Lecture Notes on

4AID4-05

Database Management System

Unit 4

Department of Artificial Intelligence & Data Science
Jaipur Engineering College & Research Centre, Jaipur

Neelkamal Chaudhary

Assistant Professor

AI&DS

Vision of the Institute

To become a renowned centre of outcome based learning and work toward academic, professional, cultural and social enrichment of the lives of individuals and communities.

Mission of the Institute

M1: Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.

M2: Identify, based on informed perception of Indian, regional and global needs, the areas of focus and provide platform to gain knowledge and solutions.

M3: Offer opportunities for interaction between academia and industry.

M4: Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders can emerge in a range of professions.

Vision Of The Department

To prepare students in the field of Artificial Intelligence and Data Science for competing with the global perspective through outcome based education, research and innovation.

Mission Of The Department

1. To impart outcome based education in the area of AI&DS.
2. To provide platform to the experts from institutions and industry of repute to transfer the knowledge to students for providing competitive and sustainable solutions.
3. To provide platform for innovation and research.

Program Outcomes (PO)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and Artificial Intelligence & Data Science specialization to the solution of complex Artificial Intelligence & Data Science problems.

2. **Problem analysis**: Identify, formulate, research literature, and analyze complex Artificial Intelligence & Data Science problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions**: Design solutions for complex Artificial Intelligence & Data Science problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of Artificial Intelligence & Data Science experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex Artificial Intelligence & Data Science activities with an understanding of the limitations.

6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional Artificial Intelligence & Data Science practice.

7. **Environment and sustainability**: Understand the impact of the professional Artificial Intelligence & Data Science in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the Artificial Intelligence & Data Science practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in Artificial Intelligence & Data Science

10. **Communication**: Communicate effectively on complex Artificial Intelligence & Data Science activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance**: Demonstrate knowledge and understanding of the Artificial Intelligence & Data Science and management principles and apply these to one"s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change in Artificial Intelligence & Data Science.

Program Educational Objectives (PEO)

PEO1: To provide students with the fundamentals of Engineering Sciences with more emphasis in Artificial Intelligence & Data Science by way of analyzing and exploiting engineering challenges.

PEO2: To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real life problems in Artificial Intelligence & Data Science

PEO3: To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues for Artificial Intelligence & Data Science.

PEO4: To provide students with an academic environment aware of excellence, leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career in Artificial Intelligence & Data Science.

PEO5: To prepare students to excel in Industry and Higher education by Educating Students along with High moral values and Knowledge in Artificial Intelligence & Data Science.

COURSE OUTCOME: After studying this subject, student will be able

CO-1	Design an ER model for an enterprise
CO-2	Perform and analysis Query database using Relational Algebra, Relational Calculus and SQL
CO-3	Apply normalization based on functional dependency.
C0-4	Illustrate for serialzability among concurrent transactions and apply concurrency control protocols, and Outline database recovery techniques

CO_PO Mapping

SUBJECT CODE	subject name		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
4AID4-05	Database Management System	CO-1	3	3	3	3	3	2	1	2	1	2	2	2
		CO-2	3	3	3	3	3	2	1	1	1	2	2	2
		CO-3	3	3	3	3	3	2	1	1	1	2	2	2
		CO-4	3	3	3	2	2	2	1	1	1	2	2	2

