PubID} → {Price} hence not in 3NF**

Third Normal Form (3NF)

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In the table able, [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Type].

Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functional dependency, and this structure does not satisfy third normal form.

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Third Normal Form (3NF)

Example 2 (Not in 3NF)

Scheme → {<u>Studio</u>, StudioCity, CityTemp}

1. Primary Key → {Studio}
2. {Studio} → {StudioCity}
3. {StudioCity} → {CityTemp}
4. {Studio} → {CityTemp}
5. Both StudioCity and CityTemp depend on the entire key hence 2NF
6. CityTemp transitively depends on Studio hence violates 3NF

Example 3 (Not in 3NF)

Scheme → {BuildingID, Contractor, Fee}

1. Primary Key → {BuildingID}
2. {BuildingID} → {Contractor}
3. {Contractor} → {Fee}
4. {BuildingID} → {Fee}
5. Fee transitively depends on the BuildingID
6. Both Contractor and Fee depend on the entire key hence 2NF

BuildingID	Contractor	Fee
100	Randolph	1200
150	Ingersoll	1100
200	Randolph	1200
250	Pitkin	1100
300	Randolph	1200

3NF - Decomposition

1. Move all items involved in transitive dependencies to a new entity.
2. Identify a primary key for the new entity.
3. Place the primary key for the new entity as a foreign key on the original entity.

Example 1 (Convert to 3NF)

Old Scheme → {Title, PubID, PageCount, Price }

New Scheme → {Title, PubID, PageCount}

New Scheme → {PubID, PageCount, Price}

3NF - Decomposition

Example 2 (Convert to 3NF)

Old Scheme → {<u>Studio</u>, StudioCity, CityTemp}

New Scheme → {<u>Studio</u>, StudioCity}

New Scheme → {<u>StudioCity</u>, CityTemp}

Example 3 (Convert to 3NF)

Old Scheme → {BuildingID, Contractor, Fee}

New Scheme → {BuildingID, Contractor}

New Scheme → {Contractor, Fee}

BuildingID	Contractor
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

Contractor	Fee
Randolph	1200
Ingersoll	1100
Pitkin	1100

Boyce-Codd Normal Form (BCNF)

For a table to be in BCNF (3.5NF) , there are two requirements

RULE 1: The table should be in the Third Normal Form.

RULE 2: And, for any dependency A → B, A should be a super key.

- In simple words, it means, that for a dependency A → B, A cannot be a non-prime attribute, if B is a prime attribute.
- Any non-prime attribute should never define any prime attribute.

OR

- A relational schema R is considered to be in Boyce–Codd normal form (BCNF) if, for every one of its dependencies X → Y, one of the following conditions holds true:

- X → Y is a trivial functional dependency (i.e., Y is a subset of X)
- X is a superkey for schema R

Boyce-Codd Normal Form (BCNF)

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- In the table above:
  - One student can enroll for multiple subjects. For example, student with student_id 101, has opted for subjects - Java & C++
  - For each subject, a professor is assigned to the student.
  - And, there can be multiple professors teaching one subject like we have for Java.
- What do you think should be the Primary Key?
  - Well, in the table above **student_id, subject** together form the primary key, because using student_id and subject, we can find all the records of the table.

Boyce-Codd Normal Form (BCNF)

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- One more important point to note here is, one professor teaches only one subject, but one subject may have two different professors.
- Hence, there is a dependency between subject and professor here, where subject depends on the professor name.
- **Why this table is not in BCNF?**
- In the table above, **student_id, subject** form primary key, which means **subject** column is a **prime attribute**.
- But, there is one more dependency, professor → subject.
- And while subject is a prime attribute, professor is a **non-prime attribute**, which is not allowed by BCNF.

BCNF - Decomposition

1. Place the two candidate primary keys in separate entities
2. Place each of the remaining data items in one of the resulting entities according to its dependency on the primary key.

Student Table

student_id	p_id
101	1
101	2
and so on...	

And, **Professor Table**

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++
and so on...		

EXAMPLE

pk(Patno,time)

Patient No	Patient Name	Appointment Id	Time	Doctor
1	John	0	09:00	Zorro
2	Kerr	0	09:00	Killer
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

DB(Patno,PatName,appNo,time,doctor)

Patno -> PatName
Patno,appNo -> Time,doctor
Time -> appNo

Now we have to decide what the primary key of DB is going to be. From the information we have, we could chose:

DB(Patno,PatName,appNo,time,doctor) pk(Patno,appNo)

Example - DB(<u>Patno,</u>PatName,<u>appNo</u>,time,doctor)

1NF Eliminate repeating groups. (None)

DB(<u>Patno,</u>PatName<u>,appNo</u>,time,doctor)

2NF Eliminate partial key dependencies

R1(<u>Patno,appNo</u>,time,doctor) **R2(<u>Patno</u>,PatName)**

Patno,appNo -> Time,doctor **Patno -> PatName**

Time -> appNo

R1(<u>Patno,appNo</u>,time,doctor) R2(<u>Patno</u>,PatName)

3NF Eliminate transitive dependencies

None: so just as 2NF

BCNF: Every determinant is a super key

R1(<u>Patno,appNo</u>,time,doctor)   R2(<u>Patno</u>,PatName)

Go through all determinates where ALL of the left hand attributes are present in a relation and at least ONE of the right hand attributes are also present in the relation.

Patno -> PatName
Patno is present in DB, but not PatName, so not relevant.

Patno,appNo -> Time,doctor
All LHS present, and time and doctor also present, so relevant.

Is determinant a super key? Patno,appNo IS the key, so this is a super key. Thus this is OK for BCNF compliance.

Time -> appNo

Time is present, and so is appNo, so relevant.

Is determinant a super key? If it was then we could rewrite DB as:

R1(Patno,appNo,<u>time</u>,doctor)

This will not work, as you need both time and Patno together to form a unique key.

Thus this determinant is not a candidate key, and therefore DB is not in BCNF. We need to fix this.

BCNF: rewrite to

R1(<u>Patno,time</u>,doctor)

R2(<u>Patno</u>,PatName)

R3(<u>time</u>,appNo)

Database Design & Normalization

BY: Prof. Prachi Shah

Closure of a set of Attributes

- **<u>Closure of an Attribute Set-</u>**

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.
- Closure of attribute set $\{X\}$ is denoted as $\{X\}^+$.

