Dependencies

Stéphane Bressan

"[...] one can say that the relational model is almost devoid of semantics. [...] One approach to remedy this deficiency is to devise means to specify the missing semantics. These semantic specifications are called semantic or integrity constraints [...]. Of particular interest are the constraints called data dependencies or dependencies for short."
Fundementals of Dependency Theory
[TTC 1987],
by Moshe Y. Vardi

Readings

For those who are interested, the following book is free.

Abiteboul S.,Hull R., and Vianu V. “Foundations of Databases” (`http://webdam.inria.fr/Alice/pdfs/all.pdf`).

Dependencies

Definition

A dependency is a constraint on the instances of the relations in the schema of the following form.

$$\forall t_1 \in r_1 \cdots \forall t_n \in r_n$$
$$(\phi(t_1, \cdots, t_n) \to$$
$$(\exists t_{n+1} \in r_{n+1} \cdots \exists t_m \in r_m \ \psi(t_1, \cdots, t_n, t_{n+1}, \cdots, t_m)))$$

where $r_1, \cdots, r_m$ are instances of the relations in the schema and ψ and ϕ are logical statements about the tuples in the instances of the relations in the schema.

Definition

An instance s (a table or a set of tables) of a schema S (a relation or a set of relations) violates a dependency σ, if and only if it does not satisfy σ.
This is noted:

$$s \not\models \sigma$$

Definition

An instance s (a table or a set of tables) of a schema S (a relation or a set of relations) violates a set of dependencies Σ, if and only if it does not satisfy at least one of the dependencies in Σ.
This is noted:

$$(s \models \Sigma)$$

$$\leftrightarrow$$

$$(\forall \sigma \in \Sigma \ (s \models \sigma))$$

For a schema S, for a relation R, with a functional dependency σ, with a set of functional dependencies Σ, S with σ, R with Σ, refers to the set of valid instances of S, of R, with respect to σ, to the functional dependencies in Σ, respectively.

When we say that a dependency σ, a set of dependencies Σ, holds on a relation R, on a schema S, we only consider the valid instances of S, R, with σ, with Σ, respectively.

In general, constraints and dependencies can be used to maintain integrity, define normal forms, and to improve database design with respect to integrity and efficiency.

Functional Dependencies

employee

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

The table `employee` 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$ per month, etc.

Salaries are determined by the position.

This kind of business rule can be translated into an integrity constraint called a functional dependency. It is an integrity constraint. We write:

$$\{position\} \rightarrow \{salary\}$$

$$\{position\} \rightarrow \{salary\}$$

This means that we should not encounter a table in which two employees have the same position but different salaries.

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	4000

Definition

An instance r (a table) of a relation schema R satisfies the functional dependency σ: $X \rightarrow Y$ with $X \subset R$ and $Y \subset R$, if and only if if two tuples of r agree on their X-values, then they agree on their Y-values.

$$(r \models \sigma)$$

$$\leftrightarrow$$

$$(\forall t_1 \in r\ \forall t_2 \in r\ (t_1.X = t_2.X \rightarrow t1.Y = t2.Y))$$

$X \rightarrow Y$ reads: X functionally determines Y, X determines Y, Y is functionally dependent on X, or, more casually, X implies Y.

The following SQL query finds whether the constraint is violated.

```
SELECT *
FROM employee e1, employee e2
WHERE e1.position = e2.position AND e1.salary <> e2.salary;
```

The following SQL CHECK constraints implements the functional dependency.

```
CHECK NOT EXISTS (SELECT *
FROM employee e1, employee e2
WHERE e1.position = e2.position AND e1.salary <> e2.salary);
```

$R = \{A, B, C, D\}$

The following instance of R is valid for the functional dependency $\{A, B\} \rightarrow \{D\}$.

A	B	C	D
1	2	a	4
1	2	b	4
1	3	c	4

The following instance of R is violates the functional dependency $\{A, B\} \rightarrow \{D\}$.

