UNIT-4

SCHEMA REFINEMENT and NORMAL FORMS

Syllabus: Problems caused by redundancy, Decompositions, problem related to decomposition, reasoning about FDS, FIRST, SECOND, THIRD Normal forms, BCNKF, Lossless join Decomposition, Dependency preserving Decomposition, Schema refinement in Data base Design, Multi valued Dependencies, FORTH Normal Form.

Schema Refinement: The purpose of Schema refinement is used for a refinement approach based on decompositions. Redundant storage of information (i.e. duplication of data) is main cause of problem. This redundancy is eliminated by decompose the relation.

Q. Problems caused by redundancy: Redundancy is a method of storing the same information repeatedly. It means, storing the same data more than one place with in a database is can lead several problems. Such as

1) **Redundant Storage:** It removes the multi-valued attribute. It means, some tuples or information is stored repeatedly.

2) **Update Anomalies:** Suppose, if we update one row (or record) then DBMS will update more than one similar row, causes update anomaly.

For example, if we update the department name those who getting the salary 40000 then that will update more than one row of employee table is causes update anomaly.

3) **Insertion Anomalies:** In insertion anomaly, when allows insertion for already existed record again, causes insertion anomaly.

4) **Deletion Anomalies:** In deletion anomaly, when more than one record is deleted instead of specified or one, causes deletion anomaly.

Consider the following example,

Eno	ename	salary	rating	hourly_wages	hours_worked
111	suresh	25000	8	10	40
222	eswar	30000	8	10	30
333	sankar	32000	5	7	30
444	padma	31000	5	7	32
555	aswani	35000	8	10	40

In this example,

1) Redundancy storage: The rating value 8 corresponds to the hourly_wages 10 and there is repeated three times.
2) Update anomalies: If the hourly_wages in the first tuple is updated, it does not make changes in the corresponding second and last tuples. Because, the key element of tuples is emp_id.
   i.e. update employee set hourly_wages = 12 where emp_id = 140; But the question is update hourly_wages from 10 to 12.
3) Insertion anomalies: If we have to inset a new tuple for an employee, we should know both the values *rating* value as well as *hourly_wages* value.
4) Deletion anomalies: Delete all the tupes through a given rating value, causes deletion anomaly.

Q. Decompositions: A relation schema (table) R can be decomposition into a collection of smaller relation schemas (tables) (i.e. R_1, R_2, ... R_m) to eliminate anomalies caused by the redundancy in the original relation R is called Decomposition of relation. This shown in Relational Algebra as

$R_1 \subseteq R$ for $1 \leq i \leq m$ and $R_1 \cup R_2 \cup R_3 \ldots R_m = R$.

(or)

A decomposition of a relation schema R consists of replacing the relation schema by two or more relation schemas that each contains a subset of the attributes of R and together include all attributes in R.

(or)

In simple, words, "The process of breaking larger relations into smaller relations is known as decomposition".

Consider the above example that can be decomposing the following hourly_emps relation into two relations.

Hourly_emp (<u>eno</u>, ename, salary, rating, houly_wages, hours_worked)

This is decomposed into

Hourly_empsd(<u>eno</u>, ename, salary, rating, hours_worked) **and**

Wages(<u>rating</u>, houly_wages)

Eno	ename	salary	rating	hours_worked
111	suresh	25000	8	40
222	eswar	30000	8	30
333	sankar	32000	5	30
444	padma	31000	5	32
555	aswani	35000	8	40

rating	hourly_wages
8	10
5	7

<u>Q. Problems Related to Decomposition:</u>

The use of Decomposition is to split the relation into smaller parts to eliminate the anomalies. The question is

1. What are problems that can be caused by using decomposition?
2. When do we have to decompose a relation?

The answer for the **first** question is, two properties of decomposition are considered.

1) **<u>Lossless-join Property:</u>** This property helps to identify any instance of the original relation from the corresponding instance of the smaller relation attained after decomposition.
2) **<u>Dependency Preservation</u>**: This property helps to enforce constraint on smaller relation in order to enforce the constraints on the original relation.

The answer for the **second** question is, number of normal forms exists. Every relation schema is in one of these normal forms and these normal forms can help to decide whether to decompose the relation schema further or not.

The **Disadvantage** of decomposition is that it enforces us to join the decomposed relations in order to solve the queries of the original relation.

Q. What is Relation?: A relation is a named two–dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows.

For example, a relation named Employee contains following attributes, emp-id, ename, dept name and salary.

