**1)Introduction to schema refinement**: The same information is more than one place within DB.

**A)Problems Caused By Redudancy:** That redundant information leads to several problems.

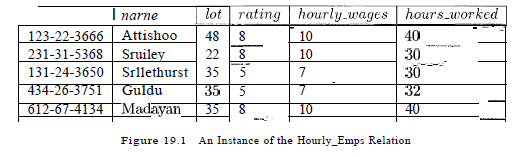
a)Redundant Storage: SOU1C information is stored repeatedly.

b) Update Anomalies: If one copy of such repeated data is updated, an

inconsistency is created unless all copies are similarly updated.

c) Insertion Anomalies: It may not be possible to store certain information unless some other, unrelated, information is stored as well.

d)Deletion Anomalies: It may not be possible to delete certain information without losing SOme other, unrelated, information as well.



FD is rating🡪 hourly-wages.

If same value appears in rating column of two tuples, same value must also appear in hourly-wages column as well. This is redundancy. Redudancy leads to problems.

*1.*Redudant *Storage:* The rating value 8 corresponds to the hourly wage 10,and this association is repeated three times.

2.Update anomalies*:* The *hourly\_wages* in the first tuple could be updated without making a similar change in the second tuple.

*3.* Insertion Anomalies*:* We cannot insert a tuple for an employee unless We know the hourly wage for the employee's rating value.

*4.*Deletion Anomalies*:* If we delete all tuples with a given rating value (e.g.,we delete the tuples for Snlcthurst and Guldu)we lose the association between that *rating* value and its *hourly\_wage* value.

The problems are eliminated by Decomposition or null.

A)NULL:- The null can’t eliminate redundant storage and update anamolies.In only some cases, The null can help to eliminate insertion&deletion anamolies.

EX:-

Problem:- we can’t insert employee record, unless we know rating value for employee hourly wages.

Solution: Null can deal insertion anomaly. We can insert employee record with null value in rating column.

Null value can’t address all insertion anamolies.

Example:- we cannot record the hourly wage for a rating unless there is an ernployee with that rating, because we cannot store a null value in the *ssn* field, which is a prirnary key field.

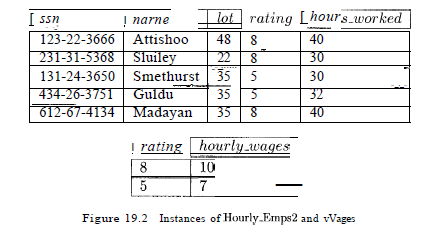
**B) Decomposition**:- The decomposition can eliminate redundancy problems. But it should be used with cautions; otherwise decomposition leads to new problems.

Decomposition is dividing relation(R) into smaller relataion.

we can decornpose hourly\_Ernps into two relations:

*Hourly\_Emps2(ssn, name, lot, rating,hours\_worked)*

Wages (*rating, hourly\_wages)*

**

**2)Functional Dependencies***:-*

The FD is kind of constraint.

R is relation Schema.

X and Y are non-empty sets of attributes in R.

We say that instance of R satisfies x🡪Y. If t1.X=t2.X , then t1.Y=t2.Y

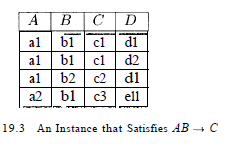
It can be read as

X functionally determines Y.

(or)

Y functionally depends on X.

Example:-1.



The FD is AB🡪 C . The A&B together functionally determines c.

Write FD’s From the following instance.

|  |  |  |
| --- | --- | --- |
| A | B | C |
| A1 | B1 | C1 |
| A1 | B1 | C2 |
| A2 | B3 | C3 |
| A2 | B1 | C2 |

The primary key is also special type of FD. The attributes in primary key play role of X and set of other attributes play role of Y.

2.1) Trival Dependencies:- A FD is trival dependencies if and only if the right side is subset to left side.

EX: 1 {a,b}🡪a

2.{s#,b#,c#}🡪{s#,b#}.

2.2) Non-Trival Dependencies:- A FD is non-trival dependencies if and only if the right side is not subset to left side.

EX: 1.{a,b}🡪c.

**3.Reasoning about FD’s:**

Workers(ssn,name,lot,did,since)

Consider following FD’s:

1. SSN🡪 did
2. Did🡪 lot.

If workers instance will hold above 2 FD’s then workers instance may hold additional functional Dependencies which is SSN🡪lot.

**3.1)Closure set of FD’s:-**

R is relation schema.

F is set of FD’s.

Closure of F is denoated as F+.

F+  is set of FD’s which is Infered/calculated from F.

