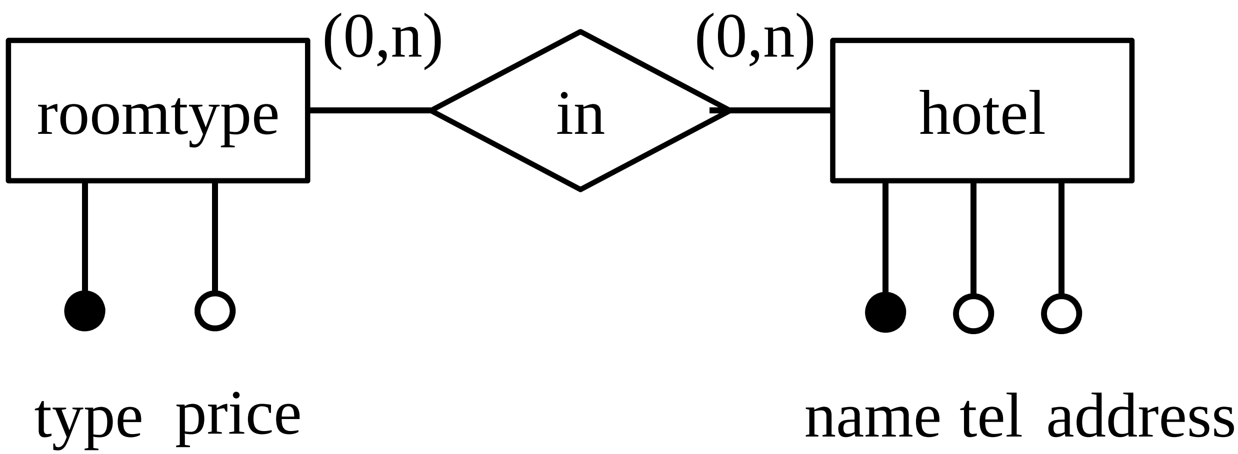
Functional Dependencies

Stéphane Bressan

NUS
National University of Singapore


(0,n)
(0,n)
roomtype
in
hotel
type
price
name
tel
address

name	tel	address
Sangria Clarke Quay	65166516	1 Clarke Quay
Sangria Holland V	65165555	13 Holland Drive

type	name
superior	Sangria Clarke Quay
standard	Sangria Clarke Quay
suite	Sangria Clarke Quay
superior	Sangria Clarke Quay
standard	Sangria Holland V
suite	Sangria Holland V
junior suite	Sangria Holland V

type	price
superior	145
standard	75
suite	250
junior suite	200
executive	175

One Table

type	price	name	tel	address
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	75	Sangria Clarke Quay	65166516	1 Clarke Quay
suite	250	Sangria Clarke Quay	65166516	1 Clarke Quay
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	76	Sangria Holland V	65166516	13 Holland Drive
suite	250	Sangria Holland V	65165555	13 Holland Drive
junior suite	200	Sangria Holland V	65165555	13 Holland Drive
executive	175			

A room of a given type has a fixed price for all hotels in the chain. A hotel, known by its name, has one telephone number and one address.

Redundant Storage

type	price	name	tel	address
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	75	Sangria Clarke Quay	65166516	1 Clarke Quay
suite	250	Sangria Clarke Quay	65166516	1 Clarke Quay
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	76	Sangria Holland V	65166516	13 Holland Drive
suite	250	Sangria Holland V	65165555	13 Holland Drive
junior suite	200	Sangria Holland V	65165555	13 Holland Drive
executive	175			

The same information is repeated, possibly unnecessarily.

Update Anomaly

type	price	name	tel	address
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	75	Sangria Clarke Quay	65166516	1 Clarke Quay
suite	250	Sangria Clarke Quay	65166516	1 Clarke Quay
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	76	Sangria Holland V	65166516	13 Holland Drive
suite	250	Sangria Holland V	65165555	13 Holland Drive
junior suite	200	Sangria Holland V	65165555	13 Holland Drive
executive	175			

The price of standard rooms is replicated and was wrongly entered in one of the replicas.

