**DBMS (R20) 2nd Year 1st Semester 2021-22 AY**

**Unit-4 Syllabus**

**Schema Refinement** (Normalization): Purpose of Normalization or Schema Refinement, Concept of Functional Dependency, Normal Forms Based on Functional Dependency (1NF, 2NF and 3 NF), Concept of Surrogate Key, Boyce-Codd Normal Form (BCNF), Lossless Join and Dependency Preserving Decomposition, Fourth Normal Form (4NF), Fifth Normal Form (5NF).

**Schema Refinement & Normalization**

**Functional Dependency**:

Let R be a relation, let X and Y are two sets of attributes of R. For every pair of tuples t1 and t2 in an instance r of R, if t1[X] =t2[X] => t1[Y] =t2[Y], then the functional dependency X→Y holds on R. i.e., for every pair of tuples t1 and t2 with same value on X, the values in Y must be same.

Ex:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | D |
| a1 | b1 | c1 | d1 |
| a1 | b1 | c1 | d2 |
| a1 | b2 | c2 | d1 |
| a2 | b1 | c3 | d1 |

Consider the above instance.

The following FDs hold on the relation: C→B and C→A. The following FDs do not hold on the relation: B→C and A→B.

**Note**:

By looking at an instance it is meaningful to say that an FD doesn’t hold. But it isn’t meaningful to say a FD holds on a relation by looking at one or even more instances.

**Closure of Set of Functional Dependencies**:

Let R be a relational schema. Let F is the set of functional dependencies that hold on R. The set of all Functional Dependencies that are logically implied from F is called closure of F and is denoted by F+.

**Ex**:

Let R (A,B,C) F={A→B,B→C} Then F+ ={A→B , B→C , A→C}

In a FD X→Y, X is called the **determinant** and Y is called **dependent**.

**Armstrong’s Axioms**:

These are used to find closure of a set F of functional dependencies.

1. **Reflexive Rule**: If X ⊇Y then X→Y
2. **Augmentation Rule**: If X→Y then XZ→YZ
3. **Transitivity Rule**: If X→Y & Y→Z then X→Z, Where X,Y and Z are sets of attributes of R.

**Additional Rules:**

Even though, Armstrong’s axioms are enough to compute closure of F, using them alone is time consuming and tedious process. Hence, few additional rules are proposed.

1. **Union Rule**: If X→Y and X→Z then X→YZ.
2. **Decomposition Rule**: If X→YZ then X→Y & X→Z.
3. **Pseudo Transitivity Rule**: If X→Y and YZ→W then XZ→W where W is a set of attributes of R.

Armstrong’s axioms are sound & complete i.e., they do not produce any wrong FDs and they are enough to produce all implied FDs.

Examples:

1. Consider R (A,B,C,G,H,I) and F= {A→B,A→C,CG→H,CG→I,B→H},

Check whether AG→I can be implied and also determine F+.

Sol: Consider A→C & CG→I

By pseudo transitivity, AG→I (Replace C with its determinant)

Consider A→C, CG→H =>AG→H

A→B, B→H =>A→H

A→C, CG→I =>AG→I

∴F+={A→B ,A→C ,CG→H ,CG→I ,B→H ,AG→H ,A→H ,AG→I}

1. Consider R(A,B,C,D,E)

F={A→BC , CD→E , B→D ,E→A} F+=?

Sol: A→BC =>A→B,A→C

B→D , CD→C =>CB→E

A→B , B→D =>A→D

A→C , CD→E =>AD→E

E→A , A→BC => E→BC

F+= {F}U{AD→E , A→D, E→BC, BC→E}

Consider the FD’s: A→D & AD→E

Here, D is not necessary in AD→E. D is called extraneous attribute in AD→E and can be removed. It is enough to write A→E instead of AD→E.

**Closure of Set of** A**ttributes**:

Let R is a relation schema and F is a set of FDs that hold on R.

Let X is a set of attributes of R. The closure of X is the set of all attributes of R that can be logically determined from X under F. The closure of ‘X’ is denoted by X+.