Closure of a set of Attributes

- DB (Name, Color, Category, Department, Price)
- Example: FD
- {name} -> {color}
- {category} -> {department}
- {color, category} -> {price}

- ***Example Closures:***
- {name}+ = {name, color}
- {name, category}+ = {name, category, color, dept, price}
- {color}+ = {color}

Closure of a set of Attributes

EXAMPLE 1: Consider a relation R (A , B , C , D , E , F , G) with the

FD= { $A \rightarrow BC$,

$BC \rightarrow DE$,

$D \rightarrow F$,

$CF \rightarrow G$

}

Closure of attribute A-

- $A^+ = \{ A \}$
- $= \{ A , B , C \}$ (Using $A \rightarrow BC$)
- $= \{ A , B , C , D , E \}$ (Using $BC \rightarrow DE$)
- $= \{ A , B , C , D , E , F \}$ (Using $D \rightarrow F$)
- $= \{ A , B , C , D , E , F , G \}$ (Using $CF \rightarrow G$)
- Thus,
- **$A^+ = \{ A , B , C , D , E , F , G \}$ A->B A->F**

Closure of a set of Attributes

- **Closure of attribute D-**

- $D^+ = \{ D, F\}$

- **Closure of attribute set {B, C}-**

- $\{ BC \}^+ = \{B, C, D, E, F, G\}$

Closure of a set of Attributes

- **EXAMPLE**

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

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

- Compute {A, B}+ = {A, B, }
- Compute {A, F}+ = {A, F, B, C, D, E }

EXAMPLE

- Consider the given functional dependencies-

$$F=\{\quad AB \rightarrow CD,\quad AF \rightarrow D,\quad DE \rightarrow F,\quad C \rightarrow G,\quad F \rightarrow E,\quad G \rightarrow A \quad\}$$

Which of the following options is/are false?

(A) $\{ CF \}^+ = \{ A , C , D , E , F , G \}$

(B) $\{ BG \}^+ = \{ A , B , C , D , G \}$

(C) $\{ AF \}^+ = \{ A , C , D , E , F , G \}$

(D) $\{ AB \}^+ = \{ A , C , D , F , G \}$

Why Do We Need the Closure?

- **Testing for keys**
  - To test if ‘A’ is a key, we compute A+, and check if A+ contains all attributes of R.
- **Testing functional dependencies**
  - To check if a functional dependency a -> b holds (or, in other words, is in F+), just check if b belongs a+.
  - That is, we compute a+ by using attribute closure, and then check if it contains b.
  - Is a simple and cheap test, and very useful
- **Computing closure of F**
  - For getting additional functional dependencies possible in a relation.

Finding keys in relation

- **Finding the Keys Using Closure- Super Key-**

- If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation.
- Thus, we can say-
- **“The closure of a super key is the entire relation schema.”**

- **Example-**
- In the **EXAMPLE 1** the closure of attribute A is the entire relation schema.
- Thus, attribute A is a super key for that relation.

Finding keys in relation

- **R (ROLLNO, NAME, MAIL, AGE, BRANCH)**

- **ROLLNO+ = {ROLLNO, NAME, MAIL}**

Finding keys in relation

- **Candidate Key-**
- If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation.

- **Example-**
- In **EXAMPLE 1**
- No proper subset of attribute A contains all the attributes of the relation.
- Thus, attribute A is also a candidate key for that relation.

Finding keys in relation

* ex: 2: $R(ABCDE)$

$\{AB \rightarrow C, CD \rightarrow E, DE \rightarrow B\}$

is 'AB' a candidate key? if not is 'ABD'?

- $(AB)^+ = \{ABC\}$ ✗ - NO.
- $(ABD)^+ = \{ABCDE\}$ ✓ - yes.

Finding additional fd’s in relation

2) Finding additional FDS:

R(ABCD) F.D. = {A → BC, B → CD, D → AB}

Find weather AD → C is possible or not?

(AD)+ = {ADBC}

C is there, so yes AD → C is possible

AD -> C IS POSSIBLE?

(AD)+ = {A,D,B,C}   AD->C AD-> B

Finding additional fd's in relation

ex2: R(ABCD)

F.D.S: $\{A \to B,\ A \to C,\ CD \to E,\ B \to D,\ E \to A\}$

which of the following FD is not implied by above set?

a) $CD \to AC$ b) $BD \to CD$ c) $BC \to CD$ d) $AC \to BC$

$(CD)^+ = \{EABCD\}$ ✓ $(BD)^+ = \{BD\}$ ✗ $(BC)^+ = \{BCDEA\}$ ✓

$(AC)^+ = \{ACBD\}$ ✓

Finding candidate key & super keys in relation

Step 1: Identify attributes that are not dependent on any other attribute. (not present on right hand side)

Step 2: Combine those attributes and find closure of those attributes.

Step 3: If closure has all attributes of relation, then it is the ONLY candidate key. If not, try all possible combinations.

Finding candidate key & super keys in relation

⇒ Find key / super key in this relation
R(ABCD) : F.D.S : { A → B , B → D }

R(ABCD)

FD = { A->B,

B->D}

STEP 1: Identify attributes that are not dependent on any other attribute. (not present on right hand side)

A, C

Finding candidate key & super keys in relation

⇒ Find key / super key in this relation
R(ABCD) : F.D.S : { A → B , B → D }

R(ABCD)

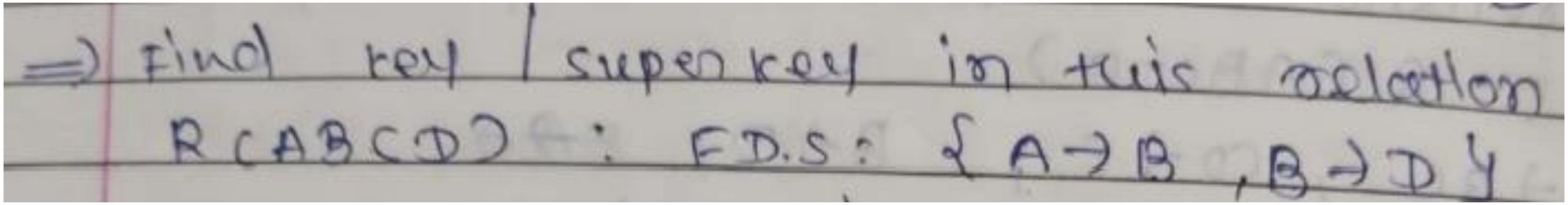
FD = { A->B,

B->D}

Step 2: Combine those attributes and find closure of those attributes

(AC)+ = {A, C, B, D}

Finding candidate key & super keys in relation


⇒ Find key / super key in this relation
R(ABCD) : F.D.S : { A→B , B→D }

R(ABCD)

FD = { A->B,

B->D}

STEP 3: If closure has all attributes of relation, then it is a candidate key. If not, try all possible combinations.