Q)How is F+ is calculated from F?

A) we apply armstron’s axioms repeatedly over F to infer F+.

The Armstrong axoms are

1. Reflexivity: if x is superset or equal to y, then x🡪y.
2. Augmentation: if x🡪y then xz🡪yz for any z.
3. Transitivity: if x🡪y and y🡪z then x🡪z.
4. Union: if x🡪y and x🡪z then x🡪z.
5. Decomposition: if x🡪yz then x🡪y and x🡪z.
6. Self: A🡪A.

Armstrong axioms are sound, because they donot generate any incorrect FD’s.

Example:-

R(A,B,C)

F={A🡪B,B🡪C}

Calculate F+ using above information.

Transivity:A🡪C

Augmentation: AC🡪BC ,BA🡪CA,AB🡪BC.

Reflexivity: ABC🡪ABC.

F+={ A🡪C, AC🡪BC ,BA🡪CA,AB🡪BC, ABC🡪ABC}

3.2) **Attribute Closure:-**

If we just want to check whether a given dependency, say, *X* ---+ *Y,* is in the closure of a set ~F' of FDs, we can do so efficiently with out computing Fl+. We first compute the attribute closure X+ with respect to *F, w*hich is the set of attributes *A* such that *X* -7 *A* can be inferred using the Armstrong Axioms.

The algorithm for computing the attribute closure of a set *X* of attributes is shown in Figure 19.4.

*closure* = *X;*

repeat until there is no change: {

if there is an FD *U* -} *V* in *F* such that *U* ~ *closllre,*

then set *clo,sure* == *closure* U *v·*

}

Figure 19.4 Computing the Attribute Closure of Attribute Sct *X*

**4)Properties of Decomposition**: The decomposition should not introduce new problems. We should check whether decomposition allows us to recover original relation and whether it allows us to check IC effectively.

4.1) Lossless-Join Decomposition:-

Let R is Relation-schema.

F is set of FD’s that hold over R.

The Decomposition of R into relations with attributes sets R1 & R2 is lossless if and only if F+ contain either FD:

R1∩ R2🡪R1

(or)

R1 ∩ R2 🡪R2

In other words R1 ∩ R2 forms super key for either R1 or R2. The decomposition is loss less decomposition.

The Decomposition which is not loss-less decomposition is called lossy-decomposition.

Example:-1 inst-dept( id,name,salary,dept-name,building,budget)

It is decomposed into

Instructor(id,name,salary,dept\_name)

Dept(detpt\_name,building,budget)

F={dept-name🡪building,dept\_name🡪budget,dept\_name🡪dept\_name}

Check is decomposition is loss-less decomposition?

Solution:

Instructor ∩ dept={dept\_name}

Dept\_name🡪dept.

(or)

Dept\_name is superkey for dept table. Therefore decomposition is loss-less decomposition.

Example:2



Example to lossy-decomposition:-



4.2) Dependency –preserving Decomposition:- Consider a relation R

F is set of FD’s.

R is decomposed or divided into R1 with FD { f1 } and R2 with { f2 }, then

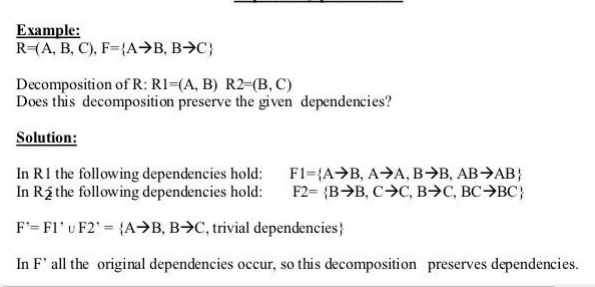
there can be three cases:

**f1 U f2 = F** -----> Decomposition is dependency preserving.

**f1 U f2** is a subset of F -----> Not Dependency preserving.

**f1 U f2** is a super set of F -----> This case is not possible.

Example:



5)NORMALIZATION:- The normalization is DB design technique which organizes tables in manner that reduces redundancy of data & dependency of data. In normalization, the table is divided into smaller tables and links them using Relationship.

Several Normal forms have been proposed. If relation schema is in one of these normal forms. We know that certain kinds of problems can

Not araise. The normal forms based on FD’s are:

If relation schema is in BCNF, then relation schema does not has redudancy,insertion anamoly , updation anamoly and deletion anomaly.

5.1)1NF:- A relation is said to be in first normal form (1NF), if it does not contain any multi-valued attributes, i.e all the values of columns in a relation are atomic.

