Relational Design Theory

CSE462 Database Concepts

Demian Lessa

Department of Computer Science and Engineering
State University of New York, Buffalo

February 25 – March 11, 2011

Outline

Overview

How does one design a relational schema?

- From requirements, high-level notations, etc.
- Initial schemas tend to combine too much into one relation.
  - This creates redundancy in the data.
  - Redundancies can be eliminated systematically.
- Relational theory.
  - Provides the foundation for designing good relational schemas.
  - Involves the study of dependencies invoving data elements.
  - Permits us to distinguish good from bad schemas.
  - Gives us the means to fix flawed designs.

Overview

- Certain dependencies in relation schemas may cause problems.
  - Often referred to as anomalies.
- Data dependencies.
  - Functional dependencies generalize the notion of key.
  - Multivalued dependencies identify sets of independent attributes.
- Dependencies are used to define normal forms for relation schemas.
- Normal forms are used in a process called normalization.
  - Decompose relations into two or more relations to remove anomalies.
  - Each decomposition targets a particular normal form.

Next...

Functional Dependencies

A functional dependency (FD) on a relation R is a statement of the form:

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

where $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ are attributes of R. We say that $A_1, A_2, \ldots, A_n$ functionally determines $B_1, B_2, \ldots, B_m$.

Functional Dependencies

A functional dependency (FD) on a relation R is a statement of the form:

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

where $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ are attributes of R. We say that $A_1, A_2, \ldots, A_n$ functionally determines $B_1, B_2, \ldots, B_m$.

- Let t, u be tuples in some R instance. If t and u agree on their $A_1, A_2, \ldots, A_n$ values, they must agree on their $B_1, B_2, \ldots, B_m$ values.

Functional Dependencies

A functional dependency (FD) on a relation R is a statement of the form:

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

where $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ are attributes of R. We say that $A_1, A_2, \ldots, A_n$ functionally determines $B_1, B_2, \ldots, B_m$.

- Let t, u be tuples in some R instance. If t and u agree on their $A_1, A_2, \ldots, A_n$ values, they must agree on their $B_1, B_2, \ldots, B_m$ values.
- If some FD F is true for every instance of R, then we say R satisfies F. You can think of F as a constraint on R.

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Intuitively, what seems to be wrong with `Movies1`?

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Intuitively, what seems to be wrong with `Movies1`?
- What does the FD below mean? Does it hold?

  `title year` $\rightarrow$ `length genre studioName`

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Intuitively, what seems to be wrong with `Movies1`?
- What does the FD below mean? Does it hold?
  `title year` $\rightarrow$ `length genre studioName`
- How about this other FD?
  `title year` $\rightarrow$ `starName`

Example: Functional Dependencies

Observation

Even if we had one tuple for each of the movies, the FD

$$\texttt{title year} \rightarrow \texttt{starName}$$

would still be false: by definition, a FD is a property of all possible instances of the relation, not just this particular one.

Key

A set $\{A_1, A_2, \ldots, A_n\}$ of attributes is a key for a relation R if:

1. $A_1, A_2, \ldots, A_n$ functionally determine all other attributes of R, that is, no two distinct tuples in R agree on all of $A_1, A_2, \ldots, A_n$.
2. No proper subset of $\{A_1, A_2, \ldots, A_n\}$ satisfies (1).

Observations:

- A relation may have more than one key.
  - One key is normally designated as the primary key.
  - In the theory of FDs, primary keys are no different than other keys.
- A superkey is a set of attributes that contains a key.
  - Every key is a superkey.
  - Not every superkey is a key.

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`?

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`? `{title, year, starName}`
  - Prop 1:

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName)`.

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`? `{title, year, starName}`
  - Prop 1: assume $t_1 \neq t_2$ agree on `{title, year}`; they must agree on `{length, genre, studioName}`; if they agree on `starName` then $t_1 = t_2$.
  - Prop 2:

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`? `{title, year, starName}`
  - Prop 1: assume $t_1 \neq t_2$ agree on `{title, year}`; they must agree on `{length, genre, studioName}`; if they agree on `starName` then $t_1 = t_2$.
  - Prop 2: is `{title, year}` a key?

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`? `{title, year, starName}`
  - Prop 1: assume $t_1 \neq t_2$ agree on `{title, year}`; they must agree on `{length, genre, studioName}`; if they agree on `starName` then $t_1 = t_2$.
  - Prop 2: is `{title, year}` a key? `{year, starName}`?

Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName)`.

- Recall: `title year` $\rightarrow$ `length genre studioName`
- What is the key for `Movies1`? `{title, year, starName}`
  - Prop 1: assume $t_1 \neq t_2$ agree on `{title, year}`; they must agree on `{length, genre, studioName}`; if they agree on `starName` then $t_1 = t_2$.
  - Prop 2: is `{title, year}` a key? `{year, starName}`? how about `{title, starName}`?

Required

- Read section 3.1 from chapter #3.
- Go over problems 3.1.1 and 3.1.3.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;
3. thus, $t_1 = (a, b, c_1)$ and $t_2 = (a, b, c_2)$, where $b = b_1 = b_2$;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;
3. thus, $t_1 = (a, b, c_1)$ and $t_2 = (a, b, c_2)$, where $b = b_1 = b_2$;
4. from `B` $\rightarrow$ `C`, it follows that t_1 and t_2 must agree on `C`;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;
3. thus, $t_1 = (a, b, c_1)$ and $t_2 = (a, b, c_2)$, where $b = b_1 = b_2$;
4. from `B` $\rightarrow$ `C`, it follows that t_1 and t_2 must agree on `C`;
5. thus, $t_1 = (a, b, c)$ and $t_2 = (a, b, c)$, where $c = c_1 = c_2$;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;
3. thus, $t_1 = (a, b, c_1)$ and $t_2 = (a, b, c_2)$, where $b = b_1 = b_2$;
4. from `B` $\rightarrow$ `C`, it follows that t_1 and t_2 must agree on `C`;
5. thus, $t_1 = (a, b, c)$ and $t_2 = (a, b, c)$, where $c = c_1 = c_2$;
6. therefore, if two tuples in `R` agree on `A`, they must agree on `C`;

Example: Reasoning about FDs

Assume that relation `R(A, B, C)` satisfies FDs `A` $\rightarrow$ `B` and `B` $\rightarrow$ `C`. Does it satisfy `A` $\rightarrow$ `C`?

Show that if two tuples agree on `A` and `B`, they must agree on `C`.

1. let $t_1 = (a, b_1, c_1)$ and $t_2 = (a, b_2, c_2)$ agree on `A`;
2. from `A` $\rightarrow$ `B`, it follows that t_1 and t_2 must agree on `B`;
3. thus, $t_1 = (a, b, c_1)$ and $t_2 = (a, b, c_2)$, where $b = b_1 = b_2$;
4. from `B` $\rightarrow$ `C`, it follows that t_1 and t_2 must agree on `C`;
5. thus, $t_1 = (a, b, c)$ and $t_2 = (a, b, c)$, where $c = c_1 = c_2$;
6. therefore, if two tuples in `R` agree on `A`, they must agree on `C`;
7. we conclude that R satisfies `A` $\rightarrow$ `C`.

Reasoning About FDs

FDs may be presented in different yet equivalent ways.

- Two sets of FDs $\mathcal{S}$ and $\mathcal{T}$ are equivalent if the sets of relation instances satisfying $\mathcal{S}$ is exactly the same as those satisfying $\mathcal{T}$.
- A set of FDs $\mathcal{S}$ follows from a set of FDs $\mathcal{T}$ if every relation instance that satisfies all FDs in $\mathcal{T}$ also satisfies all FDs in $\mathcal{S}$.
- $\mathcal{S}$ and $\mathcal{T}$ are equivalent if and only if $\mathcal{S}$ and $\mathcal{T}$ follow from each other.

Splitting Rule

We can replace an FD

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

by the set of FDs

$$A_1 A_2 \cdots A_n \rightarrow B_1$$

$$\cdots$$

$$A_1 A_2 \cdots A_n \rightarrow B_m$$

We may replace a FD with a set of FDs having the same `lhs` and only one of the attributes on the `rhs`.

Combining Rule

We can replace a set of FDs

$$A_1 A_2 \cdots A_n \rightarrow B_1$$

$$\cdots$$

$$A_1 A_2 \cdots A_n \rightarrow B_m$$

by the single FD

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

We may replace a set of FDs having a common `lhs` by one FD with the same `lhs` and all `rhs` attributes combined into one set.

Splitting and Combining Rules

In both rules, the input and output FD sets are equivalent. But why do these rules hold?