1. Let R (A,B,C,D,E) and F={A→BC, CD→E , B→D, E→A}

Compute A+ and {CD}+.

Sol: Start with {A}+={A}

A→BC =>A+={A,B,C}, B→D =>A+={A,B,C,D}, CD→E =>A+={A,B,C,D,E}

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

Similarly,

Start with {CD}+={C,D} , CD→E =>{C,D} = {C,D,E},

E→A =>{CD}+ = {C,D,E,A}, A→B =>{CD}+={C,D,E,A,B}

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

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

As {A}+ and {CD}+ includes all attributes of R and as both are minimal, {A} and {CD}

are candidate keys of R. As E→A, E+=A+. ∴ {E} is third candidate key of R.

\*\*\*If the closure of a set of attributes X computed under F includes all attributes of the relation R and if X is minimal, i.e., No FDs exist among attribute of X, then X is a ***Candidate Key*** of R. Some relation may have multiple candidate keys. In such a case, one of them is designated as ***Primary Key***of the table. A candidate key is simply called ***Key***. All supersets of a candidate key are called ***Super Keys***. All attributes present in all candidate keys together are called ***Key Attributes*** or ***Prime* *Attributes***. The attributes that are not members of any key are called ***Non-Key Attributes***. \*\*\*

1. Given R(A,B,C,D,E)

F1= {A→B, AB→C , D→AC ,D→E}

F2= {A→BC, D→AE}

Check whether F1 and F2 are equivalent.

Sol: Consider F1:

D→AC =>D→A, D→C

A→B, AB→ C => A→C

F1 = {A→B, A→C, D→C,D→E ,D→A}

F2 = {A→B, A→C, D→A, D→E}

= {A→B, A→C , D→C, D→A , D→E})=F1.

(Or)

F1= {A→B , A→C, D→A,D→E}

= {A→BC,D→AE}=F2. Hence, F1 and F2 are equivalent.

An attribute “A” is ***extraneous*** in a FD X→Y (either on left hand side or on right hand side) if we can safely remove “A” without changing the closure F+ of F.

Ex: 1) if AB→C and A→C, then B is extraneous.

2) If AB→CD and A→C, then C is extraneous in ‘CD’.

**Canonical Cover/Minimal Cover of** **FDs**:

A canonical cover Fc for F is a set of FDs such that Fc logically implies all FDs in F+ and F logically implies all FD’s in Fc+. There shouldn’t be any extraneous attribute in Fc .

Each determinant in Fc must be unique.

1. Given R(A,B,C) F={A→BC , B→C , A→B , AB→C}

Find an irreducible equivalent (minimal cover) for F.

Sol:

1. Rewrite all FDs to contain single attribute on RHS.

A→B (1), A→C (2), B→C (3) , A→B(4) , AB→C(5)

ii) 1 and 4 are same. Hence, remove (4)

B in 5 is extraneous because A→B. Hence (5) can be written as A→C which is same as FD (2). Hence, remove (5). FD (2) can be removed because it is implied from (1) & (3).

∴ Fc = {A→B, B→C} is irreducible equivalent to F.

1. Given R(A,B,C) F={A→BC , B→AC ,C→AB} Determine Fc.

Sol:

1. A→B , A→C , B→A , B→C , C→A , C→B

(1) (2) (3) (4) (5) (6)

ii) A→B , B→C =>A→C Hence, remove (2)

C→A , A→B =>C→B Hence, remove (6)

B→C , B→A =>B→A Hence, remove (3)

∴ Minimal Cover of F, Fc ={A→B , B→C , C→A}

1. Given R(A,B,C,D,E,F) and

F={AB→C,C→A,BC→D,ACD→B,BE→C,EC→FA, CF→BD,D→E}

Determine minimal cover Fc and also find candidate keys of R.

Sol:

i)AB→C , C→A , BC→D , ACD→B ,BE→C , EC→F, EC→A , CF→B,

(1) (2) (3) (4) (5) (6) (7) (8)

CF→D , D→E.

