Normalization (Decomposition)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

Dr. Hrudaya Kumar Tripathy
School of Computer Engineering,

Normalization

- **Normalization** is the process of decomposing or breaking a relation schema R into fragments (i.e. smaller schemas) R1, R2, ... Rn such that the following conditions hold:
  - ✓ **Lossless decomposition:** The fragments should contain the same information as the original relation
  - ✓ **Dependency preservation:** All the functional dependencies should be preserved within each fragment Ri
  - ✓ **Good form:** Each fragment Ri should be free from any type of redundancy
- In other words, normalization is the process of refining the relational data model. It is used because of the following reasons:
  - ✓ It improves database design
  - ✓ It ensures minimum redundancy of data
  - ✓ It removes anomalies for database activities

Decomposition

- **What is Decomposition?**
  - Decomposition is the process of breaking down in parts or elements.
  - It replaces a relation with a collection of smaller relations.
  - It breaks the table into multiple tables in a database.
  - It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
  - If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition:

- Following are the properties of Decomposition,
  1. Lossless Decomposition
  2. Dependency Preservation
  3. Lack of Data Redundancy

Lossless Decomposition

- **Lossless Decomposition**
  1. Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.
  2. It gives a guarantee that the join will result in the same relation as it was decomposed.

- **Example:**
- Let's take 'E' is the Relational Schema, With instance 'e'; is decomposed into: E1, E2, E3, ... En; With instance: e1, e2, e3, ... en.
- If e1 ⋈ e2 ⋈ e3 . . . . ⋈ en, then it is called as 'Lossless Join Decomposition'.

Lossless Decomposition...

Rules/Property :

The decomposition of a base relation is said to be lossless if the original relation can be recovered back by joining the fragment relations.

- Let R be the base relation, which is decomposed into R1, R2, ...Rn. This decomposition is lossless iff R=R1 on R2 on ... on Rn.

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

  - Union of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.
    - **Att(R1) ∪ Att(R2) = Att(R)**
  - Intersection of attributes of R1 and R2 must not be NULL
    - **Att(R1) ∩ Att(R2) ≠ ϕ**
  - 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)

Lossless Decomposition

- The decomposition of a base relation is said to be lossless if the original relation can be recovered back by joining the fragment relations
- Let R be the base relation, which is decomposed into R1, R2, ... Rn . This decomposition is lossless iff R= R1 ⋈ R2 ⋈ ... ⋈ Rn . In other words, the consecutive fragments are interrelated by the primary key and foreign key relationships.