A	B	C	D
1	2	a	4
1	2	b	3
1	3	c	4

$R = \{A, B, C, D\}$

The following (empty) instance of R is valid for the functional dependency $\{A, B\} \rightarrow \{D\}$. It is the smallest instance that does not violate the functional dependency.

A	B	C	D

The following instance of R is violates the functional dependency $\{A, B\} \rightarrow \{D\}$. It is the smallest instance that violates the functional dependency.

A	B	C	D
1	2	a	4
1	2	b	3

There are many possible functional dependencies in the case.

The following functional dependencies (among many others) hold in the case.

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

The following functional dependencies (among many others) do not hold in the case. Note that they may accidentally hold on a given instance (in particular, they always hold on the empty instance).

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

Definition

A functional dependency $X \rightarrow Y$ is trivial if and only if $Y \subset X$.

$R = \{A, B, C\}$

$\{A\} \rightarrow \{A\}$ is trivial.
$\{A, B\} \rightarrow \{A\}$ is trivial.
$\{A, B\} \rightarrow \emptyset$ is trivial.

$R = \{number, name, department, position, salary\}$

$\{number\} \rightarrow \{number\}$ is trivial. $\{name, department, salary\} \rightarrow \{name, salary\}$ is trivial.

Definition

A functional dependency $X \to Y$ is non-trivial if and only if $Y \not\subset X$.

$R = \{A, B, C\}$

$\{A\} \to \{B\}$ is non-trivial.
$\{A, C\} \to \{B, C\}$ is non-trivial.

$R = \{number, name, department, position, salary\}$

$\{position\} \rightarrow \{salary\}$ is non-trivial.
$\{number\} \rightarrow \{name\}$ is non-trivial.
$\{number\} \rightarrow \{number, name, department, position\}$ is non-trivial.
$\{number\} \rightarrow \{number, name, department, position, salary\}$ is non-trivial.

Definition

A functional dependency $X \to Y$ is completely non-trivial if and only if $Y \neq \emptyset$ and $Y \cap X = \emptyset$.

$R = \{A, B, C\}$

$\{A\} \to \{B\}$ is completely non-trivial.
$\{A, C\} \to \{B, C\}$ is not completely non-trivial.

$R = \{number, name, department, position, salary\}$

$\{position\} \rightarrow \{salary\}$ is completely non-trivial.
$\{number\} \rightarrow \{name\}$ is completely non-trivial.
$\{number\} \rightarrow \{name, department, position\}$ is completely non-trivial.
$\{number\} \rightarrow \{name, department, position, salary\}$ is completely non-trivial.

A superkey is a set of attributes of a relation whose knowledge determines the value of the entire t-uple.

Definition

Let R be a relation. Let $S \subset R$ be a set of attributes of R. S is a superkey of R if and only if $S \rightarrow R$.

A candidate key is a minimal superkey (for inclusion).

Definition

Let R be a relation. Let $S \subset R$ be a set of attributes of R. S is a candidate of R if and only if $S \rightarrow R$ and for all $T \subset S$, $T \neq S$, T is not a superkey of R.

The primary key is the candidate key that the designer prefers or the candidate key if there is only one.

$\{number\}$ is a superkey of the table because
$\{number\} \rightarrow \{number, name, department, position, salary, position, salary\}$ holds.

$\{number\}$ is a candidate key of the table because there is no subset S of the set $\{number\}$ such that
$S \rightarrow \{number, name, department, position, salary, position, salary\}$ holds.

$\{number, name\}$ is a superkey of the table because
$\{number, name\} \rightarrow \{number, name, department, position, salary\}$ holds.

$\{number, name\}$ is not a candidate key of the table because
$\{number\} \rightarrow \{number, name, department, position, salary\}$ holds.

Definition

Let Σ be a set of functional dependencies on a relation schema R. A prime attribute is an attribute that is appears in some candidate key of R with Σ (otherwise it is called a non-prime attribute).

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

The candidate keys of R with Σ are $\{A, B\}$ and $\{C\}$.
A is a prime attribute of R with Σ.
B is a prime attribute of R with Σ.
C is a prime attribute of R with Σ.
D is a non-prime attribute of R with Σ.