4AID4-05: Database Management System

Credit: 3 **Max. Marks: 100(IA:30, ETE:70)**

3L+0T+0P **End Term Exam: 3 Hours**

SN	Contents	Hours
1	**Introduction:** Objective, scope and outcome of the course.	1
2	**Introduction to database systems:** Overview and History of DBMS. File System v/s DBMS.Advantage of DBMS Describing and Storing Data in a DBMS.Queries in DBMS.Structure of a DBMS. **Entity Relationship model:** Overview of Data Design Entities, Attributes and Entity Sets, Relationship and Relationship Sets. Features of the ER Model- Key Constraints, Participation Constraints, Weak Entities, Class Hierarchies, Aggregation, Conceptual Data Base, and Design with ER Model- Entity v/s Attribute, Entity vs Relationship Binary vs Ternary Relationship and Aggregation v/s ternary Relationship Conceptual Design for a Large Enterprise.	7
3	**Relationship Algebra and Calculus:** Relationship Algebra Selection and Projection, Set Operations, Renaming, Joints, Division, Relation Calculus, Expressive Power of Algebra and Calculus. **SQL queries programming and Triggers:** The Forms of a Basic SQL Query, Union, and Intersection and Except, Nested Queries, Correlated Nested Queries, Set-Comparison Operations, Aggregate Operators, Null Values and Embedded SQL, Dynamic SQL, ODBC and JDBC, Triggers and Active Databases.	8
4	**Schema refinement and Normal forms:** Introductions to Schema Refinement, Functional Dependencies, Boyce-Codd Normal Forms, Third Normal Form, Normalization-Decomposition into BCNF Decomposition into 3-NF.	8
5	**Transaction Processing:** Introduction-Transaction State, Transaction properties, Concurrent Executions. Need of Serializability, Conflict vs. View Serializability, Testing for Serializability, Recoverable Schedules, Cascadeless Schedules.	8
6	**Concurrency Control: Implementation of Concurrency:** Lock-based protocols, Timestamp-based protocols, Validation-based protocols, Deadlock handling, **Database Failure and Recovery:** Database Failures, Recovery Schemes: Shadow Paging and Log-based Recovery, Recovery with Concurrent transactions.	8
	Total	**40**

UNIT-4

SCHEMA REFINEMENT AND NORMALISATION

Unit 3 contents at a glance:

1. Introduction to schema refinement,
2. functional dependencies,
3. reasoning about FDs.
4. Normal forms: 1NF, 2NF, 3NF, BCNF,
5. properties of decompositions,
6. normalization,
7. schema refinement in database design(Refer Text Book),
8. other kinds of dependencies: 4NF, 5NF, DKNF
9. Case Studies(Refer text book)

.

1. Schema Refinement:

The Schema Refinement refers to refine the schema by using some technique. The best technique of schema refinement is **decomposition.**

Normalisation or Schema Refinement is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

Redundancy refers to repetition of same data or duplicate copies of same data stored in different locations.

Anomalies: Anomalies refers to the problems occurred after poorly planned and normalised databases where all the data is stored in one table which is sometimes called a flat file database.

Anomalies or problems facing without normalization(problems due to redundancy) :

Anomalies refers to the problems occurred after poorly planned and unnormalised databases where all the data is stored in one table which is sometimes called a flat file database. Let us consider such type of schema –

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k
Primary Key(SID,CID)				

Here all the data is stored in a single table which causes redundancy of data or say anomalies as SID and Sname are repeated once for same CID . Let us discuss anomalies one by one.

Due to redundancy of data we may get the following problems, those are-

1. insertion anomalies : It may not be possible to store some information unless some other information is stored as well.

2. redundant storage: some information is stored repeatedly

3. update anomalies: If one copy of redundant data is updated, then inconsistency is createdunless all redundant copies of data are updated.

4. deletion anomalies: It may not be possible to delete some information without losing someother information as well.

Problem in updation / updation anomaly – If there is updation in the fee from 5000 to 7000, then we have to update FEE column in all the rows, else data will become inconsistent.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	~~5k~~ 7k
S2	A	C1	C	~~5k~~ 7k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

Insertion Anomaly and Deletion Anomaly- These anomalies exist only due to redundancy, otherwise they do not exist.

InsertionAnomalies: New course is introduced C4, But no student is there who is having C4subject.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

NULL	NULL	CA	DB	12k

To Insert that Row, It is Required to Put Dummy Data..

Therefore,

xx	xx	CA	DB	12k

Because of insertion of some data, It is forced to insert some other dummy data.

Deletion Anomaly :

Deletion of S3 student cause the deletion of course.

Because of deletion of some data forced to delete some other useful data.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
~~S3~~	~~B~~	~~C2~~	~~C~~	~~10k~~
~~S3~~	~~B~~	~~C2~~	~~JAVA~~	~~15k~~

Solutions To Anomalies : Decomposition of Tables – Schema Refinement

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C2	JAVA	15k

SID	Sname	CID
S1	A	C1
S2	A	C1
S1	A	C2
~~S3~~	~~B~~	~~C2~~
~~S3~~	~~B~~	~~C3~~

PK(SID,CID)