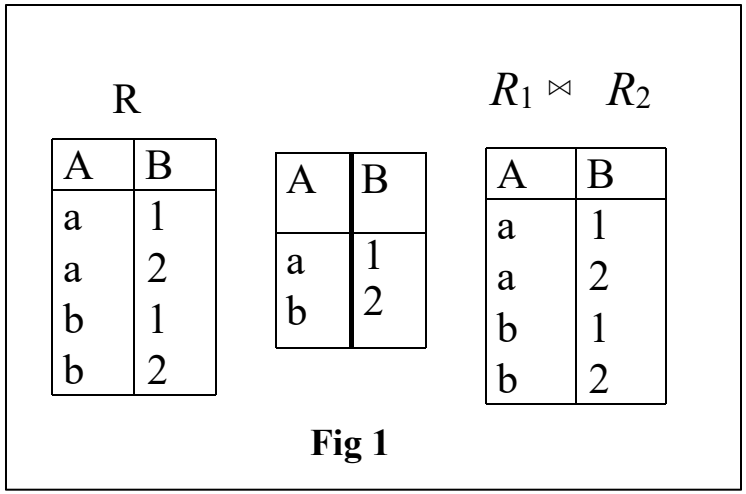

Fig 1

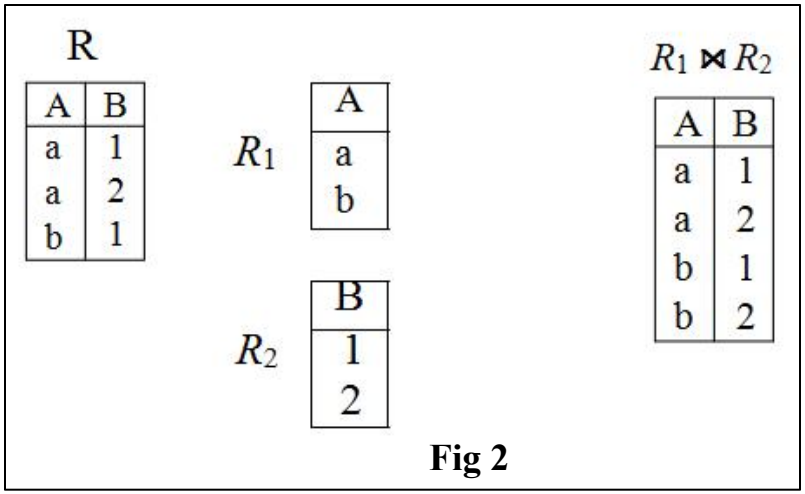

Fig 2

- In Fig.1, as R1 ⋈ R2 = R, **the decomposition is lossless.**
- In Fig.2, as R1 ⋈ R2 != R, **the decomposition is lossy.**

Lossless Decomposition...

- When the base relation schema is decomposed into the fragmented relation schemas, the consecutive relations should be related by primary key - foreign key pair on the common column; so that natural join be possible on the common column.

- Thus, we have to check whether the common column is the key of any relation or not:
  - ✓ **If the common column is the key, the decomposition is lossless**
  - ✓ **If the common column is not the key, the decomposition is lossy**

Lossless Decomposition(Examples)

A	B	C	D
1	a	p	x
2	b	q	y

A	B
1	a
2	b

C	D
p	x
q	y

R1 x R2 we got more extra tuple . so it is not same as orogina table

A	B	C	D
1	a	p	x
1	a	q	y
2	b	p	x
2	b	q	y

Lossless Decomposition...

Relation: R

A	B	c
1	a	p
2	B	q
3	A	r

R1

A	B
1	a
2	b
3	a

R2

B	C
a	p
b	r
a	r

R1 Natural Join R2

A	B	C
1	a	p
1	a	r
2	b	q
3	a	p
3	a	r

Extra attribute are generating also after natural join because the common attribute is not a key attribute. It should be (B) key attribute and the values should be unique . so it is not lossless

Lossless Decomposition...

Q. 1 As per the table which option is lossless Decomposition.

A	B	C	D	E
a	122	1	p	w
b	234	2	q	x
a	568	1	r	y
c	347	3	s	z

1. R1(AB) , R2(CD) (lossy because not satisfying Rule-1)
2. R1(ABC) , R2(DE) (lossy because not satisfying Rule-2)
3. R1(ABC), R2(CDE) (lossy because not satisfying Rule-3, common attribute is not a key)
4. R1(ABCD), R2 (ACDE)
5. R1(ABCD), R2 (DE)
6. R1(ABC), R2(BCD), R3(DE)

Q.2 As per the dependency which option is lossless Decomposition.

R(VWXYZ)
Z->Y
Y->Z
X ->YZ
VZ ->X

1. **R1(VWX), R2(XYZ) (Lossless)**
2. R1(VW), R2(YZ) (not satisfying rule-1)
3. R1(VWX), R2(YZ) (No common attribute hence lossy)
4. R1(VW), R2(WXYZ) (here W is common so calculate close)

Calculate Closure , if the attribute closure contain all attribute of the relation ,then that option is correct.

Lossless Decomposition...

- In the above example, it means that, if natural joins of all the decomposition give the original relation, then it is said to be lossless join decomposition.

Example: <Employee_Department> Table

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

- Decompose the above relation into two relations to check whether a decomposition is lossless or lossy.
- Now, we have decomposed the relation that is Employee and Department.

Lossless Decomposition...

Relation 1 : <Employee> Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Bangalore	25000

- Employee Schema contains (Eid, Ename, Age, City, Salary).

Relation 2 : <Department> Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

- Department Schema contains (Deptid, Eid, DeptName).
- So, the above decomposition is a Lossless Join Decomposition, because the two relations contains one common field that is 'Eid' and therefore join is possible.
- Now apply natural join on the decomposed relations.

Lossless Decomposition...

Employee ⋈ Department

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Bangalore	25000	D005	Human Resource

Hence, the decomposition is Lossless Join Decomposition.