(AC)+ = {A, C, B, D}, YES, CANDIDATE KEY.

EXAMPLE 1

Let R(A, B, C, D, E, F) be a relation scheme with the following functional dependencies-

F = {A → B,

C → D,

D → E }

STEP 1:

Here, the attributes which are not present on RHS of any functional dependency are A, C and F.

So, essential attributes are- A, C and F.

EXAMPLE 1

Let R(A, B, C, D, E, F) be a relation scheme with the following functional dependencies-

F={A $\rightarrow$ B,

C $\rightarrow$ D,

D $\rightarrow$ E }

STEP 2:

(ACF)+ = {A,C,F,B,D,E}

STEP 3:

YES, CANDIDATE KEY.

EXAMPLE 2

Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies-

FD={$C \rightarrow F$,

$E \rightarrow A$,

$EC \rightarrow D$,

$A \rightarrow B$ }

Which of the following is a candidate key for R?

1. CD
2. EC
3. AE
4. AC

Also, determine the total number of candidate keys , super keys.

EXAMPLE 2

STEP 1: Essential attributes of the relation are- C and E.

STEP 2: We find the closure of CE. So, we have-

{ CE }+

= { C , E }

= { C , E , F } (Using $C \rightarrow F$)

= { A , C , E , F } (Using $E \rightarrow A$)

= { A , C , D , E , F } (Using $EC \rightarrow D$)

= { A , B , C , D , E , F } (Using $A \rightarrow B$)

STEP 3: We conclude that CE can determine all the attributes of the given relation.

So, CE is the only possible candidate key of the relation.

EXAMPLE 2

<u>Total Number of Candidate Keys-</u>

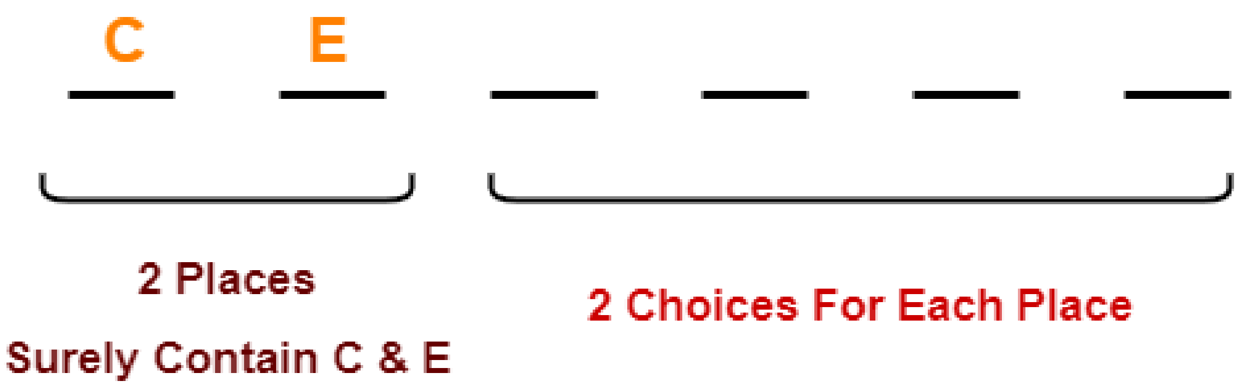
Only one candidate key CE is possible.

<u>Total Number of Super Keys-</u>

- There are total 6 attributes in the given relation of which-
- There are 2 essential attributes- C and E.
- Remaining 4 attributes are non-essential attributes.
- Essential attributes will be definitely present in every key.
- Non-essential attributes may or may not be taken in every super key.

EXAMPLE 2

<u>Total Number of Super Keys-</u>



So, number of super keys possible = 2 x 2 x 2 x 2 = 16.

Thus, total number of super keys possible = 16.

EXPLANATION

Let a Relation R have attributes {a1,a2,a3} & a1 is the candidate key. Then how many super keys are possible?

Here, any superset of a1 is the super key.

Super keys are = {a1,

a1 a2,

a1 a3,

a1 a2 a3}

a1,_,_

Thus we see that 4 Super keys are possible in this case.

EXAMPLE 3

Let R = (A, B, C, D, E) be a relation scheme with the following dependencies-

$AB \rightarrow C$

$C \rightarrow D$

$B \rightarrow E$

Determine the total number of candidate keys.

STEP 1: Essential attributes of the relation are- A and B.

STEP 2:

{ AB }+ = { A , B , C , D , E }

A ,B, ____, _____, ____ 2^3 = 8

AB is the only possible candidate key of the relation.

EXAMPLE 4

Consider the relation scheme R(E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies-

{ E, F } $\rightarrow$ { G }

{ F } $\rightarrow$ { I , J }

{ E, H } $\rightarrow$ { K, L }

{ K } $\rightarrow$ { M }

{ L } $\rightarrow$ { N }

What is the candidate key for R?

1. { E, F }
2. { E, F, H }
3. { E, F, H, K, L }
4. { E }

EXAMPLE 5

Consider the relation scheme R(A, B, C, D, E, H) and the set of functional dependencies-

$A \rightarrow B$

$BC \rightarrow D$

$E \rightarrow C$

$D \rightarrow A$

What are the candidate keys of R?

1. AE, BE
2. AE, BE, DE
3. AEH, BEH, BCH
4. AEH, BEH, DEH

EXAMPLE 5

Essential attributes of the relation are- E and H.

So, attributes E and H will definitely be a part of every candidate key.

The only possible option is (D).

EXAMPLE 6

R = ABCD, F={A -> BCD, C -> A}

Find all possible candidate keys.

Every attribute is present on R.H.S.

Start with single attributes.

A+ = ABCD

B+ = B

C+ = ABCD

D+ = D

COMBINE ALL REMAINING ATTRIBUTES AND CHECK

BD+ = BD, NOT A CANDIDATE KEY.