Deletion Anomaly

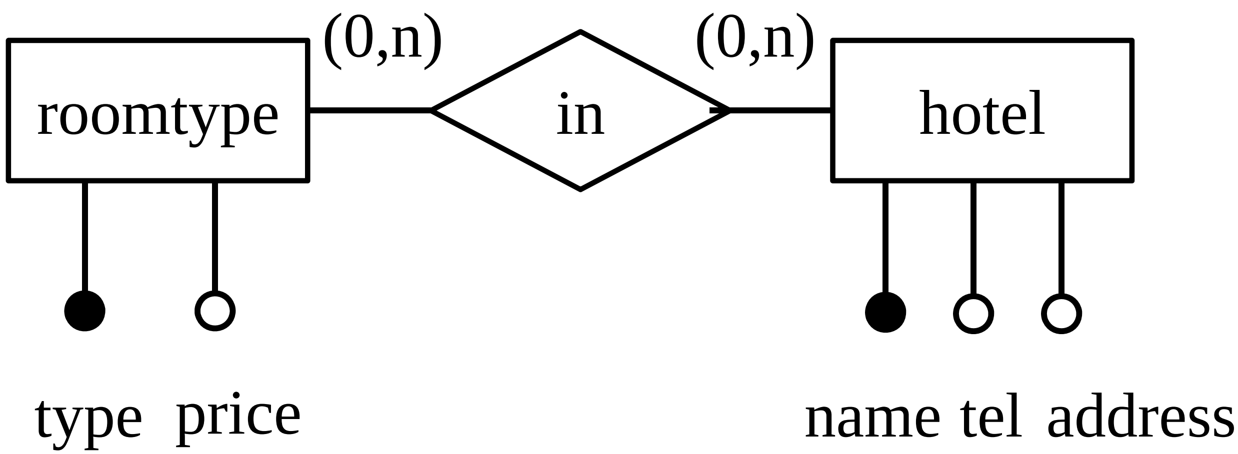
type	price	name	tel	address
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	75	Sangria Clarke Quay	65166516	1 Clarke Quay
suite	250	Sangria Clarke Quay	65166516	1 Clarke Quay
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	76	Sangria Holland V	65166516	13 Holland Drive
suite	250	Sangria Holland V	65165555	13 Holland Drive
junior suite	200	Sangria Holland V	65165555	13 Holland Drive
executive	175			

If Sangria Holland V stops offering junior suites, then their price disappears from the database.

Insertion Anomaly

type	price	name	tel	address
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	75	Sangria Clarke Quay	65166516	1 Clarke Quay
suite	250	Sangria Clarke Quay	65166516	1 Clarke Quay
superior	145	Sangria Clarke Quay	65166516	1 Clarke Quay
standard	76	Sangria Holland V	65166516	13 Holland Drive
suite	250	Sangria Holland V	65165555	13 Holland Drive
junior suite	200	Sangria Holland V	65165555	13 Holland Drive
executive	175			

No hotels offers executive rooms yet. We cannot store their price.



We have the following functional dependencies:

{type} → {price}

and

{name} → {tel, address}

Learning Objectives

- **Define functional dependencies**
- **Manipulate functional dependencies (closure and equivalence)**
- **Reason about functional dependencies (Armstrong's axioms)**
- **Remove redundancy in dependencies and in sets of functional dependencies (minimal cover)**

Functional Dependencies

For a relation scheme R, a functional dependency from a set S of attributes of R to a set T of attributes of R exists if and only if:

For every instance of R, if two tuples in R agree on the values of the attributes in S, then they agree on the values of the attributes in T.

We write: S $\rightarrow$ T

Functional Dependencies

For R = `{type, price, name, tel, address}`, a functional dependency `{type}` $\rightarrow$ `{price}` exists if and only if:

For every instance of R, if two tuples in R agree on the value of the `type` attribute (they have the same type), then they agree on the value of the `price` attribute (they have the same price).

