# SNS COLLEGE OF TECHNOLOGY

**(An Autonomous Institution)**

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

## Department of MCA

## DBMS Lossless Decomposition in DBMS

**Course Name : 23CAT603 - DATA BASE MANAGEMENT SYSTEM**

**Class : I Year / I Semester**

**Unit III – Lossless Decomposition in DBMS**

# Lossless Decomposition in DBMS

The original relation and relation reconstructed from joining decomposed relations must contain the same number of tuples if the number is increased or decreased then it is Lossy Join decomposition.

Lossless join decomposition ensures that never get the situation where spurious tuples are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations.

**What is Lossless Decomposition?**

Lossless join decomposition is a decomposition of a relation R into relations R1, and R2 such that if we perform a natural join of relation R1 and R2, it will return the original relation R. This is effective in removing redundancy from databases while preserving the original data.

In other words by lossless decomposition, it becomes feasible to reconstruct the relation R from decomposed tables R1 and R2 by using Joins.

Only **1NF,2NF,3NF,** and **BCNF** are valid for lossless join decomposition.

In Lossless Decomposition, we select the common attribute and the criteria for selecting a common attribute is that the common attribute must be a candidate key or super key in either relation R1, R2, or both.

Decomposition of a relation R into R1 and R2 is a lossless-join decomposition if at least one of the following functional dependencies is in F+ (Closure of functional dependencies)

# Lossless Decomposition in DBMS

**Example of Lossless Decomposition**

— Employee (Employee_Id, Ename, Salary, Department_Id, Dname)

Can be decomposed using lossless decomposition as,

— Employee_desc (Employee_Id, Ename, Salary, Department_Id)

— Department_desc (Department_Id, Dname)

Alternatively the lossy decomposition would be as joining these tables is not possible so not possible to get back original data.

– Employee_desc (Employee_Id, Ename, Salary)

– Department_desc (Department_Id, Dname)

$R1 \cap R2 \rightarrow R1$

OR

$R1 \cap R2 \rightarrow R2$

# Lossless Decomposition in DBMS

- In a database management system (DBMS), a lossless decomposition is a process of decomposing a relation schema into multiple relations in such a way that it preserves the information contained in the original relation. Specifically, a lossless decomposition is one in which the original relation can be reconstructed by joining the decomposed relations.
- To achieve lossless decomposition, a set of conditions known as Armstrong's axioms can be used. These conditions ensure that the decomposed relations will retain all the information present in the original relation. Specifically, the two most important axioms for lossless decomposition are the reflexivity and the decomposition axiom.
- The reflexivity axiom states that if a set of attributes is a subset of another set of attributes, then the larger set of attributes can be inferred from the smaller set. The decomposition axiom states that if a relation R can be decomposed into two relations R1 and R2, then the original relation R can be reconstructed by taking the natural join of R1 and R2.
- There are several algorithms available for performing lossless decomposition in DBMS, such as the BCNF (Boyce-Codd Normal Form) decomposition and the 3NF (Third Normal Form) decomposition. These algorithms use a set of rules to decompose a relation into multiple relations while ensuring that the original relation can be reconstructed without any loss of information.

# Lossless Decomposition in DBMS

**Advantages of Lossless Decomposition**

**Reduced Data Redundancy:** Lossless decomposition helps in reducing the data redundancy that exists in the original relation. This helps in improving the efficiency of the database system by reducing storage requirements and improving query performance.

**Maintenance and Updates:** Lossless decomposition makes it easier to maintain and update the database since it allows for more granular control over the data.

**Improved Data Integrity:** Decomposing a relation into smaller relations can help to improve data integrity by ensuring that each relation contains only data that is relevant to that relation. This can help to reduce data inconsistencies and errors.

**Improved Flexibility:** Lossless decomposition can improve the flexibility of the database system by allowing for easier modification of the schema.

# Lossless Join and Dependency Preserving Decomposition

Decomposition of a relation is done when a relation in a relational model is not in appropriate normal form. Relation R is decomposed into two or more relations if decomposition is lossless join as well as dependency preserving.

**Lossless Join Decomposition**

If we decompose a relation R into relations R1 and R2,

Decomposition is lossy if $R1 \bowtie R2 \supset R$

Decomposition is lossless if $R1 \bowtie R2 = R$

**To check for lossless join decomposition using the FD set, the following conditions must hold:**

1. The Union of Attributes of R1 and R2 must be equal to the attribute of R. Each attribute of R must be either in R1 or in R2.

Att(R1) U Att(R2) = Att(R)

# Lossless Join and Dependency Preserving Decomposition

2. The intersection of Attributes of R1 and R2 must not be NULL.

Att(R1) ∩ Att(R2) ≠ Φ

3. The common attribute must be a key for at least one relation (R1 or R2)

Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is a lossless join decomposition as:

1.First condition holds true as Att(R1) U Att(R2) = (ABC) U (AD) = (ABCD) = Att(R).

2.Second condition holds true as Att(R1) ∩ Att(R2) = (ABC) ∩ (AD) ≠ Φ

3.The third condition holds as Att(R1) ∩ Att(R2) = A is a key of R1(ABC) because A->BC is given.

**Dependency Preserving Decomposition**

If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from a combination of functional dependency of R1 and R2. For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of R1(ABC).

# Lossless Join and Dependency Preserving Decomposition

**Advantages of Lossless Join and Dependency Preserving Decomposition**

**Improved Data Integrity:** Lossless join and dependency preserving decomposition help to maintain the data integrity of the original relation by ensuring that all dependencies are preserved.

**Reduced Data Redundancy:** These techniques help to reduce data redundancy by breaking down a relation into smaller, more manageable relations.

**Improved Query Performance:** By breaking down a relation into smaller, more focused relations, query performance can be improved.

**Easier Maintenance and Updates:** The smaller, more focused relations are easier to maintain and update than the original relation, making it easier to modify the database schema and update the data.

**Better Flexibility:** Lossless join and dependency preserving decomposition can improve the flexibility of the database system by allowing for easier modification of the schema.

# Lossless Join and Dependency Preserving Decomposition

**Disadvantages of Lossless Join and Dependency Preserving Decomposition**

**Increased Complexity:** Lossless join and dependency-preserving decomposition can increase the complexity of the database system, making it harder to understand and manage.

**Costly:** Decomposing relations can be costly, especially if the database is large and complex. This can require additional resources, such as hardware and personnel.

**Reduced Performance:** Although query performance can be improved in some cases, in others, lossless join and dependency-preserving decomposition can result in reduced query performance due to the need for additional join operations.

**Limited Scalability:** These techniques may not scale well in larger databases, as the number of smaller, focused relations can become unwieldy.

# References


1. https://www.geeksforgeeks.org/lossless-join-and-dependency-preserving-decomposition/?ref=lbp
2. https://www.geeksforgeeks.org/lossless-join-and-dependency-preserving-decomposition/?ref=lbp