Definition

Let Σ be a set of functional dependencies of a relation schema R. The closure of Σ, noted Σ^+, is the set of all functional dependencies logically entailed by the functional dependencies in Σ.

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

$\Sigma^+ = \{\{A\} \to \{B\}, \{C\} \to \{A\}, \{A\} \to \{A\}, \{D\} \to \{D\}, \{A, B\} \to \{A\}, \{A, C\} \to \{B, C\}, \{A, D\} \to \{B\}, \{C\} \to \{B\}, \cdots\}$

Find

- a trivial functional dependency in Σ^+.
- a non-trivial but not completely non-trivial functional dependency in Σ^+.
- a completely non-trivial functional dependency in Σ^+.

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

$\{A, D\} \rightarrow \{B, C\} \in \Sigma^+?$

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

$\{A, D\} \rightarrow \{B, C\} \in \Sigma^+?$

Armed with only the definition of functional dependency, the problems of computing Σ^+ and of testing membership to Σ^+ are daunting tasks.

Definition

Two sets of of functional dependencies Σ and Σ' are equivalent if and only if they have the same closure.

$$\Sigma^+ = \Sigma'^+$$

Are $\Sigma = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{C\} \rightarrow \{A\}\}$ and $\Sigma' = \{\{C\} \rightarrow \{A, B\}, \{A\} \rightarrow \{B, C\}, \{B\} \rightarrow \{A\}, \{A, B\} \rightarrow \{C\}\}$ equivalent?

The answer is yes, but we need more tools to check that efficiently (without computing Σ^+ and Σ'^+).

Definition

Let Σ be a set of functional dependencies of a relation schema R. The closure of a set of attributes $S \subset R$, noted S^+, is the set of all attributes that are functionally dependent on S.

$$S^+ = \{A \in R \mid \exists(S \rightarrow \{A\}) \in \Sigma^+\}$$

The closure of a set of attributes can be computed by the fix-point iterative application of the functional dependencies as production rules.

Algorithm 1: Attribute Closure Algorithm

input : S, Σ
output: S^+
begin
  $\Omega := \Sigma$; `// Ω stands for‘‘unused’’`
  $\Gamma := S$; `// Γ stands for ‘‘closure’’`
  while $X \rightarrow Y \in \Omega$ *and* $X \subset \Gamma$ **do**
    $\Omega := \Omega - \{X \rightarrow Y\}$;
    $\Gamma := \Gamma \cup Y$;
  return Γ;

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

Compute $\{C\}^+$ using Algorithm 1.

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

1. $\Omega = \{\{A\} \to \{B\}, \{C\} \to \{A\}\}$
   $\Gamma = \{C\}^+$
2. use $\{C\} \to \{A\}$ $(\{C\} \subset \Gamma)$
   $\Omega = \{\{A\} \to \{B\}\}$
   $\Gamma = \{C\} \cup \{A\} = \{C, A\}$
3. use $\{A\} \to \{B\}$ $(\{A\} \subset \Gamma)$
   $\Omega = \emptyset$
   $\Gamma = \{C, A\} \cup \{B\} = \{C, A, B\}$
4. return $\Gamma = \{C, A, B\}$

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

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

We compute $\{C, D\}^+$.

We have $\{C, D\}$, therefore $C \in \{C, D\}^+$ and $D \in \{C, D\}^+$.
We know that $\{C\} \rightarrow \{A\}$ and $\{C\} \subset \{C, D\}^+$, therefore $A \in \{C, D\}^+$.
We know that $\{A\} \rightarrow \{B\}$ and $\{A\} \subset \{C, D\}^+$, therefore $B \in \{C, D\}^+$.
Therefore $\{C, D\}^+ = \{A, B, C, D\}$

How to prove that the algorithm is sound and complete?