Functional Dependencies are Integrity Constraints

For R = `{type, price, name, tel, address}` and a functional dependency `{type}` → `{price}`, a functional dependency is an integrity constraint that could be checked as follows.

```
CHECK ( NOT EXISTS (
        SELECT *
        FROM R r1, R r2
        WHERE r1.type = r2.type
        AND r1.price<>  r2.price)),
```

We check that it is not violated. It is not the case that two rooms of the same type have different prices.

Functional Dependencies are best Implemented as Primary Keys or Unique Constraints

For R1 = `{type, price}` and a functional dependency `{type}` $\rightarrow$ `{price}`, a functional dependency is an integrity constraint that could be checked as follows (but it also eliminates duplicates).

```
type VARCHAR(36) UNIQUE,
```

or

```
type VARCHAR(36) PRIMARY KEY,
```

We could also add to the table R2 = `{type, name}` the following foreign key constraint.

```
type VARCHAR(36) REFERENCES R1(type),
```

Superkeys

A set of attributes whose knowledge determines the value of the entire tuple is a superkey.

For `R = {type, price, name, tel, address}`

with the following set of functional dependencies

`F = {{type}→{price},{name}→{tel, address}}`

`{type, price, name, tel, address},`
`{type, price, name, tel},`
`{type, price, name},`
`{type, name, tel, address},`
`{type, name, tel},`
`{type, name}` are superkeys.

Candidate Keys

A minimal (for inclusion*) set of attributes whose knowledge determines the value of the entire tuple is a candidate key.

For `R = {type, price, name, tel, address}`

with the following set of functional dependencies

`F = {{type}→{price},{name}→{tel, address}}`

`{type, name}` is the only candidate key.

*** For example `{firstname, lastname}` is not smaller than `{passport}` for inclusion.**
`{lastname}` is smaller than `{firstname, lastname}` for inclusion.

Candidate Keys

A minimal (for inclusion*) set of attributes whose knowledge determines the value of the entire tuple is a candidate key.

For `R = {type, price, name, tel, address}`

with the following set of functional dependencies

`F = {{type}→{price},{name}→{address}, {address}→{tel},{tel}→{name}}`

`{type, name}`,

`{type, address}`

`{type, tel}` are the candidate keys.

*** For example `{firstname, lastname}` is not smaller than `{passport}` for inclusion. `{lastname}` is smaller than `{firstname, lastname}` for inclusion.**

Primary Keys

If there are several candidate keys, the designer chooses one candidate key to be the primary key.

Trivial FDs

$X \rightarrow Y$

$Y \subset X$

For example:

`{type, name}` → `{type}`

Non-Trivial FDs

$X \rightarrow Y$

$Y \not\subset X$

For example:

{type} $\rightarrow$ {price}
{type, name} $\rightarrow$ {price, name}

Completely Non-Trivial FDs

$X \rightarrow \mathbf{Y}$

$Y \not\subset \mathbf{X}$ **and** $Y \cap \mathbf{X} = \emptyset$

For example:

`{type}` $\rightarrow$ `{price}`

Reasoning about Functional Dependencies

It is sometimes possible to infer new functional dependencies from a set of given functional dependencies or to simplify a set of given functional dependencies.

(independently from any particular instance of the relation scheme or of any additional knowledge)

Reasoning about Functional Dependencies

For example:

From

`{name} → {tel}`

and

`{name} →{address}`

We can infer

`{name} → {tel, address}`

Armstrong's Axioms

Let X, Y, Z be subsets of the relation scheme of a relation R

Reflexivity:

If $Y \subset X$, then $X \rightarrow Y$

Augmentation:

If $X \rightarrow Y$, then $X \cup Z \rightarrow Y \cup Z$

Transitivity:

If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

Armstrong's Axioms

Reflexivity: If $Y \subset X$, then $X \rightarrow Y$

For `R = {type, price, name, tel, address}`
with the following set of functional dependencies
`F = {{type}`$\rightarrow$`{price},{name}`$\rightarrow$`{tel, address}}`