EXAMPLES HOMEWORK

$R(ABCD)$ F.D. = $\{AB \rightarrow CD,\ C \rightarrow D,\ B \rightarrow A,\ A \rightarrow B\}$

$R(ABCDE)$ F.D. = $\{AB \rightarrow C,\ C \rightarrow D\}$

$R(ABCDE)$ F.D. = $\{AB \rightarrow C,\ BC \rightarrow A,\ CA \rightarrow B,\ B \rightarrow D,\ D \rightarrow E\}$

FINDING NORMAL FORM OF RELATION

Steps to find the highest normal form of a relation:

1. Find all possible candidate keys of the relation.
2. Divide all attributes into two categories: prime attributes and non-prime attributes.
3. Check for conditions for 1nf, 2nf and so on.

FINDING NORMAL FORM OF RELATION

Trivial

X->Y

Y subset of X

Non-trivial

Y subset not X

FINDING NORMAL FORM OF RELATION

1NF: EVERY RELATION IS BY DEFAULT IN 1 NF.

2NF: NO PARTIAL DEPENDENCY SHOULD BE THERE.

3NF: NO TRANSITIVE DEPENDENCY SHOULD BE THERE.

BCNF: L.H.S SHOULD BE A SUPER KEY (for non-trivial fd).

To be in 3NF : $X \rightarrow a$ (for every non-trivial dependency)

1) X is either super key or
2) a is prime attributes.

NORMAL FORMS

ii) 2NF: using F.D. & closure.

ex 1. R(ABCD) : F.D. = { A→B, C→D }

is this in 2NF or not?

WHAT ARE PRIME ATTRIBUTES?

A->B   C->D

FIND CANDIDATE KEYS:

(AC)+ = {A,B,C,D}

PRIME ATTRIBUTES: A,C

NON- PRIME ATTRIBUTES: B,D

Check for partial dependency: A->B & C->D

NOT IN 2NF.

EXAMPLE 1: NORMAL FORMS

Find the highest normal form of a relation

R(A,B,C,D,E) with FD set {A->D, B->A, BC->D, AC->BE}

Step 1. C+ = {C}

Combine C with every remaining attribute (a,b,d,e)

AC+, BC+, DC+, EC+

So there will be two candidate keys {AC, BC}.

Step 2. Prime attribute {A,B,C}

non-prime {D,E}.

EXAMPLE 1: NORMAL FORMS

Step 3. The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attribute.

The relation is not in 2nd Normal form because A->D is partial dependency (A which is subset of candidate key AC is determining non-prime attribute D) and 2nd normal form does not allow partial dependency.

So the highest normal form will be 1st Normal Form.

EXAMPLE 2: NORMAL FORMS

Example 2. Find the highest normal form of a relation R(A,B,C,D,E) with FD set as {BC->D, AC->BE, B->E}

Step 1. As we can see, (AC)+ ={A,C,B,E,D} but none of its subset can determine all attribute of relation, So AC will be candidate key.

AC B, ACE, ACD

Step 2. Prime attribute : {A,C}

non-prime {B,D,E}

Step 3. The relation R is in 1st normal form as a relational DBMS does not allow multi-valued or composite attribute.

EXAMPLE 2: NORMAL FORMS

2NF CHECKING:

BC->D is in 2nd normal form (BC is not proper subset of candidate key AC) and

AC->BE is in 2nd normal form (AC is candidate key) and

B->E is in 2nd normal form (B is not a proper subset of candidate key AC).

3NF CHECKING:

The relation is not in 3rd normal form because in BC->D (neither BC is a super key nor D is a prime attribute) and

B->E (neither B is a super key nor E is a prime attribute) but to satisfy 3rd normal for, either LHS of an FD should be super key or RHS should be prime attribute.

So the highest normal form of relation will be 2^{nd} Normal form.

EXAMPLE 3: NORMAL FORMS

ex:5 R (id, projid, name, hrs, projname)

FD = { id, projid → name, hrs, projname

id → name

projid → projname }

EXAMPLE 3: NORMAL FORMS

e.x. ⇒ $(eid, projid)^{+} : \{id, projid, name, hrs, projname\}$

partial dependency: $id \rightarrow name$

$projid \rightarrow projname$

$T_1(\underline{id}, name)$, $T_2(\underline{projid}, projname)$

$T_3(\underline{id, projid}, hrs)$

EXAMPLE 4: NORMAL FORMS

Example. Find the highest normal form of a relation R(A,B,C,D,E) with FD set {B->A, A->C, BC->D, AC->BE}

X->Y

X=SUPERKEY

Step 1. Find all candidate keys.

A+= {A,C,B,E,D}

B+={B,A,C,D,E}

C+={C}

D+={D}

E+={E}

CD, DE, EC

EXAMPLE 4: NORMAL FORMS

Step 2. Prime attribute: {A,B}

non-prime {C,D,E}

Step 3.

The relation is in 2nd normal form because B->A is in 2nd normal form (B is a super key) and A->C is in 2nd normal form (A is super key) and BC->D is in 2nd normal form (BC is a super key) and AC->BE is in 2nd normal form (AC is a super key).

The relation is in 3rd normal form because LHS of all FD's are super keys. The relation is in BCNF as all LHS of all FD's are super keys. So the highest normal form is BCNF.

EXAMPLE 5: NORMAL FORMS

v) given relations are in which form? 1, 2, 3 ?

→) R(ABCDE) F.D. : {A → B, AB → C, C → D, D → E}

EXAMPLE 5: NORMAL FORMS

vi) given relations are in which form? {1, 2, 3}

→) R(ABCDE) F.D. : {A → B, AB → C, C → D, D → E}

(A)+ : {ABCDE} / c.k. ✓

(B)+ : {B} ✗ , (C)+ : {C, D, E} ✗

(D)+ : {D, E} ✗ , (E)+ : {E} ✗

prime : {A} , non-prime {B, C, D, E}

A → B ✓ 3NF
S.K.

AB → C ✓ 3NF
S.K.

C → D ~~not~~ ✗

D → E ✗

T1 (C, D)

T2 (D, E)

T3 (A, B, C)

Not in 3NF. (BCZ C → D, D → E)

→ ~~…~~ partial dependency : Not any

EXAMPLES: NORMAL FORMS

(*) Find NF: GATE Que:

(1) R(ABCDEF) FD = { A → BCDEF
BC → ADEF
DEF → ABC }

BCNF.