Definition

A set Σ of functional dependencies is minimal if and only if:

- The right hand-side of every functional dependency in Σ is minimal. Namely, every functional dependency is of the form $X \rightarrow \{A\}$.
- The left hand-side of every functional dependency is minimal. Namely, for every functional dependency in Σ of the form $X \rightarrow \{A\}$ there is no functional dependency $Y \rightarrow \{A\}$ in Σ^+ such that Y is a proper subset of X.
- The set itself is minimal. Namely, non of the functional dependency in Σ can derived from the other functional dependencies in Σ.

Definition

A minimal cover of a set of functional dependencies Σ is set of functional dependencies Σ' that is both minimal and equivalent to Σ.

An algorithm for the computation of the minimal cover Σ''' of a set of functional dependencies Σ has the following three steps.

1. Simplify (minimise) the right hand-side of every functional dependency in Σ to get Σ'.
2. Simplify (minimise) the left hand-side of every functional dependency in Σ' to get Σ''.
3. Simplify (minimise) the set Σ'' to get Σ'''.

The three steps have to be done in this order.

An algorithm for the computation of the compact minimal cover Σ'''' of a set of functional dependencies Σ has the following four steps:

1. Simplify (minimise) the right hand-side of every functional dependency in Σ to get Σ'.
2. Simplify (minimise) the left hand-side of every functional dependency in Σ' to get Σ''.
3. Simplify (minimise) the set Σ'' to get Σ'''.
4. Regroup all the functional dependencies with the same left-hand side in Σ''' to get Σ'''' (reverse of Step 1).

The four steps have to be done in this order.

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

Compute the attribute covers.

Find all the candidate keys.

Find a minimal cover.

Find a compact minimal cover.

Compute all the singletons covers.

$$\{A\}^+ = \{A\}$$
$$\{B\}^+ = \{B, C, D, E\}$$
$$\{C\}^+ = \{B, C, D, E\}$$
$$\{D\}^+ = \{D\}$$
$$\{E\}^+ = \{E\}$$

Compute all the pairs covers.

$\{A, B\}^+ = \{A, B, C, D, E\}$
$\{A, C\}^+ = \{A, B, C, D, E\}$
$\{A, D\}^+ = \{A, D\}$
$\{A, E\}^+ = \{A, E\}$
$\{B, C\}^+ = \{B, C, D, E\}$
$\{B, D\}^+ = \{B, C, D, E\}$
$\{B, E\}^+ = \{B, C, D, E\}$
$\{C, D\}^+ = \{B, C, D, E\}$
$\{C, E\}^+ = \{B, C, D, E\}$
$\{D, E\}^+ = \{D, E\}$

Any set of attributes containing $\{A, B\}$ or $\{A, C\}$ is a superkey. $\{A, B\}$ and $\{A, C\}$ are candidate keys.

Compute all the remaining triplet covers.

$$\{A, D, E\}^+ = \{A, D, E\}$$
$$\{B, D, E\}^+ = \{B, C, D, E\}$$
$$\{C, D, E\}^+ = \{B, C, D, E\}$$
$$\{B, C, E\}^+ = \{B, C, D, E\}$$
$$\{B, C, D\}^+ = \{B, C, D, E\}$$

Compute all the remaining quadruplet covers.

$$\{B, C, D, E\}^+ = \{B, C, D, E\}$$

We know that all quintuplet covers are superkeys.

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

We compute a minimal cover.

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

We simplify the right-hand sides (easy).

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

We simplify the left-hand sides (very difficult).

$\Sigma'' = \{$
$\cancel{\{A, B\} \rightarrow \{C\}}$, (is replaced with $\{B\} \rightarrow \{C\}$)
$\cancel{\{A, B\} \rightarrow \{D\}}$, (is replaced with) $\{B\} \rightarrow \{D\}$,
$\cancel{\{A, B\} \rightarrow \{E\}}$, (is replaced with $\{B\} \rightarrow \{E\}$)
$\cancel{\{A, C\} \rightarrow \{B\}}$, (is replaced with $\{C\} \rightarrow \{B\}$),
$\cancel{\{A, C\} \rightarrow \{D\}}$, (is replaced with $\{C\} \rightarrow \{D\}$),
$\cancel{\{A, C\} \rightarrow \{E\}}$, (is replaced with $\{C\} \rightarrow \{E\}$),
$\{B\} \rightarrow \{C\}$,
$\{C\} \rightarrow \{B\}$,
$\{C\} \rightarrow \{D\}$,
$\{B\} \rightarrow \{E\}$,
$\{C\} \rightarrow \{E\}\}$

We simplify the left-hand sides (very difficult).

