Normalisation

Stéphane Bressan

Three Methods

The three common methods for relational database schema design are the Decomposition Method, the Synthesis Method, and the Entity-Relationship Approach.

Decomposition

The decomposition method is based on the assumption that a database can be represented by a universal relation[a] which contains all the attributes of the database (this is called the universal relation assumption). This relation is then decomposed into smaller relations, fragments, in order to remove redundant data.

[a] Synthesis method assumes universal relation assumption also, However the decomposition and synthesis method can be applied to parts of the design.

Synthesis

The synthesis method is based on the assumption that a database can be described by a given set of attributes and a given set of functional dependencies. The 3NF or BCNF relations, fragments, are then synthesized based on the given set of dependencies.

Entity-Relationship

You should be familiar with the entity-relationship approach but we will discuss it in a refresher video lecture.

Two Criteria

The two main criteria for the decomposition and synthesis methods are lossslessness (reconstructability) and dependency preservation (covering).

number	name	department	position	salary
1XU3	Dewi Srijaya	sales	clerk	2000
5CT4	Axel Bayer	marketing	trainee	1200
4XR2	John Smith	accounting	clerk	2000
7HG5	Eric Wei	sales	assistant manager	2200
4DE3	Winnie Lee	accounting	manager	3000
8HG5	Sylvia Tok	marketing	manager	3000
null	null	null	security guard	1500

The table above records the salaries of the different employees in our organisation. An agreement with the trade unions imposes that salaries are determined by the position. The actual value has been negotiated and fixed. The salary of a clerk is 2000$ per month, the salary of a manager is 3000$, the salary of a security guard is 1500$ per month, etc.

The solution to avoid anomalies is to decompose the table into two fragments.

number	name	department	position
1XU3	Dewi Srijaya	sales	clerk
5CT4	Axel Bayer	marketing	trainee
4XR2	John Smith	accounting	clerk
7HG5	Eric Wei	sales	assistant manager
4DE3	Winnie Lee	accounting	manager
8HG5	Sylvia Tok	marketing	manager

position	salary
clerk	2000
trainee	1200
assistant manager	2200
manager	3000
security guard	1500

Losslessness

The natural join (see the universal relation assumption, assume no null values) of all the fragments is equivalent to the original relation.

$$R = R_1 \bowtie \cdots \bowtie R_n$$

This is a join dependency (binary decomposition can be tested with multi-valued dependencies)

The decomposition is lossless if the full outer join of the two tables on the condition that their primary and foreign keys are equal reconstitutes the initial table

employee			
number	name	department	position
1XU3	Dewi Srijaya	sales	clerk
5CT4	Axel Bayer	marketing	trainee
4XR2	John Smith	accounting	clerk
7HG5	Eric Wei	sales	assistant manager
4DE3	Winnie Lee	accounting	manager
8HG5	Sylvia Tok	marketing	manager

salary	
position	salary
clerk	2000
trainee	1200
assistant manager	2200
manager	3000
security guard	1500

```sql
SELECT *
FROM employee e FULL OUTER JOIN salary s ON e.position = s.position
```

Consider a relation R with a set of functional dependencies Σ. A set Σ' of projected functional dependencies on R' from R with Σ, where $R' \subset R$, is the set of functional dependencies equivalent to the set of functional dependencies $X \rightarrow Y$ in Σ^+ such that $X \subset R'$ and $Y \subset R'$.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

What is a set of projected functional dependencies Σ' on $R' = \{A, B, D, E\}$ from R with Σ?

$$R = \{A, B, C, D, E\}$$
$$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$$

$$\Sigma' = \{\{A, B\} \to \{E\}, \{B\} \to \{E\}, \{B\} \to \{D\}\}$$

Dependency Preservation

We want to preserve the information captured by the functional dependencies. The union of the projected sets of functional dependencies, Σ_1, ..., Σ_n, must be equivalent to the original set of functional dependencies, Σ.

$$\Sigma^+ = (\Sigma_1 \cup \cdots \cup \Sigma_n)^+$$

Dependency Preservation

We want to preserve the information captured by the functional dependencies. The union of the projected sets of functional dependencies, Σ_1, ..., Σ_n, must be equivalent to the original set of functional dependencies, Σ.

$$\Sigma^+ = (\Sigma_1 \cup \cdots \cup \Sigma_n)^+$$

Is it the following true?

$$\Sigma_1^+ \cup \Sigma_2^+ = (\Sigma_1 \cup \Sigma_2)^+$$

(In general, no!)