(2) name, courseno → grade
rollno, courseno → grade
3NF
name → rollno
roll → name

(3) Emp (empcode, name, street, city, state, pincode)

2NF
FD = { pincode → city, state,
street, city, state → pincode

EXAMPLES: NORMAL FORMS

* BCNF: For every non-trivial dependency $X \to a$, X is Super key.

ex: $R(A,B,C,D)$ F.D. $= \{AB \to C, C \to D\}$

$AB \to C$ $\quad (AB)^+ = \{ABCD\}$ ✓

$\alpha \to a$

$C \to D$ $\quad (C)^+ = \{CD\}$ ✗

Not in BCNF.

i) can be in 3NF?

↳ Find candidate key: $(AB)^+ = \{ABCD\}$ C.K. only.

$C \to D$ ✗ Not in 3NF.

ii) can be in 2NF?

↳ Yes, No partial dependency.

Equivalence of Two Sets of Functional Dependencies-

Two different sets of functional dependencies for a given relation may or may not be equivalent.

If F and G are the two sets of functional dependencies, then following 3 cases are possible-

Case-01: F covers G ($F \supseteq G$)

Case-02: G covers F ($G \supseteq F$)

Case-03: Both F and G cover each other ($F = G$)

Equivalence of Two Sets of Functional Dependencies-

A relation R(A,B,C,D) having two FD sets

FD1 = {A->B, B->C, AB->D} and FD2 = {A->B, B->C, A->C, A->D}

Step 1. Checking whether all FDs of FD1 are present in FD2

- A->B in set FD1 is present in set FD2.
- B->C in set FD1 is also present in set FD2.
- AB->D in present in set FD1 but not directly in FD2 but we will check whether we can derive it or not. For set FD2, (AB)+ = {A,B,C,D}. It means that AB can functionally determine A, B, C and D. So AB->D will also hold in set FD2.

As all FDs in set FD1 also hold in set FD2, FD2 ⊃ FD1 is true.

Equivalence of Two Sets of Functional Dependencies-

Step 2. Checking whether all FDs of FD2 are present in FD1

- A->B in set FD2 is present in set FD1.
- B->C in set FD2 is also present in set FD1.
- A->C is present in FD2 but not directly in FD1 but we will check whether we can derive it or not. For set FD1, (A)+ = {A,B,C,D}. It means that A can functionally determine A, B, C and D. SO A->C will also hold in set FD1.
- A->D is present in FD2 but not directly in FD1 but we will check whether we can derive it or not. For set FD1, (A)+ = {A,B,C,D}. It means that A can functionally determine A, B, C and D. SO A->D will also hold in set FD1.

As all FDs in set FD2 also hold in set FD1, FD1 ⊃ FD2 is true.

Equivalence of Two Sets of Functional Dependencies-

Step 3. As FD2 ⊃ FD1 and FD1 ⊃ FD2 both are true FD2 =FD1 is true. These two FD sets are semantically equivalent.

Equivalence of Two Sets of Functional Dependencies-

Q2: A relation R2(A,B,C,D) having two FD sets

FD1 = {A->B, B->C,A->C} and FD2 = {A->B, B->C, A->D}

FINDING MINIMAL COVER / CANONICAL COVER / FD CLOSURE

Steps to find canonical cover:

1. Deconstruct FD from R.H.S. [Write the given set of functional dependencies in such a way that each functional dependency contains exactly one attribute on its right side.]
2. Remove redundant/non-essential attributes from L.H.S.
3. Removal of redundant FD's (if any).

EXAMPLE 1: FINDING MINIMAL COVER

Find canonical cover:

The following functional dependencies hold true for the relational scheme R (W , X , Y , Z) –

$X \rightarrow W$

$WZ \rightarrow XY$

$Y \rightarrow WXZ$

<u>Step-01:</u> Write all the functional dependencies such that each contains exactly one attribute on its right side-

$X \rightarrow W$

$WZ \rightarrow X$

$WZ \rightarrow Y$

$Y \rightarrow W$

$Y \rightarrow X$

$Y \rightarrow Z$

EXAMPLE 1: FINDING MINIMAL COVER

STEP 2: Remove redundant attributes form L.H.S. In this step ignore those FD's where L.H.S has single attributes.

$WZ \rightarrow X$: From LHS can we remove W? (can remove if we can get W with the help of Z)

'w' $\Rightarrow (z)^+ = \{Z\}$ - NO

'z' $\Rightarrow (w)^+ = \{W\}$. NO.

$WZ \rightarrow X$

$WZ \rightarrow Y$

'w' $\Rightarrow$ NO , 'z' $\rightarrow$ NO.

$WZ \rightarrow Y$

EXAMPLE 1: FINDING MINIMAL COVER

STEP 3: Removal of redundant FD's (if any).

We are getting following from step 2:

1. $X \rightarrow W$
2. $WZ \rightarrow X$
3. $WZ \rightarrow Y$
4. $Y \rightarrow W$
5. $Y \rightarrow X$
6. $Y \rightarrow Z$

EXAMPLE 1: FINDING MINIMAL COVER

STEP 3: Removal of redundant FD's (if any).

1. $X \rightarrow W$ (*) can remove?

   $(X)^+ = (X .. \}$ NO.

2. $WZ \rightarrow X$ (*)

   $(WZ)^+ = \{WZYX\}$ yes.

3. $WZ \rightarrow Y$ (*)

   $(WZ)^+ = \{WZX\}$ NO.

4. $Y \rightarrow W$ (*)

   $(Y)^+ = (YXZW)$ yes.

5. $Y \rightarrow X$ (*)

   $(Y)^+ = (YZW)$ NO

6. $Y \rightarrow Z$ (*)

   $(Y)^+ = \{YXW .. \}$ NO.

EXAMPLE 1: FINDING MINIMAL COVER

Finally, the canonical cover is-

$X \rightarrow W$

$WZ \rightarrow Y$

$Y \rightarrow X$

$Y \rightarrow Z$

EXAMPLE 2: FINDING MINIMAL COVER

Find canonical cover: Consider the following set F of functional dependencies:

F= {

A -> BC

B -> C

A -> B

AB -> C

}

EXAMPLE 2: FINDING MINIMAL COVER

Step-01: Write all the functional dependencies such that each contains exactly one attribute on its right side-

A -> B

A -> C

B -> C

A -> B

AB -> C

STEP 2: Remove redundant attributes form L.H.S. In this step ignore those FD's where L.H.S has single attributes.