- Assume two tuples agree in $A_1, A_2, \ldots, A_n$.
- Single FD: "then, the tuples agree in all $B_1, B_2, \ldots, B_m$."
- Set of FDs: "then, the tuples agree in B_1 and they agree in B_2, ..., and they agree in B_m."
- These conclusions are exactly the same thing!

Splitting and Combining Rules

In both rules, the input and output FD sets are equivalent. But why do these rules hold?

- Assume two tuples agree in $A_1, A_2, \ldots, A_n$.
- Single FD: "then, the tuples agree in all $B_1, B_2, \ldots, B_m$."
- Set of FDs: "then, the tuples agree in B_1 and they agree in B_2, ..., and they agree in B_m."
- These conclusions are exactly the same thing!

Why do you suppose splitting cannot be applied to the left side?

Exmaple: Splitting and Combining Rules

The FD:

$$\texttt{title year} \rightarrow \texttt{length genre studioName}$$

is equivalent to the set of FDs:

$$\texttt{title year} \rightarrow \texttt{length}$$
$$\texttt{title year} \rightarrow \texttt{genre}$$
$$\texttt{title year} \rightarrow \texttt{studioName}$$

Exmaple: Splitting and Combining Rules

The FD:

$$\texttt{title year} \rightarrow \texttt{length}$$

is not equivalent to the set of FDs:

$$\texttt{title} \rightarrow \texttt{length}$$
$$\texttt{year} \rightarrow \texttt{length}$$

Trivial FDs

A constraint of any kind on a relation is said to be trivial if it holds for every instance of the relation. An FD

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

is trivial if

$$\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$$

That is, the attributes on the `rhs` form a subset of the ones on the `lhs`. Every trivial FD holds in every relation. Thus, trivial FDs may be assumed without further justification.

Trivial-Dependency Rule

If some (but not all) of the attributes on the `rhs` of an FD also appear on the `lhs`, those that appear on both sides may be dropped from the `rhs`. That is,

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

is equivalent to

$$A_1 A_2 \cdots A_n \rightarrow C_1 C_2 \ldots C_k$$

where $\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} \setminus \{A_1, A_2, \ldots, A_n\}$.

It should be clear that we can prove this rule by first splitting the FD and then combining the resulting non-trivial FDs.

Exmaple: Trivial FDs

The FDs below are trivial:

$$\texttt{title year} \rightarrow \texttt{title}$$

$$\texttt{title year} \rightarrow \texttt{year}$$

However, the FD below is not trivial:

$$\texttt{title year} \rightarrow \texttt{title length}$$

but may be simplified using the trivial-dependency rule:

$$\texttt{title year} \rightarrow \texttt{length}$$

Closure of Attributes

Set Closure (Mathematics)

In set theory, we say that a set is closed under some operation if performing that operation on members of the set always produces a member of the set.

Closure of Attributes

Definition: Closure of Attributes

Given a set of attributes $\{A_1, \ldots, A_n\}$ and a set of FDs $\mathcal{S}$, the closure of $\{A_1, \ldots, A_n\}$ under the FDs in $\mathcal{S}$ is the set of attributes **B** such that every relation that satisfies all the FDs in $\mathcal{S}$ also satisfies $\{A_1, \ldots, A_n\} \rightarrow \mathbf{B}$.

The closure of $\{A_1, \ldots, A_n\}$ is denoted $\{A_1, \ldots, A_n\}^+$. Note that every $A_i \in \{A_1, \ldots, A_n\}$ is trivially a member of $\{A_1, \ldots, A_n\}^+$.

Applications:

- $A_1 \cdots A_n \rightarrow \mathbf{B}$ if and only if $\mathbf{B} \subseteq \{A_1, \ldots, A_n\}^+$.
- $\{A_1, \ldots, A_n\}$ is a superkey for a relation if and only if $\{A_1, \ldots, A_n\}^+$ is the set of all relation attributes. (Why not a key?)

Closure of Attributes

Algorithm

INPUT: Set of attributes $\{A_1, \ldots, A_n\}$ and set of FDs $\mathcal{S}$.

OUTPUT: The closure $\{A_1, \ldots, A_n\}^+$.

1. Split the FDs of $\mathcal{S}$ so that each FD has a single attribute on their `rhs`.
2. Let X be the output set and initialize it to $\{A_1, \ldots, A_n\}$.
3. Search for some FD $B_1 \cdots B_m \rightarrow C$ where each $B_i \in X$ but $C \notin X$. Add C to X and repeat until no FD is found.
4. Output X.