1. **It is a fact that `{type}` $\subset$ `{type, name}`,**
2. **Therefore `{type, name}` $\rightarrow$ `{type}`, by reflexivity with (1).**

Armstrong's Axioms

Augmentation: If $X \rightarrow Y$, then $X \cup Z \rightarrow Y \cup Z$

For `R = {type, price, name, tel, address}`
with the following set of functional dependencies
`F = {{type}`$\rightarrow$`{price},{name}`$\rightarrow$`{tel, address}}`

1. **We know that `{type}` $\rightarrow$ `{price}`**
2. **Therefore `{type, name}` $\rightarrow$ `{price, name}` by augmentation of (1) with `{name}`.**

Armstrong's Axioms

Transitivity: If X→Y and Y→Z, then X→Z

For `R = {type, price, name, tel, address}`

with the following set of functional dependencies

`F = {{type}→{price},{name}→{address}, {address}→{tel},{tel}→{name}}`

1. **We know that `{name}→{address}`,**
2. **We know that `{address}→{tel}`**
3. **Therefore `{name}→{tel}` by transitivity of (1) and (2).**

Other Axioms

For instance, Weak-Augmentation:
Let X, Y, Z be subsets of the relation R

$$\text{If } X \rightarrow Y \text{, then } X \cup Z \rightarrow Y$$

Proof

1. Let R be a relation scheme
2. Let $X \rightarrow Y$ be a functional dependency on R
3. Therefore $X \cup Z \rightarrow Y \cup Z$ by Augmentation of (2) with Z
4. We know that $Y \cup Z \rightarrow Y$ by Reflexivity because $Y \subset Y \cup Z$
5. Therefore $X \cup Z \rightarrow Y$ by Transitivity of (3) and (4)

Q.E.D.

Armstrong's Axioms

The Armstrong's axioms are sound

The Armstrong's axioms are complete

Finding Keys: Example

Example: Consider the relation scheme `R = {A,B,C,D}` with the set of functional dependencies:

$$F = \{\{A\}\rightarrow\{C\},\ \{B\}\rightarrow\{D\}\}.$$

Is `{A,B}` a candidate key?

Finding Keys: Example

Example: `{A,B}` is a superkey.

Proof

1. **We know that `{A}` → `{C}`.**
2. **Therefore `{A,B}` → `{A,B,C}`, by augmentation of (1) with `{A,B}`.**
3. **We know that `{B}` → `{D}`.**
4. **Therefore `{A,B,C}` → `{A,B,C,D}`, by augmentation of (3) with `{A, B, C}`.**
5. **Therefore `{A,B}` →`{A,B,C,D}` by transitivity of (2) and (4).**

Q.E.D

Finding Keys: Example

Example: `{A,B}` is a candidate key (minimal)

We must show that neither `{A}` nor `{B}` alone are candidate keys

This can be done by producing counter example relation instance verifying the functional dependencies given but neither $\{A\} \rightarrow \{A,B,C,D\}$ nor $\{B\} \rightarrow \{A,B,C,D\}$

We will however learn an algorithm to do otherwise

Closure of a Set of Functional Dependencies

For a set F of functional dependencies, we call the closure of F, noted F+, the set of all the functional dependencies that F entails.

Armstrong's Axioms

The Armstrong's axioms are complete

F+ can be computed by applying the Armstrong Axioms in all possible ways until nothing new is created (it is called a fixpoint).