(9) (10)

In (7),’E’ is extraneous since C→A. Hence, remove E from (7) and then remove (7) because it is equivalent to (2).

In (4),’A’ is extraneous since C→A. Hence, (4) becomes CD→B. No more reductions or removals are possible. Therefore,

Fc ={AB→C , C→A , BC→D, CD→B ,BE→C ,EC→F ,CF→BD ,D→E}.

Finding candidate keys:

Consider {AB}+ ={A,B,C} (from AB→C)

BC→D =>{AB}+ ={A,B,C,D}

D→E =>{AB}+ ={A,B,C,D,E}

EC→F =>{AB}+ ={A,B,C,D,E,F}

Consider {BC}+ ={B,C,D} (from BC→D)

D→E=> {BC}+ ={B,C,D,E}

EC→F=> {BC}+ ={B,C,D,E,F}

C→A=> {BC}+ ={A,B,C,D,E,F}

{CD}+ ={A,B,C,D,E,F}(As CD→B, {CD}+ will be same as that of {BC}+)

Hence, Candidate keys are: {AB},{BC},{CD}

1. Giver R(A,B,C,D,E,F,G,H) and F={A→B, ABCD→E, EF→GH, ACDF→EG}.

Determine an irreducible equivalent to F.

Sol: Rewrite all FDs to contain single attribute on RHS.

A→B, ABCD→E, EF→G, EF→H, ACDF→E, ACDF→G

1. (2) (3) (4) (5) (6)

----In, FD (2), B is extraneous as A→B. Hence remove B to form ACD→E.

----As ACD→E, in FD (5) we can remove F to make ACD→E. Now, FD (5) is same as

FD (2). Hence, remove FD (5).

---- From revised FD (2) and FD (3), we get FD (6). (Pseudo Transitivity Rule) Hence,

remove FD (6). No more reductions are possible.

Finally, Fc={A→B, ACD→E, EF→GH} is irreducible equivalent to F.

**Anomalies Due to Bad Database Design**:

Consider the following relation instance:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sid** | **Sname** | **Rating** | **Hrly\_Wage** |
| 21 | John | 8 | 200 |
| 22 | Smith | 9 | 350 |
| 23 | Horatio | 8 | 200 |
| 24 | Smith | 8 | 200 |
| 25 | Johnson | 7 | 150 |
| 26 | Lubber | 9 | 350 |

The FDs that hold on above relation are :

{Sid}→{Sname, Rating, Hrly\_Wage}

{Rating}→{Hrly\_Wage}

The above relation has redundancy in the form of <Rating, Hrly\_Wage> pairs are stored at multiple places. Due to this redundancy, the following anomalies occur:

1. **Update Anomaly**: The problem of updating one copy of redundant data without having a similar update on other copies. ( We may update Hrly\_Wage 200 as 300 for one record leaving other two records with 200)
2. **Insertion Anomaly**: The inability to insert useful data without inserting unwanted data as well. (Ex: to insert <10,500> as <Rating, Hrly\_Wage> pair, we must have a sailor with rating 10)
3. **Deletion Anomaly**: The problem of losing useful data while deleting unwanted data. (If the sailor 25 leaves the club, we miss the Hrly\_Wage value for rating 7 as there is only one sailor with rating 7).

These anomalies can be solved by decomposing the given table into two tables as follows:

|  |  |  |
| --- | --- | --- |
| **Sid** | **Sname** | **Rating** |
| 21 | John | 8 |
| 22 | Smith | 9 |
| 23 | Horatio | 8 |
| 24 | Smith | 8 |
| 25 | Johnson | 7 |
| 26 | Lubber | 9 |

|  |  |
| --- | --- |
| **Rating** | **Hrly\_Wage** |
| 7 | 150 |
| 8 | 200 |
| 9 | 350 |

**Normal Form**: A normal form defines the state of a relation with respect to functional dependencies defined on that relation. By knowing the normal form of a relation, we are sure that certain kind of problems does not occur whereas certain other kinds of problems may occur. If we want to remove those problems also the relation must be refined to a higher level.