Note: X grows every time a new FD is found in step (3). Eventually, step (3) does not find a new FD so the algorithm must terminate and output X.

Required

- Go over examples 3.8 and 3.9.
- Go over the soundness and correctness proofs for the closure algorithm in 3.2.5.

Transitivity Rule

If $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$ and $B_1 B_2 \cdots B_m \rightarrow C_1 C_2 \cdots C_k$ hold in relation R, then $A_1 A_2 \cdots A_n \rightarrow C_1 C_2 \cdots C_k$ also holds in R.

Sketch.

- Compute $\{A_1, A_2, \ldots, A_n\}^+$ with respect to the given FDs.
- $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$ implies $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}^+$.
- Thus, we can use $B_1 B_2 \cdots B_m \rightarrow C_1 C_2 \cdots C_k$.
- $B_1 B_2 \cdots B_m \rightarrow C_1 C_2 \cdots C_k$ implies $\{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+$.
- It follows that $A_1 A_2 \cdots A_n \rightarrow C_1 C_2 \cdots C_k$.

Example: Transitivity Rule

title	year	length	genre	studioName	studioAddr
Star Wars	1977	124	SciFi	Fox	Hollywood
Eight Below	2005	120	Drama	DIsney	Buena Vista
Wayne's World	1992	95	Comedy	Paramount	Hollywood

Table: `Movies2(title, year, length, genre, studioName, studioAddr).`

- Assume that the following FDs hold:

  `title year` $\rightarrow$ `genre length studioName`

  `studioName` $\rightarrow$ `studioAddr`

- By transitivity, the following FD is derived:

Example: Transitivity Rule

title	year	length	genre	studioName	studioAddr
Star Wars	1977	124	SciFi	Fox	Hollywood
Eight Below	2005	120	Drama	DIsney	Buena Vista
Wayne's World	1992	95	Comedy	Paramount	Hollywood

Table: `Movies2(title, year, length, genre, studioName, studioAddr).`

- Assume that the following FDs hold:

  `title year` $\rightarrow$ `genre length studioName`

  `studioName` $\rightarrow$ `studioAddr`

- By transitivity, the following FD is derived:

  `title year` $\rightarrow$ `studioAddr`

Definition: Minimal Basis

Given a set of FDs $\mathcal{S}$, any set of FDs equivalent to $\mathcal{S}$ is a basis for $\mathcal{S}$. A minimal basis for a relation is a basis $\mathcal{B}$ that satisfies:

1. Every FD in $\mathcal{B}$ has a single attribute on the `rhs`.
2. If any FD is removed from $\mathcal{B}$, it is no longer a basis.
3. If for any FD in $\mathcal{B}$ we remove one or more attributes from the `lhs` of the FD, the result is no longer a basis.

Note: a minimal basis has no trivial FD, as they are removed in step (2).

Required

- Go over example 3.11.

Armstrong's Axioms

Armstrong's Axioms.

- 1. Reflexivity
  - If $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$ then $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$.
- 2. Augmentation
  - If $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$ then $A_1 A_2 \cdots A_n C_1 \cdots C_k \rightarrow B_1 B_2 \cdots B_m C_1 \cdots C_k$ for any set of attributes $\{C_1, \ldots, C_k\}$. Some of the C's may appear among the A's or B's, so we eliminate duplicate attributes from both the `lhs` and `rhs`.
- 3. Transitivity
  - If $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$ and $B_1 B_2 \cdots B_m \rightarrow C_1 \cdots C_k$ then $A_1 A_2 \cdots A_n \rightarrow C_1 \cdots C_k$.
- Observations.
  - Sound and complete set of rules for deriving FDs.
  - Thus, derives any FD that follows from a given set of FDs.
  - But not systematic as the closure computation we just saw.

Projecting FDs

Let $\mathtt{R}$ be a relation with FDs $\mathcal{S}$ and $\mathtt{R}_1 = \pi_{\mathtt{L}}(\mathtt{R})$ for some attributes L from $\mathtt{R}$.

- What FDs hold in $\mathtt{R}_1$?
- Compute the projection of functional dependencies $\mathcal{S}$.
- This is the set of all FDs that
  - Follow from $\mathcal{S}$, and
  - Involve only attributes of $\mathtt{R}_1$.
- In general, this computation is exponential in the size of $\mathtt{L}$.