Emp-id	ename	dept name	salary
100	Simpson	Marketing	48000
140	smith	Accounting	52000
110	Lucero	Info-systems	43000
190	Davis	Finance	55000
150	martin	marketing	42000

marketing 42000

What are the Properties of relations?:

The properties of relations are defined on two dimensional tables. They are:

- Each relation (or table) in a database has a unique name.
- An entry at the intersection of each row and column is atomic or single. These can be no multiplied atttributes in a relation.
- Each row is unique, no two rows in a relation are identical.
- Each attribute or column within a table has a unique name.
- The sequence of columns (left to right) is insignificant the column of a relation can be interchanged without changing the meaning use of the relation.
- The sequence of rows (top to bottom) is insignificant. As with column, the rows of relation may be interchanged or stored in any sequence.

Q. Removing multi valued attributes from tables: the "second property of relations" in the above is applied to this table. In this property, there is no multi valied attributes in a relation. This rule is applied to the table or relation to eliminate the one or more multi valued attribute. Consider the following the example, the employee table contain 6 records. In this the course title has multi valued values/attributes. The employee 100 taken two courses vc++ and ms-office. The record 150 did not taken any course. So it is null.

Emp-id	name	dept-name	salary	course_title
100	Krishna	cse	20000	vc++, msoffice
140	Rajasekhar	cse	18000	C++, DBMS,DS

Now, this multi valued attributes are eliminated and shown in the following employee2 table.

Emp-id	name	dept-name	salary	course_title
100	Krishna	cs	20000	vc++
100	Krishna	cs	20000	MSoffice
140	Rajasekhar	cs	18000	C++
140	Rajasekhar	cs	18000	DBMS
140	Rajasekhar	cs	18000	DS

Functional dependencies : A functional dependency is a constraint between two attributes (or) two sets of attributes.

For example, the table EMPLOYEE has 4 columns that are Functionally dependencies on EMP_ID.

Emp_id	Ename	Dept_name	Salry

Partial functional dependency : It is a functional dependency in which one or more non-key attributes are functionally dependent on part of the primary key. Consider the following graphical representation, in that some of the attributes are partially depend on primary key.

Emp_id	Course_title	Ename	Dept_name	Salry	date completed

In this example, Ename, Dept_name, and salary are fully functionally depend on Primary key of Emp_id. But Course_title and date_completed are partial functional dependency. In this case, the partial functional dependency creates redundancy in that relation.

Q. What is Normal Form? What are steps in Normal Form?

NORMALIZATION: Normalization is the process of decomposing relations to produce smaller, well-structured relation.

To produce smaller and well structured relations, the user needs to follow six normal forms.

Steps in Normalization:

A *normal form* is state of relation that result from applying simple rules from regarding functional dependencies (relationships between attributes to that relation. The normal form are

1. First normal form
2. Second normal form
3. Third normal form
4. Boyce/codd normal form
5. Fourth normal form
6. Fifth normal form

1) **First Normal Form:** Any multi-valued attributes (also called repeating groups) have been removed,
2) **Second Normal Form:** Any partial functional dependencies have been removed.
3) **Third Normal Form:** Any transitive dependencies have been removed.
4) **Boyce/Codd Normal Form:** Any remaining anomalies that result from functional dependencies have been removed.
5) **Fourth Normal Form:** Any multi-valued dependencies have been removed.
6) **Fifth Normal Form:** Any remaining anomalies have been removed.

Advantages of Normalized Relations Over the Un-normalized Relations: The advantages of normalized relations over un-normalized relations are,

1) Normalized relation (table) does not contain repeating groups whereas, un-normalized relation (table) contains one or more repeating groups.
2) Normalized relation consists of a primary key. There is no primary key presents in un-normalized relation.
3) Normalization removes the repeating group which occurs many times in a table.
4) With the help of normalization process, we can transform un-normalized table to First Normal Form (1NF) by removing repeating groups from un-normalized tables.
5) Normalized relations (tables) gives the more simplified result whereas un-normalized relation gives more complicated results.
6) Normalized relations improve storage efficiency, data integrity and scalability. But un-normalized relations cannot improvise the storage efficiency and data integrity.
7) Normalization results in database consistency, flexible data accesses.