Based on FDs, the following normal forms are defined:

-First Normal Form (1 NF)

-Second Normal Form (2 NF)

-Third Normal Form (3 NF)

-Boyce Codd Normal Form (BCNF)

These normal forms have increasingly restrictive requirements i.e., a relation which is in a higher level NF will be in all lower level NFs.

**Well Structured Relation**: A relation which is free from redundancy and upon which we can safely perform DML operations is called a well structured relation.

**Normalization**: The process of decomposing a relation with anomalies to form smaller and well structured relations is called normalization.

Consider the following instance:

|  |  |
| --- | --- |
| P | D |
| p1 | d1 |
| p2 | d2 |
| p1 | d2 |

|  |  |  |
| --- | --- | --- |
| S | P | D |
| s1 | p1 | d1 |
| s1 | p2 | d2 |
| s2 | p1 | d2 |

|  |  |
| --- | --- |
| S | P |
| s1 | p1 |
| s1 | p2 |
| s2 | p1 |

An instance of relation SPD Instance of SP Instance of PD

|  |  |  |
| --- | --- | --- |
| S | P | D |
| s1 | p1 | d1 |
| s1 | p1 | d2 |
| s1 | p2 | d2 |
| s2 | p1 | d1 |
| s2 | p1 | d2 |

SP ⋈PD

In the above decomposition the original instance was not recollected after joining the smaller instances. Such decomposition is called lossy decomposition.

Any decomposition should satisfy the following requirements:

1. Lossless Join Decomposition.
2. Dependency Preserving Decomposition.

**Partial FD**: A FD in which a non-key attribute is determined by part of the key rather than full key is called Partial FD. (**Ex**: If A and B are key attributes, then the FD: A→C is called partial FD where C is a non-key attribute.)

**Transitive FD**: A FD that exists among non-key attributes is called Transitive FD. (Ex: If A and B are key attributes, C and D are non-key attributes of a relation R, then the FD: C→D is called Transitive FD.)

**First Normal Form** (1NF):

A relation R is in 1NF if it does not contain any multi valued attributes.

Consider the following relation (R1):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Empno | Ename | Sal | Course | Date\_of\_completion |
| 101 | John | 30000 | C++  DBMS | 10/06/2018  15/07/2018 |
| 102 | Smith | 40000 | Unix | 12/06/2018 |
| 103 | John | 30000 | DBMS  Unix | 10/06/2018  12/06/2018 |

R1 is not in 1NF as it contains multi-valued attributes “course” and “date\_of\_completion”. It can be converted into 1NF as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Empno | Ename | Sal | Course | Date\_of\_completion |
| 101 | John | 30000 | C++ | 10/06/2018 |
| 101 | John | 30000 | DBMS | 15/07/2018 |
| 102 | Smith | 40000 | Unix | 12/06/2018 |
| 102 | Smith | 40000 | DBMS | 10/06/2018 |
| 103 | John | 30000 | Unix | 12/06/2018 |

Key: <empno,course>

**Second Normal Form** (2NF): A relation R is in 2NF if it is in 1NF and does not contain any partial functional dependencies. A functional dependency in which a non-key attribute depends on only part of the key is called “Partial FD”.

Ex: R(A,B,C,D) F={A→B,C→D} key is {AC}

A→B and C→D are partial FDs.

Ex 1: Consider the following relation:

R(Empno, Ename, Sal, Course,DoC) (DoC is the Date of Completion of the course)

FD1:{Empno,Course}→{DOC}

FD2: {Empno}→{Ename,Sal} (Partial FD)

R is not in 2NF due to the presence of partial FD (FD2).It can be decomposed into a collection of 2NF relations as follows:

R1(**Empno**, Ename, Sal)

R

R2( **Empno, Course**, DoC) (Empno of R2 is foreign key referencing R1)

R1 and R2 are in 2NF.