Projecting FDs

Algorithm

INPUT: Relations $\mathtt{R}$ and $\mathtt{R}_1 = \pi_{\mathtt{L}}(\mathtt{R})$, and a set of FDs $\mathcal{S}$ that hold in $\mathtt{R}$.

OUTPUT: Set of FDs that hold in $\mathtt{R}_1$.

1. Let T be the output and initialize it as the empty set.
2. For each subset X of attributes of $\mathtt{R}_1$, compute X^+ w.r.t. $\mathcal{S}$. Add to T all nontrivial FDs $\mathtt{X} \rightarrow \mathtt{A}$ such that A is both in X^+ and an attribute of $\mathtt{R}_1$.
3. T is a basis for the FDs that hold in $\mathtt{R}_1$. If T is not minimal:
   - If some FD F in T follows from the other FDs in T, remove F from T.
   - Let $\mathtt{Y} \rightarrow \mathtt{B}$ be an FD in T, with at least two attributes in $\mathtt{Y}$, and $\mathtt{Z}$ a subset of $\mathtt{Y}$ with one attribute removed. If $\mathtt{Z} \rightarrow \mathtt{B}$ follows from the FDs in T (including $\mathtt{Y} \rightarrow \mathtt{B}$), replace $\mathtt{Y} \rightarrow \mathtt{B}$ by $\mathtt{Z} \rightarrow \mathtt{B}$.
   - Repeat these steps in all possible ways until T can no longer be changed.
4. Output T.

Required

- Go over example 3.13.
- Go over problems 3.2.2, 3.2.4, 3.2.5, 3.2.8.
- Read section 3.2 from chapter #3 and go over all remaining problems.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

Example: Schema Design

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

Problems:

- Redundancy.
  - Information repeated unnecessarily in several tuples.
- Update Anomalies.
  - Update data in one tuple and leave *the same data* unchanged in another.
- Deletion Anomalies.
  - If a set of values becomes empty, we may lose other information as side-effect.

Relation Decomposition

Definition: Relation Decomposition

Given a relation $R(A_1, A_2, \ldots, A_n)$, we may decompose R into two new relations $S(B_1, B_2, \ldots, B_m)$ and $T(C_1, C_2, \ldots, C_k)$ such that:

- $\{A_1, A_2, \ldots, A_n\} = \{B_1, B_2, \ldots, B_m\} \cup \{C_1, C_2, \ldots, C_k\}$.
- $S = \pi_{B_1, B_2, \ldots, B_m}(R)$.
- $T = \pi_{C_1, C_2, \ldots, C_k}(R)$.

Observations:

- $\{B_1, B_2, \ldots, B_m\}$ and $\{C_1, C_2, \ldots, C_k\}$ need not be disjoint.
- Decomposition is defined independently of its effectiveness in eliminating anomalies.

Example: Schema Design

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone with the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Table: `Movies2(title, year, length, genre, studioName).`

title	year	starName
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone with the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

Table: `Movies3(title, year, starName).`

Do any anomalies remain in this new design?

Boyce-Codd Normal Form

Definition: Boyce-Codd Normal Form (BCNF)

Consider a relation `R` and two sets of attributes `A` and `B`. Relation `R` is in Boyce-Codd Normal Form (BCNF) if and only if:

- Whenever there exists a non-trivial functional dependency $\texttt{A} \rightarrow \texttt{B}$ in `R`, `A` is a superkey of `R`.

If a schema conforms to BCNF, the anomalies discussed are guaranteed not to exist.

Example: Boyce-Codd Normal Form

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName)`.

Relation `Movies1` is not in BCNF.

- The only key of `Movies1` is `{title, year, starName}`.
- Consider the FD `title year` $\rightarrow$ `length genre studioName`.
- The `lhs` of the FD above is not a superkey of `Movies1`.

Example: Boyce-Codd Normal Form

title	year	length	genre	studioName
Star Wars	1977	124	SciFi	Fox
Gone with the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Table: `Movies2(title, year, length, genre, studioName)`.

Relation `Movies2` is in BCNF.

- The only key of `Movies2` is `{title, year}`.
- All other non-trivial FDs have `title` and `year` on their `lhs`.
- Hence, their `lhs`'s are superkeys and `Movies2` is in BCNF.