Deletion Anamoly Removed

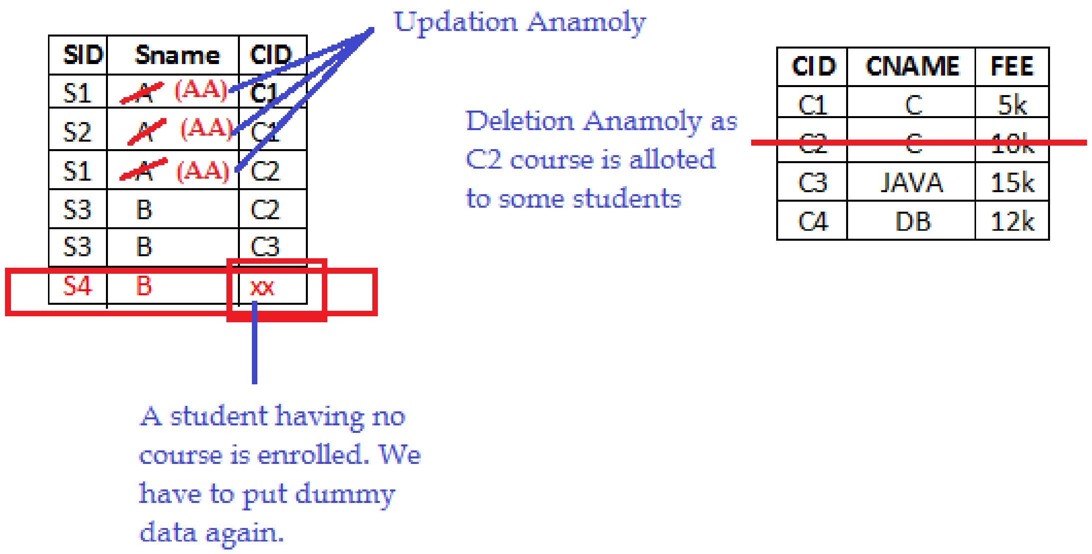
CID	CNAME	FEE
C1	C	~~5k~~ 7k
C2	C	10k
C3	JAVA	15k
C4	DB	12k

(Updation Anamoly Removed)

Insertion Anamoly Removed

PK(CID)

There are some Anomalies in this again –



What is the Solution ??
Solution : decomposing into relations as shown below

R1

SID	Sname

R2

SID	CID

R3

CID	Cname	Fee

□**TO AVOID REDUNDANCY** and problems due to redundancy, we use refinement techniquecalled **DECOMPOSITION.**

Decomposition:- Process of decomposing a larger relation into smaller relations.

□Each of smaller relations contain subset of attributes of original relation.

Functional dependencies:

□Functional dependency is a relationship that exist when one attribute uniquely determinesanother attribute.

□Functional dependency is a form of integrity constraint that can identify schema withredundant storage problems and to suggest refinement.

□A functional dependency A□B in a relation holds true if two tuples having the same value ofattribute A also have the same value of attribute B

IF t1.X=t2.X then t1.Y=t2.Y where t1,t2 are tuples and X,Y are attributes.

Reasoning about functional dependencies:

Armstrong Axioms :

Armstrong axioms defines the set of rules for reasoning about functional dependencies andalso to infer all the functional dependencies on a relational database.

Various axioms rules or inference rules:

Primary axioms:

Rule 1	**Reflexivity** If A is a set of attributes and B is a subset of A, then A holds B. { A → B }
Rule 2	**Augmentation** If A hold B and C is a set of attributes, then AC holds BC. {AC → BC} It means that attribute in dependencies does not change the basic dependencies.
Rule 3	**Transitivity** If A holds B and B holds C, then A holds C. If {A → B} and {B → C}, then {A → C} A holds B {A → B} means that A functionally determines B.

secondary or derived axioms:

Rule 1	**Union** If A holds B and A holds C, then A holds BC. If{A → B} and {A → C}, then {A → BC}
Rule 2	**Decomposition** If A holds BC and A holds B, then A holds C. If{A → BC} and {A → B}, then {A → C}
Rule 3	**Pseudo Transitivity** If A holds B and BC holds D, then AC holds D. If{A → B} and {BC → D}, then {AC → D}

Attribute closure: Attribute closure of an attribute set can be defined as set of attributes whichcan be functionally determined from it.

NOTE:

To find attribute closure of an attribute set- 1)add

elements of attribute set to the result set.

2)recursively add elements to the result set which can be functionally determined from theelements of result set.

Algorithm : Determining X^+, the closure of X under F.

```
Input : Let F be a set of FDs for relation R.

Steps:
            1. X⁺ = X                              //initialize X⁺ to X
            2. For each FD : Y -> Z in F Do
               If Y ⊆ X⁺ Then                      //If Y is contained in X⁺
               X⁺  =   X⁺  ∪  Z                    //add Z to X⁺
               End If
               End For
            3. Return  X⁺                        //Return closure of X

Output : Closure  X⁺ of X under F
```

Types of functional dependencies:

1)**Trivial functional dependency:-**If X□Y is a functional dependency where Y subset X, thesetype of FD's called as trivial functional dependency.