In the running example, the proposed decomposition is dependency preserving since:

$$\Sigma = \{\{number\} \rightarrow \{name, department, position\}, \{position\} \rightarrow \{salary\}\}$$

$$\Sigma_1 = \{\{number\} \rightarrow \{name, department, position\}\}$$

$$\Sigma_2 = \{\{position\} \rightarrow \{salary\}\}$$

$$\Sigma^+ = (\Sigma_1 \cup \Sigma_2)^+$$

Decomposition Algorithm, Idea

When a relation is not in BCNF[a], we can pick one of the functional dependencies violating the BCNF definition and use it to decompose the relation into two relations. We continue decomposing until every fragment is in BCNF.

[a] The same algorithm work for other normal forms.

Decomposition Algorithm, Idea

When a relation is not in BCNF[a], we can pick one of the functional dependencies violating the BCNF definition and use it to decompose the relation into two relations. We continue decomposing until every fragment is in BCNF.

[a] The same algorithm work for other normal forms.

The decomposition algorithm is guaranteed to find a lossless decomposition in BCNF.

Decomposition Algorithm, Idea

When a relation is not in BCNF[a], we can pick one of the functional dependencies violating the BCNF definition and use it to decompose the relation into two relations. We continue decomposing until every fragment is in BCNF.

[a] The same algorithm work for other normal forms.

The decomposition algorithm is guaranteed to find a lossless decomposition in BCNF.

The decomposition may not be dependency preserving.

Decomposition Algorithm

Let $X \rightarrow Y$ be a functional dependency in Σ that violates the BCNF definition (it is not trivial and X is not a superkey). We use it decompose R into the following two relations R_1 and R_2.

- $R_1 = X^+$,
- $R_2 = (R - X^+) \cup X$.

We must now check whether R_1 and R_2 with the respective sets of projected functional dependencies Σ_1 and Σ_2 are in BCNF.

$$R = \{A, B, C, D, E\}$$
$$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$$

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

Let us decompose R with Σ into a lossless decomposition in BCNF. Is the decomposition lossless?

$\{B\} \rightarrow \{C\}$ violates the BCNF definition (it is non-trivial and $\{B\}$ is not a superkey.

$\{B\} \rightarrow \{C\}$ violates the BCNF definition (it is non-trivial and $\{B\}$ is not a superkey.

We decompose into two fragments.

$\{B\} \rightarrow \{C\}$ violates the BCNF definition (it is non-trivial and $\{B\}$ is not a superkey.

We decompose into two fragments.

$R_1 = \{B\}^+ = \{B, C, D, E\}$ with the projected functional dependencies
$\Sigma_1 = \{\{B\} \rightarrow \{C, D, E\}, \{C\} \rightarrow \{B\}\}$.

$\{B\} \rightarrow \{C\}$ violates the BCNF definition (it is non-trivial and $\{B\}$ is not a superkey.

We decompose into two fragments.

$R_1 = \{B\}^+ = \{B, C, D, E\}$ with the projected functional dependencies
$\Sigma_1 = \{\{B\} \rightarrow \{C, D, E\}, \{C\} \rightarrow \{B\}\}$.

$R_2 = (R - \{B\}^+) \cup \{B\} = \{A, B\}$ with the projected functional dependencies $\Sigma_2 = \emptyset$.

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Yes they are in BCNF. We can stop here. Otherwise we would have to continue decomposing whichever fragments are not in BCNF.

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Yes they are in BCNF. We can stop here. Otherwise we would have to continue decomposing whichever fragments are not in BCNF.

The decomposition is guaranteed to be lossless (by properties of the algorithm).

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Yes they are in BCNF. We can stop here. Otherwise we would have to continue decomposing whichever fragments are not in BCNF.

The decomposition is guaranteed to be lossless (by properties of the algorithm).

Have we lost any functional dependency?

Are R_1 with Σ_1 and R_2 with Σ_2 in BCNF?

Yes they are in BCNF. We can stop here. Otherwise we would have to continue decomposing whichever fragments are not in BCNF.

The decomposition is guaranteed to be lossless (by properties of the algorithm).

Have we lost any functional dependency?

No, we can recover all the functional dependencies in Σ from $\Sigma_1 \cup \Sigma_2$. The decomposition is dependency preserving.

Can we choose another dependency to decompose and reach a different result?

Synthesis Algorithm

When a relation is not in 3NF, we can synthesise a schema in 3NF from a minimal cover of the set of functional dependencies.