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

We simplify the set itself by removing functional dependencies that can be derived from the others. (difficult).

$\Sigma''' = \{$
$\cancel{\{B\} \rightarrow \{D\}}$, (it can be obtained from $\{B\} \rightarrow \{C\}$ and $\{C\} \rightarrow \{D\}$)
$\{B\} \rightarrow \{C\}$,
$\{C\} \rightarrow \{B\}$,
$\{C\} \rightarrow \{D\}$,
$\cancel{\{B\} \rightarrow \{E\}}$, (it can be obtained from $\{B\} \rightarrow \{C\}$ and $\{C\} \rightarrow \{E\}$)
$\{C\} \rightarrow \{E\}\}$

Σ''' is a minimal cover of Σ.

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

We could reach different minimal covers by considering the constraints in a different order.

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

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

- The algorithm always finds a minimal cover (how to prove it? - The Armstrong Axioms -) but some minimal covers may be unreachable with the algorithm.
- For instance, if Σ is already a minimal cover, the algorithm cannot reach a different minimal cover even if it exists.
- To be guaranteed to reach all minimal covers with the algorithm one needs to start from Σ^+.

Example

We compute a compact minimal cover by regrouping the constraints with the same left-hand side (easy).

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

The other compact minimal covers are as follows.

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

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

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

Definition

Let R be a set of attributes. The following inference rules are the Armstrong Axioms.

- Reflexivity
  $\forall X \subset R\ \forall Y \subset R\ ((Y \subset X) \Rightarrow (X \rightarrow Y))$
- Augmentation
  $\forall X \subset R\ \forall Y \subset R\ \forall Z \subset R\ ((X \rightarrow Y) \Rightarrow (X \cup Z \rightarrow Y \cup Z))$
- Transitivity
  $\forall X \subset R\ \forall Y \subset R\ \forall Z \subset R\ ((X \rightarrow Y \wedge Y \rightarrow Z) \Rightarrow (X \rightarrow Z))$

Technically, the Armstrong Axioms are not axioms but inference rules.

Theorem

The Reflexivity inference rule is sound (correct, valid).

Theorem

The Augmentation inference rule is sound.

Theorem

The Transitivity inference rule is sound.

Proof of Soundness for Transitivity.

1. Let Σ be a set of functional dependencies on a relation schema R. Let $X \rightarrow Y$ and $Y \rightarrow Z$ be in Σ.
2. We know that for all valid instance r of R with Σ $(\forall t_1 \in r\ \forall t_2 \in r\ (t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]))$ by definition of a functional dependency.
3. We know that for all valid instance r of R with Σ $(\forall t_1 \in r\ \forall t_2 \in r\ (t_1[Y] = t_2[Y] \Rightarrow t_1[Z] = t_2[Z]))$ by definition of a functional dependency.
4. Therefore for all valid instance r of R with Σ $(\forall t_1 \in r\ \forall t_2 \in r\ (t_1[X] = t_2[X] \Rightarrow t_1[Z] = t_2[Z]))$ by definition of a functional dependency.
5. Therefore $X \rightarrow Z \in \Sigma^+$.
6. Q.E.D.

$\square$

Theorem

The Armstrong Axioms are complete.