2)**Non-trivial functional dependency:-**If X□Y and Y is not subset of X then it is called non-trivial functional dependency.

3)**Completely non-trivial functional dependency:-**If X□Y and X∩Y=Φ(null) then it is called completely non-trivial functional dependency.

Prime and non-prime attributes

Attributes which are parts of any candidate key of relation are called as prime attribute, othersare non-prime attributes.

Candidate Key:
Candidate Key is minimal set of attributes of a relation which can be used to identify a tupleuniquely.
Consider student table: student(<u>sno</u>, sname,sphone,age)
we can take **sno** as candidate key. we can have more than 1 candidate key in a table.types of candidate keys:

1. simple(having only one attribute)
2. composite(having multiple attributes as candidate key)

Super Key:
Super Key is set of attributes of a relation which can be used to identify a tuple uniquely.

- Adding zero or more attributes to candidate key generates super key.
- A candidate key is a super key but vice versa is not true.

Consider student table: student(<u>sno</u>, sname,sphone,age)we can take sno, (sno, sname) as super key

Finding candidate keys problems:

Example 1: Find candidate keys for the relation R(ABCD) having following FD's

AB→CD, C→A, D→A

Solution:

AB^+ = {ABCD} A and B are prime attributes

C→A replace A by c

BC^+ = {ABCD} A and C are prime attributes (A^+ = A^+ = {AC})

D→B replace B by D

AD^+ = {ABCD} A and D are prime attributes (D^+ = {BD})

CD^+ = {ABCD} (replacing A by C in AD)

AB, BC, CD, AD are candidate keys.

Example 2: Find candidate keys for R(ABCDE) having following FD's

A→BC,CD→E,B→D, E→A

Solution:

A^+ = {ABCDE} A is candidate key and prime attribute

E→A so replace A by E

E^+ = {ABCDE} E is candidate key and prime attribute

CD→E replace E by CD

CD^+ = {ABCDE} (C^+ = C and D^+ = D) no proper subset of CD is superkey. so CD is candidate key

B→D

BC^+ = {ABCDE} (B^+ = BD) BC is candidate key

A, E, CD, BC are candidate keys

Question 1 : Given a relation R(ABCDEF) having FDs {AB→C, C→D, D→E , F→B, E→F} Identify the prime attributes and non prime attributes .

Solution :

$(AB)^+$: {ABCDEF} ⇒ Super Key
$(A)^+$: {A} ⇒ Not Super Key
$(B)^+$: {B} ⇒ Not Super Key
Prime Attributes : {A,B}
(AB) → Candidate Key
↓ (as F → B)
$(AF)^+$: {AFBCDE}
$(A)^+$: {A} ⇒ Not Super key
$(F)^+$: {FB} ⇒ Not Super Key
(AF) → Candidate Key
↓
$(AE)^+$: {AEFBCD}
$(A)^+$: {A} ⇒ Not Super key
$(E)^+$: {EFB} ⇒ Not Super key
(AE) → Candidate Key
↓
$(AD)^+$: {ADEFBC}
$(A)^+$: {A} ⇒ Not Super key
$(D)^+$: {DEFB} ⇒ Not Super key
(AD) → Candidate Key
↓
$(AC)^+$: {ACDEFB}
$(A)^+$: {A} ⇒ Not Super Key
$(C)^+$: {DCEFB} ⇒ Not Super Key
⇒ Candidate Keys {AB, AF, AE, AD, AC}
⇒ Prime Attributes {A,B,C,D,E,F}
⇒ Non Prime Attributes {}

Question 2: Given a relation R(ABCDEF) having FDs {AB → C, C → DE , E → F, C → B} Identify the prime attributes and non prime attributes.

Solution :

$(AB)^+$: {A B C D E F}
$(A)^+$: {A}
$(B)^+$: {B}
(AB) ⇒ (AC), $(AC)^+$: {ABCDEF}
$(C)^+$: {DECBF}
⇒ Candidate Keys {AB, AC}
⇒ Prime Attributes {A,B,C}
⇒ Non Prime Attributes {D,E,F}

Normalization:

Normalization is a process of designing a consistent database with minimum redundancywhich support data integrity by grating or decomposing given relation into smaller relations preserving constraints on the relation.