Closure of a Set of Functional Dependencies

Consider the relation scheme R = {A,B,C,D},

- **F = {{A} →{B},{B,C} →{D}}**
- **F+ = {{A} →{A}, {B}→{B}, {C}→{C}, {D}→{D}, […], {A}→{B}, {A,B}→{B}, {A,D}→{B,D}, {A,C}→{B,C}, {A,C,D}→{B,C,D}, {{A} →{A,B}, {A,B}→{A,B}, {A,D}→{A,B,D}, {A,C}→{A,B,C}, {A,C,D}→{A,B,C,D}, {B,C} →{D}, […], {A,C} →{D}, […]}**

Equivalence of Sets of Functional Dependencies

Two sets of functional dependencies F and G are equivalent if and only if

$$F^+ = G^+$$

We write

$$F \equiv G.$$

Attribute Closure

For a set S of attributes, we call the closure of S *(with respect to a set of functional dependencies F)*, noted S+, the maximum set of attributes such that S→S+ *(as a consequence of F)*

Closure of a Set of Attributes: Example

Consider the relation scheme `R = {A,B,C,D}` with the set of functional dependencies

$$F = \{\{A\}\rightarrow\{C\},\ \{B\}\rightarrow\{D\}\}.$$

- `{A}+ = {A,C}`
- `{B}+ = {B,D}`
- `{A,B}+ = {A,B,C,D}`
- `etc.`

write the 2^n-1 (or 2^n if we consider the empty set).

Closure of a Set of Attributes: Algorithm

- **Input:**
  - R a relation scheme
  - F a set of functional dependencies
  - X $\subset$ R
- **Output:**
  - X+ the closure of X w.r.t. F

Closure of a Set of Attributes: Algorithm

- **$X^{(0)}$:= X**
- **Repeat**
  - $X^{(i+1)} := X^{(i)} \cup A$, where A is the union of the sets Z of attributes such that there exist $Y \rightarrow Z$ in F, and $Y \subset X^{(i)}$
- **Until $X^{(i+1)} := X^{(i)}$**
- **Return $X^{(i+1)}$**

Closure of a Set of Attributes: Example

R = {A,B,C,D,E,G}

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, {D}→{E,G}, {B,E}→{C}, {C,G}→{B,D}, {C,E}→{A,G}}

X = {B,D}

Closure of a Set of Attributes: Example

R = {A,B,C,D,E,G}

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, {D}→{E,G}, {B,E}→{C}, {C,G}→{B,D}, {C,E}→{A,G}}

X = {B,D}

- $X^{(0)}$ = {B,D}
- {D}→{E,G}
- $X^{(1)}$ = {B,D,E,G}
- {B,E}→{C},
- $X^{(2)}$ = {B,C,D,E,G}
- {C,E}→{A,G}
- $X^{(3)}$ = $X^{(4)}$ =X+ = {A,B,C,D,E,G}

Testing Equivalence Based on Attribute Closures

- **Let R be a relational scheme**
- **Let F and G be two sets of functional dependencies on R**

- **for each (X $\rightarrow$ Y) in F**
  - compute $X^{+(G)}$
  - if Y $\not\subset$ $X^{+(G)}$ return false
- **for each (X $\rightarrow$ Y) in G**
  - Compute $X^{+(F)}$
  - if Y $\not\subset$ $X^{+(F)}$ return false
- **return true**

Example:
If F contains $\{A\} \rightarrow \{B,C\}$
but $\{A\}^{+(G)} \rightarrow \{A,B\}$,

then $\{A\} \rightarrow \{C\}$ is entailed by F
but not by G,

therefore F and G are not equivalent

Minimal Set of Dependencies

A set of dependencies F is minimal if and only if:

1. Every right-hand side is a single attribute
2. For no functional dependency $X \rightarrow A$ in F and proper subset Z of X is $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ equivalent to F
3. For no functional dependency $X \rightarrow A$ in F is the set $F - \{X \rightarrow A\}$ equivalent to F

Minimal Cover

A set of functional dependencies F is a <u>minimal cover</u> of a set of functional dependencies G if and only if

F is minimal

F is equivalent to G

(an **<u>extended minimal cover</u>** is obtained by undoing step 1 on a minimal cover)

Minimal Cover

Every set of functional dependencies has a minimal cover

There might be several different minimal cover of the same set

Minimal Cover

A set of dependencies F is minimal if and only if:

1. **Every right-hand side is a single attribute**
2. **For no functional dependency $X \rightarrow A$ in F and proper subset Z of X is $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ equivalent to F**
3. **For no functional dependency $X \rightarrow A$ in F is the set $F - \{X \rightarrow A\}$ equivalent to F**