<u>Q. FIRST NORMAL FORM (1NF)</u>: A relation is in first normal form (1NF) contains no multi-Valued attributes. Consider the example employee, that contain multi valued attributes that are removing and converting into single valued attributes

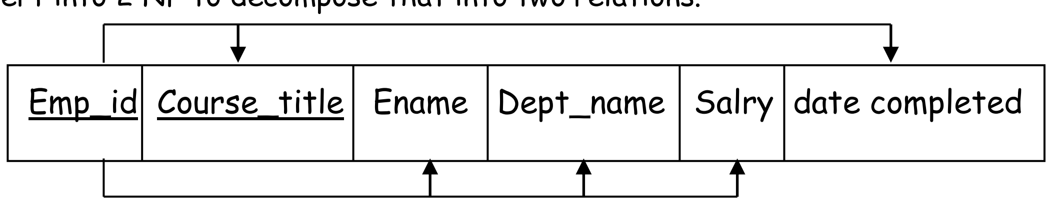
<u>Multi valued attributes in course title</u>

Emp-id	name	dept-name	salary	course_title
100	Krishna	cse	20000	vc++,
				msoffice
140	Raja	it	18000	C++,
				DBMS,
				DS

Removing the multi valued attributes and converting single valied using First NF

Emp-id	name	dept-name	salary	course_title
100	Krishna	cse	20000	vc++
100	Krishna	cse	20000	msoffice
140	Raja	it	18000	C++
140	Raja	it	18000	DBMS
140	Raja	it	18000	DS

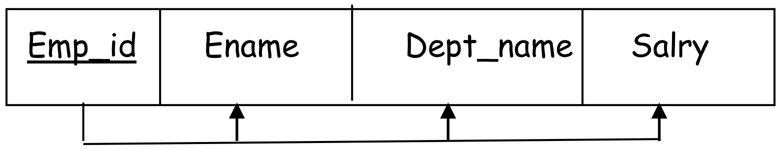
<u>SECOND NORMAL FORM(2NF)</u>: A relation in Second Normal Form (2NF) if it is in the 1NF and if all non-key attributes are fully functionally dependent on the primary key. In a functional dependency X → Y, the attribute on left hand side (i.e. x) is the primary key of the relation and right side attributes on right hand side i.e. Y is the non-key attributes. In some situation some non key attributes are partial functional dependency on primary key. Consider the following example for partial functional specification and also that convert into 2 NF to decompose that into two relations.



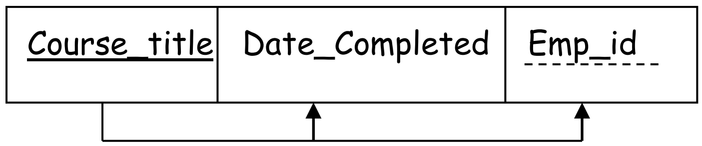
In this example, Ename, Dept_name, and salary are fully functionally depend on Primary key of Emp_id. But Course_title and date_completed are partial functional dependency. In this case, the partial functional dependency creates redundancy in that relation.

➢ To avoid this, convert this into Second Normal Form. The 2NF will decompose the relation into two relations, shown in graphical representation.

EMPLOYEE



COURSE

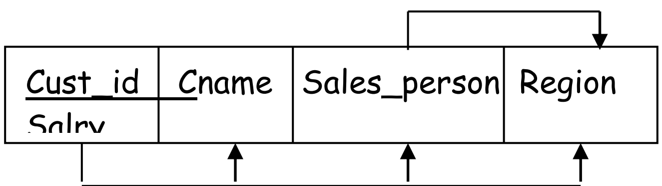


In the above graphical representation

➢ the EMPLOYEE relation satisfies rule of 1 NF in Second Normal form and
➢ the COURSE relation satisfies rule of 2 NF by decomposing into two relation.

<u>THIRD NORMAL FORM(3NF)</u>: A relation that is in Second Normal form and has no transitive dependencies present.

<u>Transitive dependency :</u> A transitive is a functional dependency between two non key attributes. For example, consider the relation Sales with attributes cust_id, name, sales person and region that shown in graphical representation.



CUST_ID	NAME	SALESPERSON	REGION
1001	Anand	Smith	South
1002	Sunil	kiran	West
1003	Govind	babu rao	East
1004	Manohar	Smith	South
1005	Madhu	Somu	North

In this example, to insert, delete and update any row that facing Anomaly.

a) <u>Insertion Anomaly</u>: A new salesperson is assigned to North Region without assign a customer to that salesperson. This causes insertion Anomaly.
b) <u>Deletion Anomaly</u>: If a customer number say 1003 is deleted from the table, we lose the information of salesperson who is assigned to that customer. This causes, Deletion Anomaly.
c) <u>Modification Anomaly</u>: If salesperson Smith is reassigned to the East region, several rows must be changed to reflect that fact. This causes, update anomaly.