- If the <Employee> table contains (Eid, Ename, Age, City, Salary) and <Department> table contains (Deptid and DeptName), then it is not possible to join the two tables or relations, because there is no common column between them. And it becomes **Lossy Join Decomposition.**

Dependency Preservation

- If a table having FD set F , is decomposed in to two tables R1 and R2 having functional dependency set F1 and F2 then:

  $F1 \subseteq F^+$

  $F2 \subseteq F^+$

$(F1 \cup F2)^+ = F^+$

Q.1 R(ABC)

F: A -> B

B-> C

C->A

So now we can decompose the above relation in two relations R1(AB) and R2(BC)

So F1: A->B , B->A

F2: B->C, C->B

here in the above decomposion we got directly A->B and B->C but we are not getting C->A, so we will calculate C+= CAB, So indirectly we got C->A

Dependency Preservation...

Let R is decomposed into {R1, R2,.....,Rn} with projected FD set {F1,F2,.....Fn}. This decomposition is dependency preserving if $F^+ = \{F1 \cup F2 \cup .....Fn\}^+$.

R = (A, B, C) and F={A→B, B→C, A→C} decomposed into:
R1 = (A, B) and F = {A→B}
R2 = (A, C) and F = {A→C}
The decomposition is lossless because R1 R2 = R
But, the decomposition is not **dependency preserving** because
$(F1 \cup F2)^+ = \{A→B, A→C\}$, Thus $(F1 \cup F2)^+ \neq F^+$ as we lost the FD B→C.

R = (A, B, C) and F={A→B, B→C, A→C} decomposed into:
R1 = (A, B) and F = {A→B}
R2 = (B, C) and F = {B→C}
The decomposition is lossless because R1 R2 = R
But, the decomposition is **dependency preserving** because
$(F1 \cup F2)^+ = \{A→B, B→C, A→C\}$, Thus $(F1 \cup F2)^+ = F^+$

Dependency Preservation

- The decomposition of a relation schema R with FDs F is a set of fragment relations Ri with FDs Fi , where Fi is the subset of dependencies in F + that include only attributes in Ri . The decomposition is dependency preserving if and only if

$$(\cup i \ \mathbf{Fi})+ = \mathbf{F}+$$

- R = (A, B, C) and F={A→B, B→C, A→C}, Key=A.

R

A	B	C
1	2	3
2	2	3
3	2	3
4	3	4

- Here, Key=A
- The decomposition is lossless because $R_1 \bowtie R_2 = R$
- But, the decomposition is not dependency preserving because

  $(F_1 \cup F_2)^+$={A→B, A→C}
- Thus, $(F_1 \cup F_2)^+ \ != F^+$ as we lost the FD B→C.

R_1 = (A, B) and F={A→B}

A	B
1	2
2	2
3	2
4	3

R_2 = (A, C) and F={A→C}

A	C
1	3
2	3
3	3
4	4

Guidelines followed in Designing Good Database

- **Guideline 1: Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity sets and relationship sets into a single relation.**
  - ✓ If a relation schema corresponds to one entity set or one relationship set, then the meaning tends to be simple and clear. Otherwise, the relation corresponds to a mixture of multiple entities and relationships and hence becomes complex and unclear.
  - ✓ Only foreign keys should be used to refer to other entities. Entity and relationship attributes should be kept apart as much as possible.
  - ✓ Bottom Line: design the schemas that can be explained easily relation by relation. In such cases, the semantics of attributes should be easy to interpret.
- **Guideline 2: Design the base relation schemas in such a way that the anomalies such as insertion, deletion, or updation anomalies are removed from the relations**
  - ✓ If any anomalies are present, note them clearly and make sure that the programs that modify (update) the database will operate correctly.

Guidelines followed in Designing Good Database

- **Guideline 3: Avoid placing attributes in a base relation whose values may frequently be NULL**
  - ✓ If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.
  - ✓ Attributes that are NULL frequently could be placed in separate relations (with the primary key).
- **Guideline 4: Design the relation schemas so that they can be joined in such a way that no spurious tuples are generated**
  - ✓ Avoid relations that contain matching attributes that are not (foreign key and primary key) combinations, because joining on such attributes may produce spurious tuples.


THANK
YOU!