Proof Sketch

1. We prove that for any set of attribute $S \in R$, then $S \to S^+$ can be derived from the Armstrong Axioms.
   1.1 This is recursively true because every step of the attribute closure algorithm is of the form:
      1.1.1 $S \to S^i$ and
      1.1.2 $X \to Y$
      1.1.3 with $X \subset S^i$.
   1.2 Therefore $S^i \to X$ by Reflexivity with (1.1.3).
   1.3 Therefore $S^i \to X \cup S^i$ by Augmentation of (1.2) with S^i.
   1.4 Therefore $X \cup S^i \to Y \cup S^i$ by Augmentation of (1.1.2) with S^i.
   1.5 Therefore $S \to S^{i+1}$, where $S^{i+1} = S_i \cup Y$, by Transitivity of (1.3) and (1.4).
   1.6 Q.E.D

...

Proof Sketch

1. We prove that if $X \rightarrow Y \in \Sigma^+$, then it can be derived from $X \rightarrow X^+$.
   1.1 We know that $Y \subset X^+$ by property of the attribute closure.
   1.2 Therefore $X^+ \rightarrow Y$ by Reflexivity.
   1.3 Therefore $X \rightarrow Y$ by Transitivity of $X \rightarrow X^+$ and $X^+ \rightarrow Y$.
   1.4 Q.E.D
2. Q.E.D.

Theorem

Let R be a relation with the set of functional dependencies Σ. We can compute Σ^+ by applying the Armstrong Axioms until no new functional dependency is produced.

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

$\Sigma^+ = \{\{A\} \rightarrow \{B\}, \{C\} \rightarrow \{A\}, \{A\} \rightarrow \{A\}, \{D\} \rightarrow \{D\}, \{A, B\} \rightarrow \{A\}, \{A, C\} \rightarrow \{B, C\}, \{A, D\} \rightarrow \{B\}, \{C\} \rightarrow \{B\}, \cdots\}$

Theorem

Weak Reflexivity is *sound*.

$\forall X \subset R \ (X \rightarrow \emptyset)$

Proof.

1. Let R be a relation schema.
2. Let $X \subset R$.
3. We know that $\emptyset \subset X$.
4. Therefore $X \to \emptyset$ by Reflexivity.
5. Q.E.D

$\square$

Theorem

Weak Augmentation is *sound.*

$$\forall X \subset R \; \forall Y \subset R \; \forall Z \subset R \; ((X \rightarrow Y) \Rightarrow (X \cup Z \rightarrow Y))$$

Proof.

1. Let R be a relation schema.
2. Let $X \subset R$.
3. Let $Y \subset R$.
4. Let $Z \subset R$.
5. Let $X \rightarrow Y$
6. We know that $X \subset X \cup Z$.
7. Therefore $X \cup Z \rightarrow X$ by Reflexivity.
8. Therefore $X \cup Z \rightarrow Y$ by Transitivity of (7) and (5).
9. Q.E.D

$\square$

Multi-valued Dependencies

Catalog		
Course	Lecturer	Text
Programming	{Tan CK, Lee SL}	{The Art of Programming, Java}
Maths	{Tan CK}	{Java}
...		

The Catalog relation is a nested relation.
It is in Non-First Normal Form (NF^2).

The indicated courses are taught by all of the indicated teachers, and use all the indicated text books.

The course determines the set of lecturers.
The course determines the set of texts.

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
...		

We transform the Catalog relation into First Normal Form (1NF).
What anomalies?

The dependencies cannot be captured by functional dependencies. They are multi-valued dependencies.

Unlike functionl dependencies, multi-valued dependencies are relation sensitive.

Catalog			
Course	Lecturer	Text	Percentage
Programming	Tan CK	The Art of Programming	30
Programming	Tan CK	Java	40
Programming	Lee SL	The Art of Programming	90
Programming	Lee SL	Java	10
DS and Alg.	Tan CK	Java	100
...			

A teacher teaches a course and uses a percentage of a text book.

The previous multi-valued dependencies do not hold anymore.