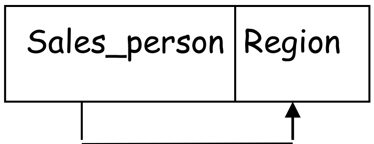
To avoid this Anomaly problem, the transitive dependency can be removed by decomposition of SALES into two relations in **<u>3NF</u>**.

Consider the following example, that removes Anomaly by decomposing into two relations.

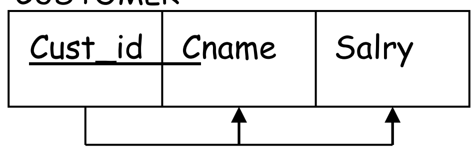
CUST_ID	NAME	SALESPERSON
1001	Anand	Smith
1002	Sunil	kiran
1003	Govind	babu rao
1004	Manohar	Smith
1005	Madhu	Somu

SALESPERSON	REGION
Smith	South
kiran	West
babu rao	East
Smith	South
Somu	North

SALES PERSON



CUSTOMER

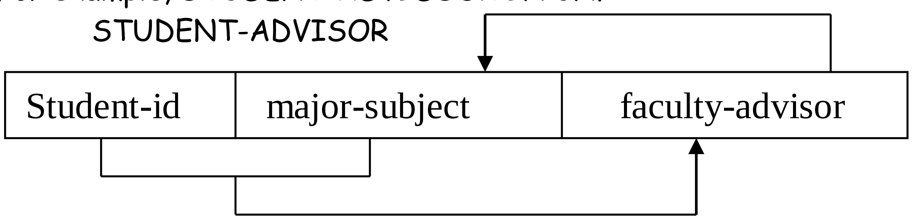


Q. BOYCE/CODD NORMAL FORM(BCNF): A relation is in BCNF if it is in 3NF and every determinant is a candidate key.

FD in F^+ of the form $X \rightarrow A$ where $X \subseteq S$ and $A \in S$, X is a super key of R.

Boyce-Codd normal form removes the remaining anomalies in 3NF that are resulting from functional dependency, we can get the result of relation in BCNF.
For example, STUDENT-ADVIDSOR IN 3NF

STUDENT-ADVISOR



STUDENT-ADVISOR relstion with simple data.

STYDENT-ID	MAJOR-SUBJECT	FACULTY-ADVISOR
1	MATHS	B
2	MATHS	B
3	MATHS	B
4	STATISTICS	A
5	STATISTICS	A

In the above relation the primary key in student-id and major-subject. Here the part of the primary key major-subject is dependent upon a non key attribute faculty–advisor. So, here the determenant the faculty-advisor. But it is not candidate key.

Here in this example there are no partial dependencies and transitive dependencies. There is only functional dependency between part of the primary key and non key attribute. Because of this dependency there is anomely in this relation.

Suppose that in maths subject the advisor' B' is replaced by X. this change must be made in two or more rows in this relation. This is an updation anomely.

To convert a relation to BCNF the first step in the original relation is modified that the determinant(non key attributes) becoms a component of the primary key of new relation. The attribute that is dependent on determinant becomes a non key attributes .

STUDENT-ADVISOR

STUDENT-ID	FACULTY-ADVIDSOR	MAJOR-SUBJEST

The second step in the conversion process is decompose the relation to eleminate the partial functional dependency. This results in two relations. these relations are in 3NF and BCNF . since there is only one candidate key. That is determinant.

Two relations are in BCNF.

ADVISOR

Faculty- advisor	major-subject

STUDENT

Student-id	faculty-advisor

In these two relations the student relation has a composite key , which contains attributes student-id and faculty-advisor. Here faculty–advisor a foreign key which is referenced to the primary key of the advisor relation.

Two relations are in BCNF with simple data.

Faculty Advisor	Major_subject
B	MATHS
A	PHYSICS

Student_id	Faculty_Advisor
1	B
2	B
3	A
4	A
5	A

Q.Fourth Normal Form (4 NF) : A relation is in BCNF that contain no multi-valued dependency. In this case, **1 NF** will repeated in this step. For example, R be a relation schema, X and Y be attributes of R, and F be a set of dependencies that includes both FDs and MVDs. (i.e. Functional Dependency and Multi-valued Dependencies). Then R is said to be in Fourth Normal Form (4NF) if for every MVD $X \rightarrow\rightarrow Y$ that holds over R, one of the following statements is true.

1) $Y \subseteq X$ or $XY = R$, or 2) X is a super key.