Normalisation removes data redundancy and it will helps in designing a good data base whichinvolves a set of normal forms as follows -

1)First normal form(1NF)

2)Second normal form(2NF)

3)Third normal form(3NF)

4)Boyce coded normal form(BCNF)

5)Forth normal form(4NF)

6)Fifth normal form(5NF)

7)Sixth normal form(6NF)

8)Domain key normal form.

Normal Forms

1st Normal Form	No repeating data groups
2nd Normal Form	No partial key dependency
3rd Normal Form	No transitive dependency
Boyce-Codd Normal Form	Reduce keys dependency
4th Normal Form	No multi-valued dependency
5th Normal Form	No join dependency

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's	Most	Yes	Yes
Eliminates redundancy due to MVD's	No	No	Yes
Preserves FD's	Yes	Maybe	Maybe
Preserves MVD's	Maybe	Maybe	Maybe

Properties of normal forms and their decompositions

1)First normal form: A relation is said to be in first normal form if it contains all atomic values orsingle values.

Example:

Domain	Courses
Programming	C , java
Web designing	HTML , PHP

The above table consist of multiple values in single columns which can be reduced intoatomic values by using first normal form as follows-

Domain	Courses
Programming	C
Programming	Java
Web designing	HTML
Web designing	PHP

2)Second normal form: A relation is said to be in second normal form if it is in first normal formwithout any partial dependencies.

In second normal form non-prime attributes should not depend on proper subset of keyattributes.

Example:

Student id	Student name	Project Id	Project name

Here (student id, project id) are key attributes and (student name, project name) are non-primeattributes. It is decomposed as-

Student id	Student name	Project id

Project id	Project name

3)Third normal form: A relation is said to be in third normal form , if it is already in second normal form and no transitive dependencies exists.

Transitive dependency – If A->B and B->C are two FDs then A->C is called transitive dependency.

A relation is in 3NF if at least one of the following condition holds in every non-trivial functiondependency X –> Y

1. X is a super key.
2. Y is a prime attribute (each element of Y is part of some candidate key).

Student id	Student name	City	country	ZIP

It is decomposed as:

Student id	Student name	ZIP

ZIP	city	country

4)Boyce normal form: It is an extension of third normal form where in a functionaldependency X A , X must be a super key.

A relation is in BCNF if in every non-trivial functional dependency X –> Y, X is a super key.

5)fourth normal form: A relation is said to be in fourth normal form if it is in third normal formand no multi value dependencies should exist between attributes.

Note: In some cases multi value dependencies may exist not more than one time in a givenrelation.

6) fifth normal form: fifth normal form is related to join dependencies.

 A relation R is said to be in fifth normal form if for every join dependency JD join {R_1 , R_2 ,…….R_N } that holds over relation R one of the following statements must be true-1)R_i =R

for some i

2)the join dependency is implied by the set of those functional dependency over relation R inwhich the left side is key attribute for R.

NOTE: if the relation schema is a third normal form and each of its keys consist of singleattribute, we can say that it can also be in fifth normal form.

 A join dependency JD join {R_1, R_2, ……R_N} is said to hold for a relation R if R_1,R_2…..R_N this decomposition is a loss less join decomposition of R.

 When a relation is in forth normal form and decompose further to eliminate redundancy and anomalies due to insert or update or delete operation, there should not be any loss of data or should not create a new record when the decompose tables are rejoin.

7) Domain key normal form: A domain key normal form keeps a constraint that every constraint on the relation is a logical sequence of definition of keys and domains.

8) Sixth normal form: A relation is said to be in sixth normal form such that the relation R should not contain any non-trivial join dependencies.

 Also sixth normal form considers temporal dimensions(time) to the relational model.

Key Points related to normal forms –

1. BCNF is free from redundancy.
2. If a relation is in BCNF, then 3NF is also also satisfied.
3. If all attributes of relation are prime attribute, then the relation is always in 3NF.
4. A relation in a Relational Database is always and at least in 1NF form.
5. Every Binary Relation (a Relation with only 2 attributes) is always in BCNF.
6. If a Relation has only singleton candidate keys(i.e. every candidate key consists of only 1 attribute), then the Relation is always in 2NF(because no Partial functional dependency possible).
7. Sometimes going for BCNF form may not preserve functional dependency. In that case go for BCNF only if the lost FD(s) is not required, else normalize till 3NF only.
8. There are many more Normal forms that exist after BCNF, like 4NF and more. But in real world database systems it’s generally not required to go beyond BCNF.

problems on normal forms:

Problem 1:

Find the highest normal form in R (A, B, C, D, E) under following functional dependencies. ABC -->
D
CD --> AE

Solution:

Important Points for solving above type of question.