Boyce-Codd Normal Form

Decomposition of a relation `R` into BCNF (Sketch).

- Chose subsets of the attributes in `R` such that:
  - Each subset becomes the schema of decomposed relation in BCNF.
  - Data in `R` is represented faithfully by the decomposed relations.
  - That is, we are able to recompose the original relation exactly.
- Intuition: use non-trivial FDs violating BCNF to drive the decomposition.

Example: Boyce-Codd Normal Form

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: `Movies1(title, year, length, genre, studioName, starName).`

Decompose using the FD `title year` $\rightarrow$ `length genre studioName`.

- Using both `lhs` and `rhs` of the FD:
  `Movies2(title, year, length, genre, studioName).`
- The `lhs` of the FD and all remaining attributes:
  `Movies3(title, year, starName).`
- This is the decomposition we have seen before!

Boyce-Codd Normal Form

Algorithm

INPUT: Relation $\mathtt{R}_0$ with a set of FDs $\mathcal{S}_0$.

OUTPUT: Decomposition of $\mathtt{R}_0$ into a collection of relations in BCNF.

PROCEDURE: Run the steps below recursively for any relation $\mathtt{R}$ and set of FDs $\mathcal{S}$ that must be decomposed into BCNF. Start with $\mathtt{R} = \mathtt{R}_0$ and $\mathcal{S} = \mathcal{S}_0$.

1. If $\mathtt{R}$ already is in BCNF, return $\{\mathtt{R}\}$.
2. Let $\mathtt{X} \rightarrow \mathtt{Y}$ be a FD violating BCNF. Choose $\mathtt{R}_1 = \mathtt{X}^+$ as one schema and let $\mathtt{R}_2$ have attributes $\mathtt{X}$ and those of $\mathtt{R}$ not in $\mathtt{X}^+$.
3. Compute the FDs $\mathcal{S}_1$ and $\mathcal{S}_2$ for $\mathtt{R}_1$ and $\mathtt{R}_2$, respectively.
4. Recursively decompose $\mathtt{R}_1$ and $\mathtt{R}_2$.
5. Return the union of all decompositions.

Required

- Go over example 3.19.
- Go over problems 3.3.1, 3.3.2, 3.3.3, 3.3.4.
- Read section 3.3 from chapter #3 and go over the remaining problems.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

Decomposition: Desirable Properties

When decomposing, we strive to eliminate all anomalies from a schema. However, decomposition may may introduce problems of its own. Thus, we aim for decompositions having the following properties:

- Elimination of Anomalies.
  - We already know an algorithm that accomplishes this (BCNF).
- Recoverability of Information.
  - Is it possible to recover the original relation from tuples in the decomposition?
  - Is there a systematic way do accomplish this?
- Preservation of Dependencies.
  - If we reconstruct the original relation by joining, will the result satisfy the original FDs from the ones projected onto the decomposed relations?

Decomposition: Desirable Properties

Lossless Joins

A decomposition of a relation $\mathtt{R}$ is a lossless-join decomposition if we can join the decomposed relations to obtain the instance of $\mathtt{R}$ back.

Intuition: Consider the relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C})$ and a FD $\mathtt{B} \rightarrow \mathtt{C}$ that is a BCNF violation. Decomposing based on this FD yields $\mathtt{R}_1(\mathtt{A}, \mathtt{B})$ and $\mathtt{R}_2(\mathtt{B}, \mathtt{C})$. Now, let $t = (a, b, c)$ be a tuple of $\mathtt{R}$. Note that (a, b) is a (projected) tuple in $\mathtt{R}_1$ and (b, c) one in $\mathtt{R}_2$. When we perform the natural join of $\mathtt{R}_1$ and $\mathtt{R}_2$, since the tuples agree on their $\mathtt{B}$ components, we obtain the original tuple t.

The BCNF decomposition algorithm guarantees that we can recover the original relation by performing a natural join on the decomposed relations.

Required

- Go over example 3.21.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C})$ on which neither $\mathtt{B} \rightarrow \mathtt{A}$ nor $\mathtt{B} \rightarrow \mathtt{C}$ holds. Say we decompose $\mathtt{R}$ into $\mathtt{R}_1(\mathtt{A}, \mathtt{B})$ and $\mathtt{R}_2(\mathtt{B}, \mathtt{C})$. Then,

A	B	C
1	2	3
4	2	5

Table: Valid $\mathtt{R}$ instance.