Ex:- Consider the following Relation (or) Table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **COURSE\_NAME** | **DATE\_COMPLETED** |
| 100 | SMITH | 25000 | C  C++ | 05-OCT-2019  10-NOV-2019 |
| 101 | CLARK | 36000 | JAVA  ORACLE | 11-JAN-2020  15-FEB-2020 |

The above relation is not in first normal form (1NF), because the relation having multiple values of a column. But every column must contains atomic values only. So, we have to convert the above relation into 1NF and is given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **COURSE\_NAME** | **DATE\_COMPLETED** |
| 100 | SMITH | 25000 | C | 05-OCT-2019 |
| 100 | SMITH | 25000 | C++ | 10-NOV-2019 |
| 101 | CLARK | 36000 | JAVA | 11-JAN-2020 |
| 101 | CLARK | 36000 | ORACLE | 15-FEB-2020 |

Hence the above relation is in First Normal Form (1NF).

5.2) 2NF:- A relation is in 2nd NF if

a) Relation is in 1NF and

b) Relation does not contain the partial dependency.

Partial Dependency:- **Partial Dependency** occurs when a non-prime attribute is functionally dependent on part of a candidate key.

Ex:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** | **COURSE\_NAME** | **DATE\_COMPLETED** |

Where primary key/candidate key is {empno,course\_name}.

The functional dependencies are :

FD1 : EMPNO,COURSE\_NAME DATE\_COMPLETED

FD2 : EMPNO ENAME,SAL

The above relation is not in 2nd NF. Because The Relation has partial dependency.

Only primary key should determines the all other non-prime attributes. But part of candidate is functionally determining the non-prime attributes( EMPNO ENAME,SAL).

Decomposition Steps:-

------------------------------

1. Create separate relation for each partial dependency.
2. Remove right hand side attribute of partial dependency from relation that is being decomposed.

EMPLOYEE:

|  |  |  |
| --- | --- | --- |
| **EMPNO** | **ENAME** | **SAL** |

COURSE:

|  |  |  |
| --- | --- | --- |
| **EMPNO** | **COURSE\_NAME** | **DATE\_COMPLETED** |

Hence the above two relations are in 2NF.

5.3)3NF:A relation is said to be in third normal form (3NF), if it satisfies the following properties:

* The relation must be in 2NF
* It does not contain any transitive dependencies.

Transitive Dependencies:- X -> Z is a transitive dependency if the following three functional dependencies hold true:

* X->Y
* Y does not ->X
* Y->Z

Consider the following relation : CUSTOMER

|  |  |  |  |
| --- | --- | --- | --- |
| CUST\_ID | CUST\_NAME | SALESPERSON\_ID | REGION |

The functional dependencies in the above relation are :

FD1 : CUST\_ID CUST\_NAME,SALESPERSON\_ID,REGION

FD2 : SALESPERSON\_ID REGION

In second functional dependency (FD2) there is a transitive dependency.

CUST\_ID SALESPERSON\_ID

SALESPERSON\_ID REGION

CUST\_ID REGION is transitive dependency.

Decompositon Algorithm:-

Step1: X🡪A is FD that causes violation of 3NF. Decompose R into R-A &XA.

Step2: If either R-A (or) XA is not in 3NF, Decompose them further by a recursive application of this algorithm.

SALESPERSON

|  |  |
| --- | --- |
| SALESPERSON\_ID | REGION |

CUSTOMER

|  |  |  |
| --- | --- | --- |
| CUST\_ID | CUST\_NAME | SALESPERSON\_ID |

Hence the above two relations are in third normal form (3NF).

5.4)BCNF:-A relation is said to be in BCNF, if it satisfies the following properties.

* The relation must be in 3NF
* The determinant must be a candidate key.

Candidate key:- An attribute or combination of attributes used for identifying rows or records uniquely in a table or relation is called a candidate key. Consider the following relation:

|  |  |  |  |
| --- | --- | --- | --- |
| STUD\_ID | LECTURER | SUBJECT | AVERAGE |

The functional dependencies are :

STUD\_ID,LECTURER SUBJECT,AVERAGE

SUBJECT LECTURER

The above relation is in 3NF but not in BCNF. So, we can decompose the relation according to decomposition algorithm.

Decomposition Algorithm:

-----------------------------------

Step1: X🡪A is FD that causes violation of BCNF. Decompose R into R-A &XA.

Step2: If either R-A (or) XA is not in BCNF, Decompose them further by a recursive application of this algorithm

**STUDENT** **SUBJECT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STUD\_ID | LECTURER | AVERAGE |  | SUBJECT | LECTURER |

Hence the above two relations are in BCNF.