1) It is always a good idea to start checking from BCNF, then 3 NF and so on.

2) If any functional dependency satisfied a normal form then there is no need to check for lowernormal form. For example, ABC –> D is in BCNF (Note that ABC is a super key), so no need to check this dependency for lower normal forms.

Candidate keys in given relation are {ABC, BCD}

BCNF: ABC -> D is in BCNF. Let us check CD -> AE, CD is not a super key so this dependency is notin BCNF. So, R is not in BCNF.

3NF: ABC -> D we don't need to check for this dependency as it already satisfied BCNF. Let usconsider CD -> AE. Since E is not a prime attribute, so relation is not in 3NF.

2NF: In 2NF, we need to check for partial dependency. CD which is a proper subset of a candidate key and it determine E, which is non prime attribute. So, given relation is also not in 2NF. S**o, the highest normal form is 1 NF.**

problem 2:

Find the highest normal form of a relation R(A,B,C,D,E) withFD

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 ofrelation, So AC will be candidate key. A or C can't be derived from any other attribute of the relation, so there will be only 1 candidate key {AC}.

Step 2: Prime attribute are those attribute which are part of candidate key {A,C} in this exampleand others will be non-prime {B,D,E} in this example.

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 in 2nd normal form because BC->D is in 2nd normal form (BC is not proper subsetof 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).

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 in B->E (neither B is a super key nor E is a prime attribute) but to satisfy 3rdnormal 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 2nd Normal form.

Decomposition: It is the process of splitting original table into smaller relations such that attribute sets of two relations will be the subset of attribute set of original table.

Rules of decomposition:

If 'R' is a relation splitted into 'R1' and 'R2' relations, the decomposition done should satisfy following-

1) Union of two smaller subsets of attributes gives all attributes of 'R'.

   R1(attributes)UR2(attributes)=R(attributes)

2) Both relations interaction should not give null value.

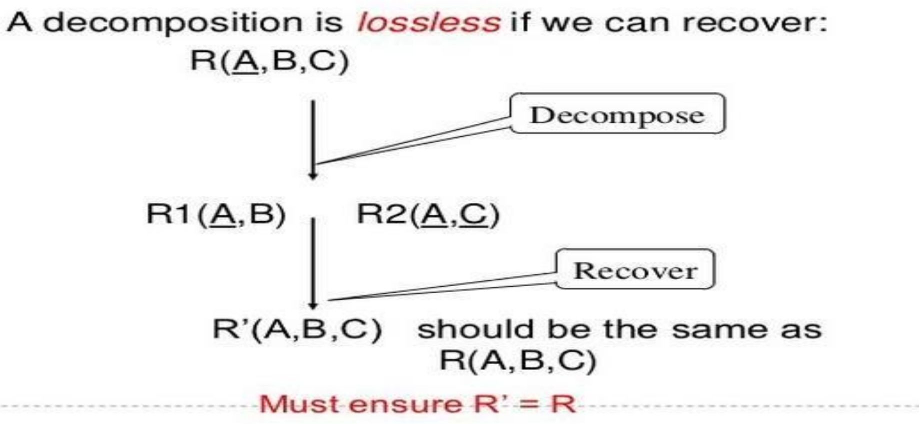
   R1(attributes)∩R2(attributes)!=null

3) Both relations interaction should give key attribute.

   R1(attribute)∩R2(attribute)=R(key attribute)

Properties of decomposition:

Lossless decomposition: while joining two smaller tables no data should be lost and should satisfy all the rules of decomposition. No additional data should be generated on natural join of decomposed tables.