Ex2: Consider a relation: Stock(store,product,cost,qty,mgr)

FD1:{Store}→{Mgr} (partial FD)

FD2:{Product}→{Cost} (partial FD)

FD3:{Store ,Product}→{Qty}

Due to the presence of partial FDs, above relation is not in 2NF. It can be converted into a collection of 2NF relations as follows:

Store( **Store**, Mgr)

Stock

Store\_Stock( **Store, Product**, Cost, Qty)

Product( **Product**, Cost) Product\_Qty( **Product, Store**, Qty)

**Third Normal Form** (3NF): A relation ‘R’ is in 3NF if it is in 2NF and does not contain any transitive functional dependencies. A FD in which a non-key attribute determines other non-key attribute is called “Transitive FD”.

Ex1:

Consider R(A,B,C,D) F = { A →B, B→C, A→D }

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

Hence, {A} is the key

Hence B,C and D are called non-key attributes

Based on F, R is 2NF but not 3NF.

R is decomposed into a collection of 3NF relations as follows. R1( **B**, C), R2(**A**, B, D)

Ex 2: Consider R(Bus\_No, Origin, Destination, Distance}

FD1: {Bus\_No}→{Origin,Destination,Distance}

FD2: {Origin,Destination}→{Distance}

Here, FD2 is partial FD. Bus\_No is key attribute and all others are non-key attributes. R is decomposed into a collection of 3NF relations as follows.

R1( **Origin, Destination**, Distance) R2( **Bus\_No**, Origin, Destination)

**Boyce-Codd Normal Form**: A relation ‘R’ is in Boyce-Codd normal form if it is in 3 NF and if the determinant of every FD is a key.

Ex 1:

Consider R(Teacher#, Student#, Course#, Grade) and following FDs:

FD1: {Teacher#}→{Course#}

FD2: {Teacher#, Student#}→{Course#, Grade}

FD3: {Student#, Course#}→{Teacher#, Grade}

In FD1, the determinant is not a key. Hence, R is not in BCNF. It can be decomposed into R collection of 2NF relations as follows:

R1( **Teacher#**, Course#) R2( **Student#, Teacher#**, Grade)

Ex 2:

Consider R(A,B,C,D) and F ={A→B, BC→D, D→E, E→A}

Find all candidate keys of R. Find the best normal form that R satisfies. Decompose R into a collection of BCNF relations.

Sol:

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

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

Therefore, the candidate keys are {AC}, {DC}, {EC} and {BC}

Hence, all attributes are key attributes or prime attributes. Hence, R is in 3NF. But, except the FD, BC→D, remaining FDs violate BCNF. Hence, the following decomposition is a collection of BCNF relations.

R1(**E**,A), R2( **B, C**, D, E) (in 2NF, But not in 3NF)

Now, decompose R2 into R21(**D**,E) and R22(**B,C**, D). Now, {R1, R21,R22} is a BCNF collection of R.

---***When a relation is in BCNF, no redundancy can be found based on FD information alone.***

- Consider the following example.

|  |  |  |
| --- | --- | --- |
| X | Y | A |
| x | y1 | a |
| x | y2 | ? |

Let R is in BCNF. Let the FD, X→A holds on R. Hence, the value of A in second tuple should be ‘a’. This appears as (X,A) pairs are redundantly stored.

But when R is in BCNF, X must be a key and hence it must determine Y also. Hence, the value of Y should be same (either y1 or y2) in both the tuples. i.e., the 2 tuples represent a single tuple. Hence, there is no redundancy.

**Desired Properties of Normalization (or) Normalization techniques**:

---Any good decomposition should satisfy 2 properties.

**Lossless join**: Let R a relation which is decomposed into R1 and R2 with sets of attributes X and Y. Let r is an instance of R.

## If *πX*(r) ⋈*π*Y(r) = r, then the decomposition of R is would be a lossless join.

**Test**: Let R be a relation and F be the set of FDs on R.

Let R is decomposed into R1 and R2

If either the FD R1**∩**R2 →R1 or R1**∩**R2 →R2 is in F+, then we say the decomposition is lossless.