Example: Consider a relation schema ABCD and suppose that are FD $A \rightarrow BCD$ and the MVD $B \rightarrow \rightarrow C$ are given as shown in Table

B	C	A	D	tuples
b	c_1	a_1	d_1	- tuple t_1
b	c_2	a_2	d_2	- tuple t_2
b	c_1	a_2	d_2	- tuple t_3

It shows three tuples from relation ABCD that satisfies the given MVD $B \rightarrow \rightarrow C$. From the definition of a MVD, given tuples t_1 and t_2, it follows that tuples t_3 must also be included in the above relation. Now, consider tuples t_2 and t_3. From the given FD $A \rightarrow BCD$ and the fact that these tuples have the same A-value, we can compute the $c_1 = c_2$. Therefore, we see that the FD B → C must hold over ABCD whenever the FD A → BCD and the MVD B → → C holds. If B → C holds, the relation is not in BCNF but the relation is in 4 NF.

The fourth normal from is useful because it overcomes the problems of the various approaches in which it represents the multi-valued attributes in a single relation.

Q. Fifth Normal Form (5 NF)

: Any remaining anomalies from 4 NF relation have been removed.

A relation schema R is said to be in Fifth Normal Form (5NF) if, for every join dependency * ($R1, \ldots R_n$) that holds over R, one of the following statements is true.

*R_i = R for some I, or

* The JD is implied by the set of those FDs over R in which the left side is a key for R. It deals with a property loss less joins.

Q. LOSSELESS-JOIN DECOMPOSITION:

Let R be a relation schema and let F be a set FDs (Functional Dependencies) over R. A decomposition of R into two schemas with attribute X and Y is said to be lossless-join decomposition with respect to F, if for every instance r of R that satisfies the dependencies in F_r.

$$\pi_x(r) \bowtie \pi_y(r) = r$$

In simple words, we can recover the original relation from the decomposed relations.

In general, if we take projection of a relation and recombine them using natural join, we obtain some additional tuples that were not in the original relation.

S	P	D
s_1	p_1	d_1
s_2	p_2	d_2
s_2	p_1	d_3

S	P
s_1	p_1
s_2	p_2
s_2	p_1

P	D
p1	d_1
p_2	d_2
p1	d_3

S	P	D
s_1	p_1	d_1
s_1	p_1	d_3
s_2	p_2	d_2
s_2	p_1	d_1
s2	p_1	d_3

The decomposition of relation schema r i.e. SPD into SP i.e. PROJECTING π_{sp} (r) and PD i.e., projecting π_{PD} (r) is therefore lossless decomposition as it gains back all original tuples of relation 'r' as well as with some additional tuples that were not in original relation 'r'.

Q. Dependency-Preserving Decomposition:

Dependency-preserving decomposition property allows us to check integrity constraints efficiently. In simple words, a dependency-preserving decomposition allows us to enforce all FDs by examining a single relation on each insertion or modification of a tuple.

Let R be a relation schema that is decomposed in two schemas with attributes X and Y and let F be a set of FDs over R. The projection of F on X is the set of FD s in the closure F^+ that involves only attributes in X. We denote the projection of F on attributes X as F_X. Note that a dependency $U \rightarrow V$ in F^+ is in F_x only if all the attributes in U and Y are in X. The decomposition of relation schema R with FDs F into schemas with attributes X and Y is dependency-preserving if $(F_x \cup F_y)^+ = F^+$.

That is, if we take the dependencies in F_x and F_y and compute the closure of their union, then we get back all dependencies in the closure of F. To enforce F_x. we need to examine only relation X (that inserts on that relation) to enforce F_y, we need to examine only relation Y.

Example: Suppose that a relation R with attributes ABC is decomposed into relations with attributes AB and BC. The set F of FDs over r includes $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow A$. Here, $A \rightarrow B$ is in F_{AB} and $B \rightarrow C$ is in FBC and both are dependency-preserving. Where as $C \rightarrow A$ is not implied by the dependencies of F_{AB} and F_{BC}. Therefore $C \rightarrow A$ is not dependency-preserving.

Consequently, F_{AB} also contains $B \rightarrow A$ as well as $A \rightarrow B$ and F_{BC} contains $C \rightarrow B$ as well as $B \rightarrow C$. Therefore $F_{AB} \cup F_{BC}$ contain $A \rightarrow B$, $B \rightarrow C$, $B \rightarrow A$ and $C \rightarrow B$.

Now, the closure of the dependencies in F_{AB} and F_{BC} includes $C \rightarrow A$ (because, from $C \rightarrow B$, $B \rightarrow A$ and transitivity rule, we compute as $C \rightarrow A$).