Minimal Cover: Example

```
F = { {A,B}→{C},
{C}→{A},
{B,C}→{D},
{A,C,D}→{B},
{D}→{E,G},
{B,E}→{C},
{C,G}→{B,D},
{C,E}→{A,G}}
```

Minimal Cover: Example, Step (1)

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, <u>{D}→{E,G}</u>, {B,E}→{C}, <u>{C,G}→{B,D}</u>, <u>{C,E}→{A,G}</u>}

F′ = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, <u>{D}→{G}</u>, <u>{D}→{E}</u>, {B,E}→{C}, <u>{C,G}→{B}</u>, <u>{C,G}→{D}</u>, <u>{C,E}→{A}</u>, <u>{C,E}→{G}</u>}

Minimal Cover: Example, Step (2)

F′ = {{C}→{A}, <u>{C,E}→{A}</u>, <u>{A,C,D}→{B}</u>, {C,G}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

F′′ = {<u>{C}→{A}</u>, <u>{C,D}→{B}</u>, {C,G}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

Minimal Cover: Example, Step (2)

F′ = {{C}→{A}, {C,E}→{A}, <u>{A,C,D}→{B}</u>, {C,G}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

Proof

1. **We know that {D} →{G}.**
2. **Therefore {C,D} →{C,G} by Augmentation of (3) with {C}.**
3. **We know that {C,G} →{B}.**
4. **Therefore {C,D} →{B} by transitivity of (2) and (3).**
5. **Therefore {A,C,D} →{B} is redundant. It can be replaced by {C,D}→{B}.**

Minimal Cover: Example, Step (2)

F′ = {{C}→{A}, {C,E}→{A}, {A,C,D}→{B}, {C,G}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

Prove that {C,E}→{A} is redundant and that it can be replaced by {C}→{A}.

Minimal Cover: Example, Step (3)

F'' = {{C}→{A}, {C,D}→{B}, {C,G}→{B},
{A,B}→{C}, {B,E}→{C}, {B,C}→{D},
{C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

F''' = {{C}→{A}, {C,D}→{B}, {A,B}→{C},
{B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E},
{C,E}→{G}, {D}→{G}}

Minimal Cover: Example, Step (3)

F'' = {{C}→{A}, {C,D}→{B}, <u>{C,G}→{B}</u>, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, {D}→{E}, {C,E}→{G}, {D}→{G}}

Proof

1. **We know that {C,G} →{D}.**
2. **Therefore {C,G} →{C,D} by Augmentation of (1) with {C}.**
3. **We know that {C,D} →{B}.**
4. **Therefore {C,G} →{B} by transitivity of (2) and (3).**
5. **Therefore {C,G} →{B} is redundant.**

Extended Minimal Cover (Step 4)

F'' = {{C}→{A}, {C,D}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, <u>{D}→{E}</u>, {C,E}→{G}, <u>{D}→{G}</u>}

F''' = {{C}→{A}, {C,D}→{B}, {A,B}→{C}, {B,E}→{C}, {B,C}→{D}, {C,G}→{D}, <u>{D}→{E,G}</u>, {C,E}→{G}}

An <u>**extended minimal cover**</u> is obtained by undoing step 1 on a minimal cover.

Minimal Cover: Algorithm

At every step we transform the set of functional dependencies into an equivalent set.

We can apply steps (1), (2), (3) iteratively in various orders. However only (1) + (2) + (3) (+4) is guaranteed to lead to a (extended) minimal cover!

1. Simplify the right-hand side.
2. Simplify the left-hand side.
3. Simplify the entire set.
4. Regroup.

Credits

The content of this lecture is based on the book "Introduction to database Systems"
By
S. Bressan and B. Catania, McGraw Hill publisher

Images and clips used in this presentation are licensed from Microsoft Office Online Clipart and Media

For questions about the content of this course and about copyrights, please contact Stéphane Bressan

steph@nus.edu.sg