**Dependency Preservation**:

If we are able to enforce each of the original FDs on smaller relations without performing a join, such a decomposition is said to be dependency preserving.

**Test**:

Let a relation R with a set of FDs ‘F’ be decomposed into 2 relations with sets of attributes X and Y.

--- Let Fx be the set of FDs from F+ that contain only attributes in X.

--- Let Fy be the set of FDs from F+ that contain only attributes in Y.

if (Fx U Fy ) = F+, the decomposition is dependency preserving.

**Ex**: Let R(A,B,C)

F = { A→ B, B→C, C→A} is split into R1(A,B) and R2(B,C)

Is this decomposition dependency preserving?

Sol:

F+ = {A→B, B→C, C→A, C→B, A→C, B→A}

X = {A,B} Y = {B,C}

Fx  = {A → B, B→A}

Fy = {B→C,C→B}

Fx U Fy = {A→B, B→C, C→B, B→A}

(Fx  U Fy)+ = {A→B, B→C,C→B, B→A,A→C,C→A}

= F+

Hence, the decomposition is dependency preserving.

**Comparison of 3NF and BCNF**:

i) When a relation is in 3NF, redundancy may present. When a relation is in BCNF, no redundancy can be found based on FDs.

ii) Both 3NF and BCNF ensures lossless join decomposition.

iii) It is always possible to ensure a dependency preserving decomposition into 3NF. BCNF does not ensure dependency preservation.

**Note**:

Databases are Select intensive and storage intensive.

Storage Intensive- Higher level normalization,

Select Intensive - Lower level normalization.

**Multi Valued Dependencies**:

Consider the following relation R(Course, Teacher, Book) I.e., R(C,T,B). The meaning of a tuple is teacher T teaches course C and B is a recommended book for C.

|  |  |  |
| --- | --- | --- |
| **Course** | **Teacher** | **Book** |
| Physics101 | Green(t1) | Optics |
| Physics101 | Green(t3) | Mechanics |
| Physics101 | Brown(t2) | Mechanics |
| Physics101 | Brown | Optics |
| Math301 | Green | Mechanics |
| Math301 | Green | Vectors |
| Math301 | Green | Geometry |

From the above instance, we can observe the following points.

- The key for R is {Course, Teacher, Book}.

- The schema is in BCNF. Hence, no need to decompose it further.

- Still, there is redundancy in R.

- The redundancy here is due to the fact that recommended books for a course are independent of instructors. This constraint cannot be expressed in terms of FDs.

This is an example of multi valued dependencies.

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | Z |  |
| a | b1 | c1 | ----tuple t1 |
| a | b2 | c2 | ----tuple t2 |
| a | b1 | c2 | ----tuple t3 |
| a | b2 | c1 | ----tuple t4 |

In the above instance, t1.x = t2.x, t1.xy = t3.xy and t2.z = t3.z

Let R be a relational schema and X and Y are subsets of attributes of R.

The MVD X→→Y is said to hold over R if in every legal instance r of R, each X value is associated with a set of Y values and this set is independent of the values in other attributes.

Consider the above instance.

Let r is an instance of R and X and Y, are subsets of attributes of R. Let Z = R - XY.

Let tuples t1 ∈r, t2 ∈r. The MVD X →→Y said to hold over R if t1.x = t2.x, then there must be some t3 ∈r such that t1.XY = t3.XY and t2.Z = t3.Z

**Fourth Normal Form** (4 NF):

A relation R with a set of FDs and MVDs is said to be in 4NF, if it in BCNF and for every MVD, X→→ Y one of the following must be true.

- Y ⊆ X

- X is a super key.

- XY = R

The relation R(Course, Teacher, Book) is not in 4NF due to the presence of the MVDs C→→T and C→→B.

- R can be decomposed into R1(Course, Teacher) and R2(Course, Book) which are in 4NF.

**Concept of Surrogate key**:

Surrogate key also called a synthetic primary key, which is automatically generated by DBMS when a new record is inserted into a table. The surrogate key can be declared as the primary key of that table. It is the sequential number outside of the database that is made available to the user and the application or it acts as an object that is present in the database but is not visible to the user or application.