Synthesis Algorithm

When a relation is not in 3NF, we can synthesise a schema in 3NF from a minimal cover of the set of functional dependencies.

- For each functional dependency $X \to Y$ in the minimal cover create a relation $R_i = X \cup Y$ unless it already exists or is subsumed by another relation.

Synthesis Algorithm

When a relation is not in 3NF, we can synthesise a schema in 3NF from a minimal cover of the set of functional dependencies.

- For each functional dependency $X \rightarrow Y$ in the minimal cover create a relation $R_i = X \cup Y$ unless it already exists or is subsumed by another relation.
- If none of the created relations contain one of the keys, pic a candidate key and create a relation with that candidate key.

Synthesis Algorithm

When a relation is not in 3NF, we can synthesise a schema in 3NF from a minimal cover of the set of functional dependencies.

- For each functional dependency $X \to Y$ in the minimal cover create a relation $R_i = X \cup Y$ unless it already exists or is subsumed by another relation.
- If none of the created relations contain one of the keys, pic a candidate key and create a relation with that candidate key.

In order to avoid unnecessary decomposition, it is generally a good idea to use a compact minimal cover (we shall do so unless we explicitly identify a problem).

Synthesis Algorithm

When a relation is not in 3NF, we can synthesise a schema in 3NF from a minimal cover of the set of functional dependencies.

- For each functional dependency $X \to Y$ in the minimal cover create a relation $R_i = X \cup Y$ unless it already exists or is subsumed by another relation.
- If none of the created relations contain one of the keys, pic a candidate key and create a relation with that candidate key.

In order to avoid unnecessary decomposition, it is generally a good idea to use a compact minimal cover (we shall do so unless we explicitly identify a problem).

This is the commonly used variant of the original Bernstein algorithm. The original Bernstein algorithm takes care of some interesting special cases not dealt with here but also misses some.

The synthesis algorithm is guaranteed to find a lossless decomposition in 3NF.

The synthesis algorithm is guaranteed to find a lossless decomposition in 3NF.

The decomposition is always dependency preserving.

The synthesis algorithm is guaranteed to find a lossless decomposition in 3NF.

The decomposition is always dependency preserving.

Very often, the decomposition is also in BCNF.

$$R = \{A, B, C, D, E\}$$
$$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$$

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

Let us decompose R with Σ into a lossless dependency preserving decomposition in 3NF.

Example

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

We compute the candidate key of R with Σ.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

We compute the candidate key of R with Σ.

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \to \{C, D, E\}, \{A, C\} \to \{B, D, E\}, \{B\} \to \{C\}, \{C\} \to \{B\}, \{C\} \to \{D\}, \{B\} \to \{E\}, \{C\} \to \{E\}\}$

We compute the candidate key of R with Σ.

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

We compute a compact minimal cover of R with Σ

$R = \{A, B, C, D, E\}$
$\Sigma = \{\{A, B\} \rightarrow \{C, D, E\}, \{A, C\} \rightarrow \{B, D, E\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\}, \{C\} \rightarrow \{D\}, \{B\} \rightarrow \{E\}, \{C\} \rightarrow \{E\}\}$

We compute the candidate key of R with Σ.

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

We compute a compact minimal cover of R with Σ

$\Sigma'' = \{\{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B, D, E\}\}$

$$\Sigma'' = \{\{B\} \to \{C\}, \{C\} \to \{B, D, E\}\}$$

$$\Sigma'' = \{\{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B, D, E\}\}$$

We synthesise a relation for each functional dependency.

$\Sigma'' = \{\{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B, D, E\}\}$

We synthesise a relation for each functional dependency.

$R_1 = \{\underline{B}, C\}$ ($\{B\}$ is guaranteed to be candidate key of R_1 by construction).
$R_2 = \{B, \underline{C}, D, E\}$ ($\{C\}$ is guaranteed to be candidate key of R_2 by construction).

$\Sigma'' = \{\{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B, D, E\}\}$

We synthesise a relation for each functional dependency.

$R_1 = \{\underline{B}, C\}$ ($\{B\}$ is guaranteed to be candidate key of R_1 by construction).
$R_2 = \{B, \underline{C}, D, E\}$ ($\{C\}$ is guaranteed to be candidate key of R_2 by construction).

R_2 subsumes R_1. We eliminate R_1.

$\Sigma'' = \{\{B\} \rightarrow \{C\}, \{C\} \rightarrow \{B, D, E\}\}$

We synthesise a relation for each functional dependency.