Definition

An instance r of a relation schema R satisfies the multi-valued dependency σ: $X \twoheadrightarrow Y$, X multi-determines Y or Y is multi-dependent on X, with $X \subset R$, $Y \subset R$ and $X \cap Y = \emptyset$ if and only if , for $Z = R - (X \cup Y)$, two tuples of r agree on their X-value, then there exists a t-uple of r that agrees with the first tuple on the X- and Y-value and with the second on the Z-value.

$$(r \models \sigma)$$

$$\leftrightarrow$$

$$(\forall t_1 \in r \ \forall t_2 \in r \ (t_1.X = t_2.X \rightarrow$$

$$\exists t_3 \in r \ (t_3.X = t_1.X \wedge t_3.Y = t_1.Y \wedge t_3.Z = t_2.Z)))$$

Each X-value in r is consistently associated with one set of Y-value in r.

With a multi-valued dependency $X \twoheadrightarrow Y$, the presence of two different t-uples with the same X-values implies the presence of two additional t-uples with the Y-values (When the Z-values are different and $Z \neq \emptyset$).

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

$$\{Course\} \twoheadrightarrow \{Lecturer\}$$

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
...		

$$\{\text{Course}\} \twoheadrightarrow \{\text{Teacher}\}$$

$$\{\text{Course}\} \twoheadrightarrow \{\text{Text}\}$$

Definition

A multi-valued dependency $X \twoheadrightarrow Y$ is trivial if and only if

1. $Y = R - X$ or
2. $Y \subset X$.

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
...		

$$\{Text\} \twoheadrightarrow \{Course, Lecturer\}$$

Theorem

The Complementation inference rule is sound.
$\forall X \subset R \; \forall Y \subset R$

$$(X \twoheadrightarrow Y) \Rightarrow (X \twoheadrightarrow R - X - Y)$$

Theorem

The Augmentation inference rule is sound.
$\forall X \subset R \; \forall Y \subset R \; \forall V \subset R \; \forall W \subset R$

$$((X \twoheadrightarrow Y) \wedge (V \subset W)) \Rightarrow (X \cup W \twoheadrightarrow Y \cup V)$$

Theorem

The Transitivity inference rule is sound.
$\forall X \subset R\ \forall Y \subset R\ \forall Z \subset R$

$$((X \twoheadrightarrow Y) \wedge (Y \twoheadrightarrow Z)) \Rightarrow (X \twoheadrightarrow Z - Y)$$

Theorem

The Replication (Promotion) inference rule is sound.
$\forall X \subset R\ \forall Y \subset R$

$$(X \rightarrow Y) \Rightarrow (X \twoheadrightarrow Y)$$

Functional dependencies are a special case of multi-valued dependencies.

Theorem

The Coalescence inference rule is sound.
$\forall X \subset R \; \forall Y \subset R \; \forall Z \subset R \; \forall W \subset R$

$$(X \twoheadrightarrow Y) \wedge (W \rightarrow Z) \wedge (Z \subset Y) \wedge (W \cap Y = \emptyset)) \Rightarrow (X \rightarrow Z)$$

Theorem

Complementation, Augmentation, Transitivity, Replication and Coalescence, with the Armstrong Axioms form a sound and complete system for functional and multi-valued dependencies.

Theorem

The Multi-valued Union inference rule is sound.
$\forall X \subset R \; \forall Y \subset R \; \forall Z \subset R$

$$((X \twoheadrightarrow Y) \wedge (X \twoheadrightarrow Z)) \Rightarrow (X \twoheadrightarrow Y \cup Z))$$

Theorem

The Multi-valued Intersection inference rule is sound.
$\forall X \subset R \; \forall Y \subset R \; \forall Z \subset R$

$$((X \twoheadrightarrow Y) \wedge (X \twoheadrightarrow Z)) \Rightarrow (X \twoheadrightarrow Y \cap Z)$$

Theorem

The Multi-valued Difference inference rule is sound.
$\forall X \subset R\ \forall Y \subset R\ \forall Z \subset R$

$$((X \twoheadrightarrow Y) \land (X \twoheadrightarrow Z)) \Rightarrow (X \twoheadrightarrow Y - Z)$$

There is no decomposition rule.

$$\cancel{(X \twoheadrightarrow Y \cup Z\}) \Rightarrow (X \twoheadrightarrow Y)}$$

How can we reason about functional and multi-valued dependencies?