AB -> C

Can we remove A? NO (Find B+, if A is in there, can remove)

Can we remove B? YES

A->C

EXAMPLE 2: FINDING MINIMAL COVER

STEP 2: Remove redundant attributes form L.H.S. In this step ignore those FD’s where L.H.S has single attributes

Final FD’s after step 2 are:

A -> B

A -> C

B -> C

A -> B

A -> C

STEP 3: Removal of redundant FD’s

A -> B

B -> C

EXAMPLE 3: FINDING MINIMAL COVER

Find canonical cover: Consider a relation R(A,B,C,D) having some attributes and below are mentioned functional dependencies.

FD = {

B -> A,

AD -> C,

C -> ABD

}

EXAMPLE 3: FINDING MINIMAL COVER

Step 1:

B -> A

AD -> C

C -> A

C -> B

C -> D

Step 2:

AD -> C

Can we remove A? NO, Check if D+ contains A.

Can we remove D? NO, Check if A+ contains D.

EXAMPLE 3: FINDING MINIMAL COVER

Step 3: Remove redundant fd's:

Below are the fd's after step 2:

B -> A

AD -> C

C -> A

C -> B

C -> D

ANS:

B -> A

AD -> C

C -> B

C -> D OR can merge C - > BD

EXAMPLE 4: FINDING MINIMAL COVER

$R(ABCD)$ FD : $\{A \to B,\ C \to B,\ D \to ABC,\ AC \to D$

EXAMPLE 5: FINDING MINIMAL COVER

which of the following is canonical cover for given fd set?

$F : \{A \to B,\ B \to C,\ C \to A,\ B \to A,\ C \to B,\ A \to C\}$

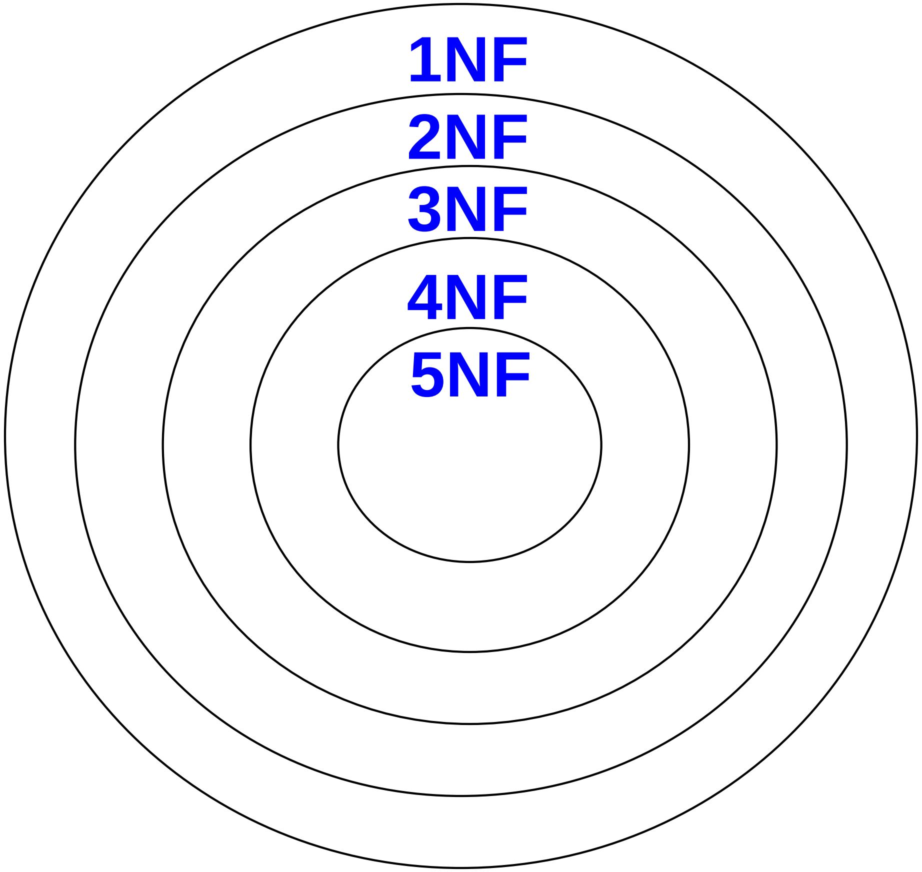
$C1 : \{A \to B,\ B \to C,\ C \to A\}$ ✓

$C2 : \{C \to B,\ B \to A,\ A \to C\}$ ✓

Database Design & Normalization

BY: Prof. Prachi Shah

Levels of Normalization



Each higher level is a subset of the lower level

First Normal Form (1NF)

A table is considered to be in 1NF if all the fields contain only scalar values (as opposed to list of values).

Second Normal Form (2NF)

For a table to be in 2NF, there are two requirements

RULE 1: The database is in first normal form

RULE 2: No Partial functional dependency is there.

All **nonkey** attributes in the table must be functionally dependent on the entire primary key, not on part of primary key.

Note: *Remember that we are dealing with non-key attributes*

Third Normal Form (3NF)

This form dictates that all **non-key** attributes of a table must be functionally dependent on a candidate key i.e. there can be no interdependencies among non-key attributes.

For a table to be in 3NF, there are two requirements

RULE 1: The table should be second normal form.

RULE 2: No transitive dependency should be there.

Boyce-Codd Normal Form (BCNF)

For a table to be in BCNF (3.5NF) , there are two requirements

RULE 1: The table should be in the Third Normal Form.

RULE 2: And, for any dependency $A \rightarrow B$, A should be a super key.

- In simple words, it means, that for a dependency $A \rightarrow B$, A cannot be a non-prime attribute, if B is a prime attribute.
- Any non-prime attribute should never define any prime attribute.