Lossless Decomposition example

- Sometimes the same set of data is reproduced:

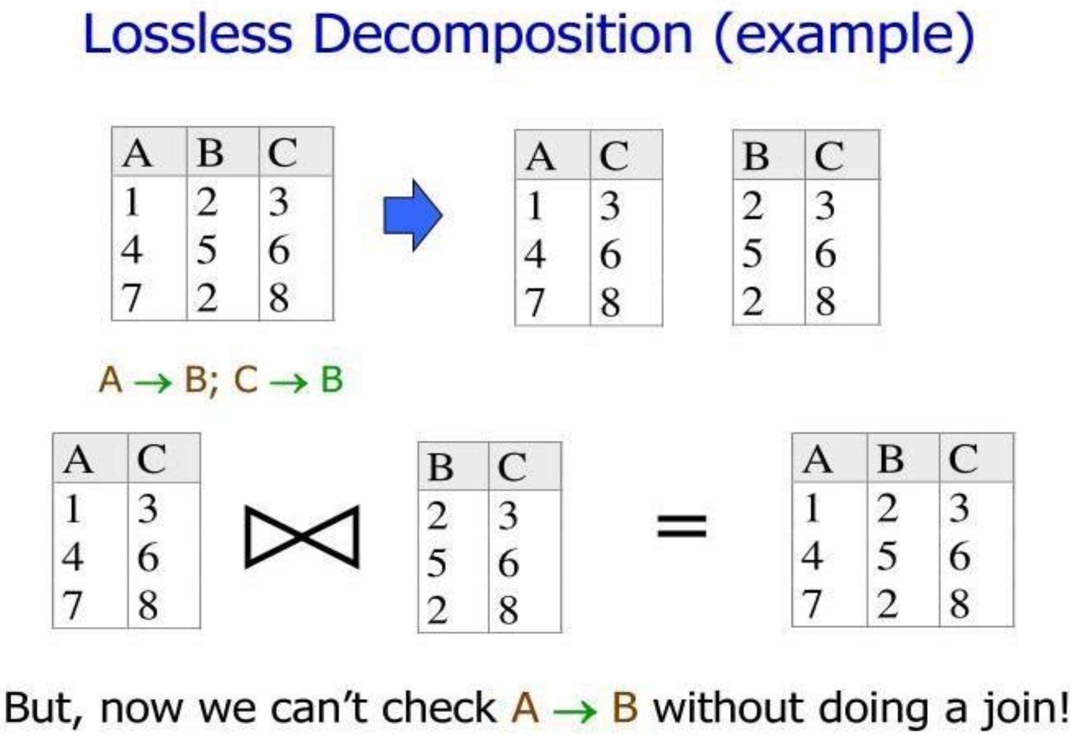
Name	Price	Category
Word	100	WP
Oracle	1000	DB
Access	100	DB

Name	Price
Word	100
Oracle	1000
Access	100

Name	Category
Word	WP
Oracle	DB
Access	DB

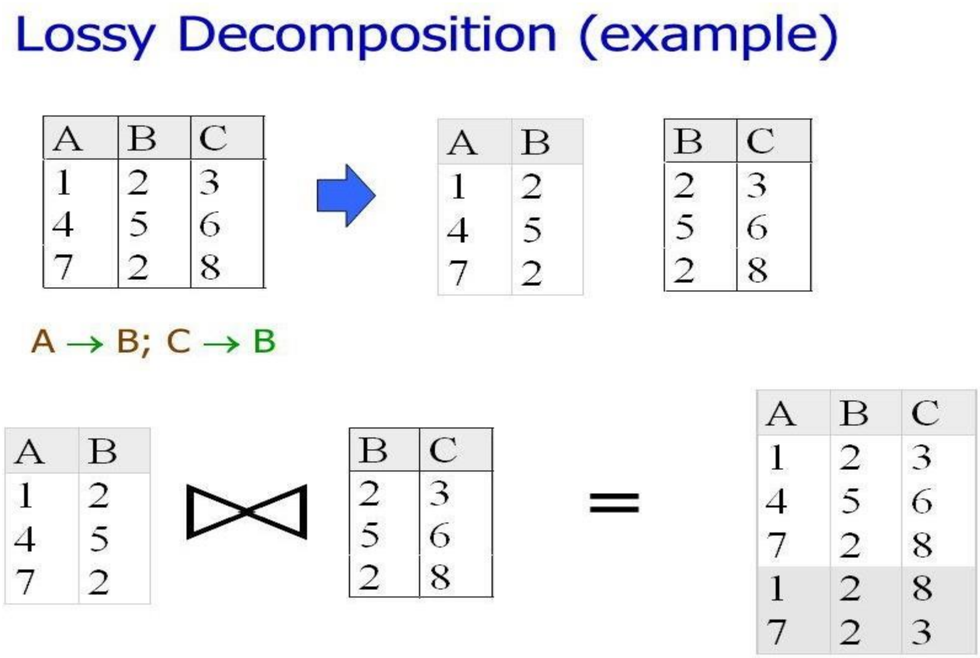
- (Word, 100) + (Word, WP) → (Word, 100, WP)
- (Oracle, 1000) + (Oracle, DB) → (Oracle, 1000, DB)
- (Access, 100) + (Access, DB) → (Access, 100, DB)

example 2 for loseless decomposition:

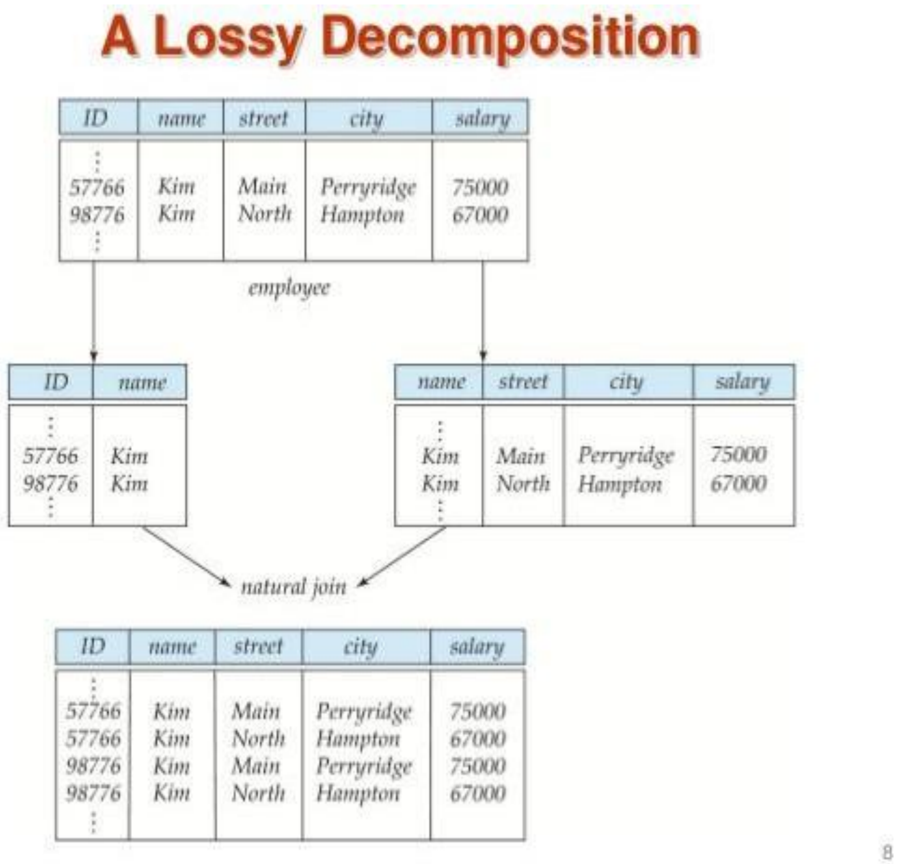


Lossy join decomposition: if information is lost after joining and if do not satisfy any one of theabove rules of decomposition.

example 1:



example 2:



In above examples, on joining decomposed tables, extra tuples are generated.so it is

lossy join decomposition.

Dependency preservation: functional dependencies should be satisfied even after splittingrelations and they should be satisfied by any of splitted tables.

Dependency Preservation

A Decomposition D = { R1, R2, R3….Rn } of R is dependency preserving wrt a set F of Functional dependency if

$(F_1 \cup F_2 \cup \ldots \cup F_m)^+ = F^+$.
Consider a relation R
R ---> F{...with some functional dependency(FD).............................}

R is decomposed or divided into R1 with FD { f1 } and R2 with { f2 }, thenthere 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 for dependency preservation:

<u>Dependency preservation</u>

<u>Example:</u>

$R=(A, B, C)$, $F=\{A \rightarrow B, B \rightarrow C\}$

Decomposition of R: $R1=(A, B)$ $R2=(B, C)$

Does this decomposition preserve the given dependencies?

<u>Solution:</u>

In R1 the following dependencies hold: $F1=\{A \rightarrow B, A \rightarrow A, B \rightarrow B, AB \rightarrow AB\}$

In R2 the following dependencies hold: $F2=\{B \rightarrow B, C \rightarrow C, B \rightarrow C, BC \rightarrow BC\}$

$F' = F1' \cup F2' = \{A \rightarrow B, B \rightarrow C$, trivial dependencies$\}$

In F' all the original dependencies occur, so this decomposition preserves dependencies.

lack of redundancy: It is also known as repetition of information. The proper decomposition should not suffer from any data redundancy.