We can say that, in case we do not have a natural primary key in a table, then we need to artificially create one in order to uniquely identify a row in the table, this key is called the surrogate key or synthetic primary key of the table. However, surrogate key is not always the primary key. Suppose we have multiple objects in a database that are connected to the surrogate key, then we will have many-to-one association between the primary keys and the surrogate key and surrogate key cannot be used as the primary key.

**Features of the surrogate key:**

* It is automatically generated by the system.
* It holds anonymous integer.
* It contains unique value for each record of the table.
* The value can never be modified by the user or application.
* Surrogate key is called the fact less key as it is added just for our ease of identification of unique values and contains no relevant fact (or information) that is useful for the table.

In a temporal database that stores data relating to time instances, it is necessary to distinguish between the surrogate key and the business key. Every row would have both a business key and a surrogate key. The surrogate key identifies one unique row in the database and the business key identifies one unique entity of the modelled world. One table row represents a slice of time holding all the entity's attributes for a defined time span. For example, a table *Employee\_Contracts* may hold temporal information to keep track of contracted working hours. The business key for one contract will be identical (non-unique) in both rows however the surrogate key for each row is unique.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Surrogate Key** | **Business Key** | **Employee Name** | **Working Hours Per Week** | **Row Valid From** | **Row Valid To** |
| 1 | BOS0120 | John Smith | 40 | 2000-01-01 | 2000-12-31 |
| 56 | P0000123 | Bob Brown | 25 | 1999-01-01 | 2011-12-31 |
| 234 | BOS0120 | John Smith | 35 | 2001-01-01 | 2009-12-31 |

**Join Dependencies and 5th Normal Form** (Projection Join Normal Form):

**Join Dependency:**  
If the join of R1 and R2 over Q is equal to relation R then we can say that a join dependency exists, where R1 and R2 are the decomposition R1 (P, Q) and R2 (Q, S) of a given relation R (P, Q, S). R1 and R2 are a lossless decomposition of R.

**Fifth normal form (5NF)** is also known as **Project-Join Normal Form (PJNF)**. It is a level of database normalization designed to reduce redundancy in relational databases. A relation is said to be in 5NF if and only if it satisfies 4NF and no join dependency exists. A relation is said to have join dependency if it can be recreated by joining multiple sub relations and each of these sub relations has a subset of the attributes of the original relation.

**Example:**

Consider the relation **R** below having the schema R(Supplier, Product, Consumer). The primary key is a combination of all three attributes of the relation.

|  |  |  |
| --- | --- | --- |
| Supplier | Product | Consumer |
| S1 | P1 | C2 |
| S1 | P2 | C1 |
| S2 | P1 | C1 |
| S1 | P1 | C1 |

|  |  |
| --- | --- |
| Supplier | Product |
| S1 | P1 |
| S1 | P2 |
| S2 | P1 |
|  |  |

**Table 1 Table 2**

|  |  |
| --- | --- |
| Consumer | Product |
| C2 | P1 |
| C1 | P2 |
| C1 | P1 |

|  |  |
| --- | --- |
| Supplier | Consumer |
| S1 | C2 |
| S1 | C1 |
| S2 | C1 |

**Table 3**

**Table 4**

The table Table1 has no FDs and no MVDs. The key for the table is {Supplier, Product, Consumer}. Hence, the table is in 4NF. Still, there is redundancy in the table in the form of <S1,P1> pair and <P1,C1> pair are redundantly stored. This redundancy is due to join dependency.   
⋈{Table 2,Table 3,Table 4} gives the original instance of the table (Table 1). Hence join dependency exists in Table 1. Therefore, Table 1 is not in 5NF or PJNF. However Table 2, Table 3 and Table 4 satisfy 5NF as they have no multi valued dependency and cannot be decomposed further. But this might not be true in all cases i.e., when we combine the decomposed tables, the resultant table may not be equivalent to the original table, in that case the original table is said to be in 5NF provided it is already in 4NF. However, 5NF is not applied in practical scenarios and remains limited to theoretical concepts.