OR

- A relational schema R is considered to be in Boyce–Codd normal form (BCNF) if, for every one of its dependencies $X \rightarrow Y$, one of the following conditions holds true:

- $X \rightarrow Y$ is a trivial functional dependency (i.e., Y is a subset of X)
- X is a superkey for schema R

EXAMPLE

pk(<u>Patno,time)</u>

Patient No	Patient Name	Appointment Id	Time	Doctor
1	John	0	09:00	Zorro
2	Kerr	0	09:00	Killer
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

DB(Patno,PatName,appNo,time,doctor)

Patno -> PatName
Patno,appNo -> Time,doctor
Time -> appNo

Now we have to decide what the primary key of DB is going to be. From the information we have, we could chose:

DB(<u>Patno,</u>PatName<u>,appNo</u>,time,doctor)   pk(<u>Patno,appNo)</u>

Example - DB(Patno,PatName,appNo,time,doctor)

1NF Eliminate repeating groups. (None)

DB(Patno,PatName,appNo,time,doctor)

2NF Eliminate partial key dependencies

R1(Patno,appNo,time,doctor) **R2(Patno,PatName)**

Patno,appNo -> Time,doctor **Patno -> PatName**

Time -> appNo

R1(Patno,appNo,time,doctor) R2(Patno,PatName)

3NF Eliminate transitive dependencies

None: so just as 2NF

BCNF: Every determinant is a super key

R1(<u>Patno,appNo</u>,time,doctor) R2(<u>Patno</u>,PatName)

Go through all determinates where ALL of the left hand attributes are present in a relation and at least ONE of the right hand attributes are also present in the relation.

Patno -> PatName

Patno is present in DB, but not PatName, so not relevant.

Patno,appNo -> Time,doctor

All LHS present, and time and doctor also present, so relevant.

Is determinant a super key? Patno,appNo IS the key, so this is a super key. Thus this is OK for BCNF compliance.

Time -> appNo

Time is present, and so is appNo, so relevant.

Is determinant a super key? If it was then we could rewrite DB as:

R1(Patno,appNo,<u>time</u>,doctor)

This will not work, as you need both time and Patno together to form a unique key.

Thus this determinant is not a candidate key, and therefore DB is not in BCNF. We need to fix this.

BCNF: rewrite to

R1(<u>Patno,time</u>,doctor)

R2(<u>Patno</u>,PatName)

R3(<u>time</u>,appNo)

Fourth Normal Form (4NF)

- Fourth normal form eliminates independent many-to-one relationships between columns.
- To be in Fourth Normal Form,

RULE 1: a relation must first be in Boyce-Codd Normal Form.

RULE 2 : a given relation may not contain more than one multi-valued attribute / dependency.

A table is said to have multi-valued dependency, if the following conditions are true,

- For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.
- Also, a table should have at-least 3 columns for it to have a multi-valued dependency.
- And, for a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

A	B	C
1	2	X
1	3	Y
1	4	Z

A->B

Fourth Normal Form (4NF)

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

As you can see in the table above, student with s_id **1** has opted for two courses, **Science** and **Maths**, and has two hobbies, **Cricket** and **Hockey**.

PROBLEM?

Well the two records for student with s_id **1**, will give rise to two more records, as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified.

Fourth Normal Form (4NF)

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

- As you can see in the table above, there is no relationship between the columns course and hobby. They are independent of each other.
- So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

Fourth Normal Form (4NF)

Example 2 (Not in 4NF)

Scheme → {Employee, Skill, ForeignLanguage}

1. Primary Key → {Employee, Skill, Language }
2. Each employee can speak multiple languages
3. Each employee can have multiple skills
4. Thus violates 4NF

Employee	Skill	Language
1234	Cooking	French
1234	Cooking	German
1453	Carpentry	Spanish
1453	Cooking	Spanish
2345	Cooking	Spanish

4NF - Decomposition

1. Move the two multi-valued relations to separate tables
2. Identify a primary key for each of the new entity.

Example 1 (Convert to 4NF)

CourseOpted Table

s_id	course
1	Science
1	Maths
2	C#
2	Php

And, **Hobbies Table**,

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

4NF - Decomposition

Example 2 (Convert to 4NF)

Old Scheme → {Employee, Skill, ForeignLanguage}

New Scheme → {Employee, Skill}

New Scheme → {Employee, ForeignLanguage}

Employee	Skill
1234	Cooking
1453	Carpentry
1453	Cooking
2345	Cooking

Employee	Language
1234	French
1234	German
1453	Spanish
2345	Spanish

Decomposition – Loss of Information

1. If decomposition does not cause any loss of information it is called a **lossless** decomposition.
2. If a decomposition does not cause any dependencies to be lost it is called a **dependency-preserving** decomposition.
3. Any table scheme can be decomposed in a lossless way into a collection of smaller schemas that are in BCNF form. However the dependency preservation is not guaranteed.
4. Any table can be decomposed in a lossless way into 3rd normal form that also preserves the dependencies.
   - 3NF may be better than BCNF in some cases

Use your own judgment when decomposing schemas

Lossless Decomposition

Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice. The information will not lose from the relation when decomposed. The join would result in the same original relation.

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Lossless Decomposition

Decompose the above table into two tables:

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	Emp_ID	Dept_Name
Dpt1	E001	Operations
Dpt2	E002	HR
Dpt3	E003	Finance

Lossless Decomposition

Now, Natural Join is applied on the above two tables −

The result will be −

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Therefore, the above relation had lossless decomposition i.e. no loss of information.

Lossless Decomposition

Consider there is a relation R which is decomposed into sub relations R1 , R2 , …. , Rn. This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.

For lossless join decomposition, we always have-

$$R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots\ldots \bowtie R_n = R$$

where $\bowtie$ is a natural join operator

HOW TO CHECK: Decomposition of a relation R into R1 and R2 is a lossless-join decomposition if

R1 ∩ R2 = Forms a super key in R1

OR

R1 ∩ R2 = Forms a super key in R2

Example 1: Lossless Decomposition

$R(ABC)$, decomposed in $R_1(AB)$ and $R_2(BC)$. check whether lossless or lossy?

Given: FD of R $\{A \rightarrow B\}$

$R_1 \cap R_2 = \{B\} \rightarrow$ Not a key in either R_1 or R_2

↳ so, lossy.

EX 2: (A)+={A,B}

(2) $R(ABC)$, decomposition of $R_1(AB)$, $R_2(AC)$ is lossless or lossy? FD = $\{A \rightarrow B\}$

→ $(R_1 \cap R_2) = \{A\}$ ↳ key in relation R_1

↳ lossless.

Example 2: Lossless Decomposition

Consider a relation schema R (A , B , C , D) with the functional dependencies {A → B, C → D}. Determine whether the decomposition of R into R1 (A , B) and R2 (C , D) is lossless or lossy.

ANS: Find R1 ∩ R2.

So, we have-

R1 (A , B) ∩ R2 (C , D) = Φ

Clearly it can not be super key of any relation because it NULL. Thus, we can conclude that the decomposition is lossy.