$R_1 = \{\underline{B}, C\}$ ($\{B\}$ is guaranteed to be candidate key of R_1 by construction).
$R_2 = \{B, \underline{C}, D, E\}$ ($\{C\}$ is guaranteed to be candidate key of R_2 by construction).

R_2 subsumes R_1. We eliminate R_1.

$R_2 = \{B, C, D, E\}$

$R_2 = \{B, C, D, E\}$

$R_2 = \{B, C, D, E\}$

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

$R_2 = \{B, C, D, E\}$

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

No relation contains one of our candidate keys of R. We add a relation with one of the candidate key.

$R_2 = \{B, C, D, E\}$

The two candidate keys are $\{A, C\}$ and $\{A, B\}$.

No relation contains one of our candidate keys of R. We add a relation with one of the candidate key.

$R_3 = \{A, C\}$ ($\{A, C\}$ is guaranteed to be candidate key of R_3 by construction).

The resulting decomposition is:

The resulting decomposition is:

$R_2 = \{B, C, D, E\}$ with $\Sigma_2 = \{\{B\} \to \{C\}, \{C\} \to \{B, D, E\}\}$. It is in BCNF ($\{B\}$ is also a candidate key)!

The resulting decomposition is:

$R_2 = \{B, C, D, E\}$ with $\Sigma_2 = \{\{B\} \to \{C\}, \{C\} \to \{B, D, E\}\}$. It is in BCNF ($\{B\}$ is also a candidate key)!

$R_3 = \{A, C\}$ with $\Sigma_3 = \emptyset$. It is in BCNF.

We could also have decomposed as:
$R_2 = \{B, C, D, E\}$.
$R_3 = \{A, B\}$.

We could also have decomposed as:
$R_2 = \{B, C, D, E\}$.
$R_3 = \{A, B\}$.

We could also have decomposed as:
$R_1 = \{B, C\}$.
$R_2 = \{B, D, E\}$.
$R_3 = \{A, B\}$.

We could also have decomposed as:
$R_2 = \{B, C, D, E\}$.
$R_3 = \{A, B\}$.

We could also have decomposed as:
$R_1 = \{B, C\}$.
$R_2 = \{B, D, E\}$.
$R_3 = \{A, B\}$.

etc.

Catalog		
Course	Lecturer	Text
Programming	Tan CK	The Art of Programming
Programming	Tan CK	Java
Programming	Lee SL	The Art of Programming
Programming	Lee SL	Java
DS and Alg.	Tan CK	Java
...		

$$Course \twoheadrightarrow Lecturer$$

$$Course \twoheadrightarrow Text$$

Example

Catalog_L	
Course	Lecturer
Programming	Tan CK
Programming	Lee SL
DS and Alg.	Tan CK
...	

Catalog_T	
Course	Text
Programming	The Art of Programming
Programming	Java
DS and Alg.	Java
...	

Theorem

A relation schema R satisfies the multi-valued dependency $X \twoheadrightarrow Y$ if and only if every valid instance of R is such that :

$$r = \pi_{X \cup Y}(r) \bowtie \pi_{X \cup \, (R - Y)}(r)$$

Theorem

A relation schema R satisfies the multi-valued dependency $X \twoheadrightarrow Y$ if and only if every valid instance of R is such that :

$$r = \pi_{X \cup Y}(r) \bowtie \pi_{X \cup\ (R-Y)}(r)$$

$R(X, Y, Z)$ is the join of its projections $R_1(X, Y)$ and $R_2(X, Z)$.

Decomposition into 4NF

If $X \twoheadrightarrow Y$ is a 4NF violation for relation R, we can decompose R using the same technique as for BCNF.

1. $X \cup Y$ is one of the decomposed relations.
2. All but $Y - X$ is the other.

Theorem

Any relation can be non-loss decomposed into an equivalent collection of 4NF relations.

Shortcomings

- The algorithm is not dependency preserving (no algorthm can be dependency preserving because there might not exists a lossless dependency preserving decomposition in Fourth Normal form. Why?).
- There may be several possible decompositions.
- It does not always find all the keys.
- Decomposition in 4NF may exists but not reachable by binary decomposition.

Another Method [by Ling Tok Wang]

1. Normalize the relation R into a set of 3NF and/or BCNF relations based on the given set of FDs.
2. For each relation not in 4NF, if all attributes belong to the same key and there exists non-trivial MVDs in the relation, then decompose the relation into 2 smaller relations (don't if you loose functional dependencies).