**Ex** 1: BCNF Decomposition Example: (With and without dependency preservation)

Let R = (A, B, C, D, E) and F = {A → B, BC → D} Hence, Candidate key is {ACE}. Both the FDs violate BCNF. Consider the decomposition of R into R1(A, B) and R2(A, C, D, E), where the projection of F on R1 is F1 = {A →B} and that on R2 is F2 = {AC→ D} (AC→ D is obtained from A → B and BC → D by pseudo-transitivity). The Candidate key of R1 is {A}. Hence, R1 satisfies BCNF. But, R2 is not in BCNF because the determinant in the FD: {AC→ D} is not candidate key of R2. Now, decompose R2 into R3(A,C,D) and R4(A,C,E). Now, R3 and R4 are in BCNF. Hence, {R1,R3,R4} is a BCNF collection of R.

But, this decomposition is not dependency preserving because to check BC→D, we need to join R1 and R3.

\*\*\*The following decomposition of R(A,B,C,D,E) under same set of FDs is dependency preserving. R1(B,C,D) R2(A,B) R3(A,C,E)\*\*\*.

**Ex** 2:

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

Identify all candidate keys for R. Identify the best normal form that R satisfies. Decompose R into a set of BCNF relations. Decompose R into a set of 3NF relations.

Sol: R =(A, B, C, D). F = {C→D, C→A, B→C}. The only candidate key is {B}

R is in 2NF but not in 3NF because FDs C→D and C→A are transitive. Now, decompose R into R1(**C**,D) and R2(A,**B**,C). R2 is still not in 3NF. Decompose R2 into R3(**C**,A) and R4(**B**,C) . Now, R1, R3 and R4 are in 3NF as well as in BCNF. The decomposition is both lossless and dependency preserving.

**Ex** 3: Given R = (A, B, C, D) and F = {AB→C, AB→D, C→A, D→B}

---Is R in 3NF, why? If it is not, decompose it into 3NF.

--- Is R in BCNF, why? If it is not, decompose it into BCNF

Candidate Keys of R are: {AB}, {BC}, {CD} and {AD}. Hence, all attributes are key attributes. As there are no partial or transitive FDs, R is in 3NF. But, C →A and D → B cause violation of BCNF. Hence, decompose R into R1(**D**,B) and R2(**D,A**,C). Still, R2 is not in BCNF. Hence, decompose R2 into R3(**C**,A) and R4(**D,C**).

Hence, {R1,R3,R4} is a BCNF collection of R. The decomposition is lossless but not dependency preserving.

**Ex 4:**  
Find the best normal form satisfied by the relation R(A, B, C, D, E) with FD set

F={ BC→D, AC→BE, B→E }

The only candidate key of R is {AC} because {AC}+={A,B,C,D,E}. Also, neither A nor C is determined from any other attribute. Hence, no other candidate key is exists.

 Prime attributes are those attributes which are part of candidate key. i.e., A and C in this example and B, D and E are non-prime attributes.

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

The relation is also in 2nd normal form because in BC→D, BC is not a proper subset of candidate key {AC}. In AC→BE, AC is candidate key. In B→E, B is not a proper subset of candidate key AC.

The relation is not in 3rd normal form because in BC→D, neither BC is a super key nor D is a prime attribute. In B→E neither B is a super key nor E is a prime attribute. Hence both the FDs violate the condition of 3rd normal form. So the highest normal form of relation is 2nd normal form.

To get a collection of 3NF relations, create R1 and R2 as follows.

R1(**B**,E) FR1= {B→E} R2(**A,C**,B,D) FR2= {BC→D, AC→B}. Now, R1 is in 3NF but R2 is not in 3NF as in FD BC→D, neither BC is a super key nor D is a prime attribute. Now, decompose R2 as follows. R21(**B,C**,D) R22(**A,C**,B).

Finally, {R1, R21,R22} is a 3NF collection of R. This collection is in BCNF also.

-----XXXXX-----