Example 3: Lossless Decomposition

Consider a relation schema R (A , B , C , D , E) with the following functional dependencies- F={ AB → CD, A → E, C → D}. Determine whether the decomposition of R into R1 (ABC) , R2 (BCD) and R3 (DE) is lossless or lossy.

ANS: YOU CAN START WITH ANY TWO RELATIONS AND FIND INTERSECTION.

R1 INT R2 = (ABC) INT (BCD) = (BC)+ = {B,C,D} = SUPER KEY R2

R12 (ABCD) INT R3 = (ABCD) INT (DE) = (D)+ = { D}

LOSSY

Example 3: Lossless Decomposition

Consider a relation schema R (A , B , C , D , E) with the following functional dependencies- F={ AB → CD, A → E, C → D}. Determine whether the decomposition of R into R1 (ABC) , R2 (BCD) and R3 (DE) is lossless or lossy.

ANS: YOU CAN START WITH ANY TWO RELATIONS AND FIND INTERSECTION.

$(R_1 \cap R_2) = \{BC\} \Rightarrow (BC)^+$

$= \{BCD\} = R_2$

$(\overset{R_1}{ABCD} \cap R_3) = \{CD\} \Rightarrow (CD)^+$

$= \{CD\} \Rightarrow$ lossy

EXAMPLE

Consider a relation schema

R (A , B , C , D , E) with the following functional dependencies-

F={ $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, D->E}. Determine whether the decomposition of R into

R1 (ABC) , R2 (BCD) and R3 (CDE) is lossless or lossy

R1 INT R2 = R12(ABCD) INT R3 = ABCD INT CDE

(CD)+=CDE

Example 4: Lossless Decomposition

Consider a relation schema R (A , B , C , D , E) with the following functional dependencies- F={A → B, B → C, C → D, D → B}. Determine whether the decomposition of R into R1 (AB) , R2 (BC) and R3 (BD) is lossless or lossy.

testing:

R1: every attributes of original relation should be covered

R2: $R_1 \cap R_2 \neq \emptyset$

R3: $R_1 \cap R_2$ should be key of any one relation.

Dependency Preserving Decomposition

- If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from combination of FD's of R1 and R2.

- A decomposition of a relation R into R1, R2, R3, …, Rn is dependency preserving decomposition with respect to the set of Functional Dependencies F that hold on R only if the following is hold;

$$(F1 \cup F2 \cup F3 \cup \ldots \cup Fn)^+ = F^+$$

Dependency Preserving Decomposition

Few key points:

- We would like to check easily that updates to the database do not result in illegal relations being created.
- It would be nice if our design allowed us to check updates without having to compute natural joins.
- We can permit a non-dependency preserving decomposition if the database is static. That is, if there is no new insertion or update in the future.

EX1: Dependency Preserving Decomposition

① $R(ABC)$ FD $= \{A \to B, B \to C\}$

Decomposition of R: $R_1(A, C)$, $R_2(B, C)$

a) FD's holds in R_1:

(Compute closure of each att. in R_1 w.r.t. original FD (F)):

$(A)^+ = \{ABC\}$, $(C)^+ = \{C\}$

$R_1 : \{A \to C\} \leftarrow F1$

b) FD holds in R_2:

$(B)^+ = \{BC\}$, $(C)^+ = \{C\}$

$\{B \to C\} \leftarrow F2$

$(F1)^+ = \{A \to A, C \to C, A \to C, AC \to AC\}$

$(F2)^+ = \{B \to B, C \to C, B \to C, BC \to BC\}$

} Find set of non-trivial FDs.

FDs: $\{A \to C, B \to C\}$

A->C

F1 UNION F2 = { A->C, B->C}

F=\{A->B, B->C\}

EX2: Dependency Preserving Decomposition

(2) $R(ABC)$ FD. $= \{A \to B,\ B \to C\}$

$R_1(AB)$, $R_2(BC)$

$F_1(A \to B)$, $F_2(B \to C)$

$(F1)^+ = \{A \to A,\ B \to B,\ A \to B,\ AB \to AB\}$

$(F2)^+ = \{B \to B,\ C \to C,\ B \to C,\ BC \to BC\}$

$(F_1 \cup F_2)^+ = \{A \to B,\ B \to C\} = (F)^+$

$\therefore$ so dependency preserving

EX3: Dependency Preserving Decomposition

(3) R (city, street, zipcode)

FD: { city, street → zipcode,

zipcode → city }

R_1 (street, zipcode), R_2 (city, zipcode)

Prachi Shah

EX3: Dependency Preserving Decomposition

$(street)^+ = \{street\}$

$\{zipcode\}^+ = \{zipcode, city\}$

so $F1 = \{street \to street,\ zipcode \to zipcode,$
$street, zipcode \to street, zipcode,$
$zipcode \to city\}$

$(city)^+ = \{city\}$

$F2 = \{city \to city,\ zipcode \to zipcode,$
$zipcode \to city,$
$city\ zipcode \to city\ zipcode\}$

$(F1 \cup F2)^+ = \{zipcode \to city\} \neq (F)^+$

so not ~~decompo~~ dependency preserving

EX4: Dependency Preserving Decomposition

4. $R(ABCD)$ $FD = \{A \to B, B \to C, C \to D, D \to A\}$

$D(AB, BC, CD)$

Fifth Normal Form (5NF)

- Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy.
- Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.
- 5NF is also known as Project-join normal form (PJ/NF).

RULE 1: The relation should be already in 4NF.

RULE 2: The relation cannot be further / additionally non loss decomposed (join dependency).

Fifth Normal Form (5NF)

ex: is ~~there~~ the table in 5NF?

pname	skill	Job
Aman	DBA	J1
mohan	Tester	J2
Rohan	prog.	J3
sohan	Analyst	J1

→ Decompose & check for join dependency.

pname	skill
Aman	DBA
mohan	tester
Rohan	prog.
sohan	Analyst

pname	Job
Aman	J1
mohan	J2
Rohan	J3
sohan	J1

skill	Job
DBA	J1
tester	J2
prog.	J3
Analyst	J1

Fifth Normal Form (5NF)

$((R_1 \bowtie R_2) \bowtie R_3) = R$

pname	skill	job
Aman	DBA	J1
mohan	teacher	J2
sohon	prog.	J3
sohan	Analyst	J1

'R' is not in 5NF.

The End..!!

Thank You….