A	B
1	2
4	2

Table: $\pi_{\mathtt{AB}}(\mathtt{R})$.

B	C
2	3
2	5

Table: $\pi_{\mathtt{BC}}(\mathtt{R})$.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C})$ on which neither $\mathtt{B} \rightarrow \mathtt{A}$ nor $\mathtt{B} \rightarrow \mathtt{C}$ holds. Say we decompose $\mathtt{R}$ into $\mathtt{R}_1(\mathtt{A}, \mathtt{B})$ and $\mathtt{R}_2(\mathtt{B}, \mathtt{C})$. Then,

A	B	C
1	2	3
4	2	5

Table: Valid $\mathtt{R}$ instance.

A	B
1	2
4	2

Table: $\pi_{\mathtt{AB}}(\mathtt{R})$.

B	C
2	3
2	5

Table: $\pi_{\mathtt{BC}}(\mathtt{R})$.

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

Table: $\pi_{\mathtt{AB}}(\mathtt{R}) \bowtie \pi_{\mathtt{BC}}(\mathtt{R})$ introduces spurious tuples!

The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple t in $\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_k}(R)$, the natural join of the decomposed relations, is also a tuple in the original relation R. That is, the decomposition does not produce spurious tuples.

Observation #1. The join is associative and commutative– join order does not matter. The join above is the set of tuples t such that t projected onto the attributes of S_i is a tuple in $\pi_{S_i}(R)$, for each $i \in [1, k]$.

Observation #2. Any tuple $t \in R$ surely is in the join. Simply note that the projection of t onto each S_i must be in $\pi_{S_i}(R)$ for every i.

Observation #3. If t is in the join, there must exist tuples $t_1, \ldots, t_k$ in R such that t is obtained by joining the projections of each t_i onto the attributes S_i, for $i \in [1, k]$. Further, t_i must agree with t on the attributes in S_i, but has unknown values otherwise.

The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple t in $\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_k}(R)$, the natural join of the decomposed relations, is also a tuple in the original relation R.

Setup. Create a tableau with the same schema as R. Add one symbolic tuple for every decomposed relation $\pi_{S_i}(R)$ as follows: use non-subscripted symbols for attributes in S_i and i-subscripted symbols for the unknowns.

The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple t in $\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_k}(R)$, the natural join of the decomposed relations, is also a tuple in the original relation R.

Chasing. Apply the FDs of R to equate symbols in the tableau until some tuple t' contains only non-subscripted symbols (i.e., t' is equal tuple t above) or no FDs can be further applied. When equating symbols, change all occurrences of one with the other.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A}, \mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A}, \mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B}, \mathtt{C}, \mathtt{D}\}$.

Setup the tableau for this decomposition and apply the chase to verify if this decomposition has the lossless join property.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A},\mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A},\mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B},\mathtt{C},\mathtt{D}\}$.

A	B	C	D
a	b_1	c_1	d
a	b_2	c	d_2
a_3	b	c	d

Step #1: Tableau setup based on the decomposition of $\mathtt{R}$.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A},\mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A},\mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B},\mathtt{C},\mathtt{D}\}$.

A	B	C	D
a	b_1	c_1	d
a	b_1	c	d_2
a_3	b	c	d

Step #2: Apply FD $\mathtt{A} \rightarrow \mathtt{B}$.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A},\mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A},\mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B},\mathtt{C},\mathtt{D}\}$.

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a_3	b	c	d

Step #3: Apply FD $\mathtt{B} \rightarrow \mathtt{C}$.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A},\mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A},\mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B},\mathtt{C},\mathtt{D}\}$.

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a	b	c	d

Step #4: Apply FD $\mathtt{CD} \rightarrow \mathtt{A}$.

Example: The Chase for Lossless Joins

Consider relation $\mathtt{R}(\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \rightarrow \mathtt{C}, \mathtt{CD} \rightarrow \mathtt{A}\}$. Say we decompose $\mathtt{R}$ into relations with sets of attributes $\mathtt{S}_1 = \{\mathtt{A},\mathtt{D}\}$, $\mathtt{S}_2 = \{\mathtt{A},\mathtt{C}\}$, and $\mathtt{S}_3 = \{\mathtt{B},\mathtt{C},\mathtt{D}\}$.

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a	b	c	d

Conclusion: Last tuple is evidence of lossless join decomposition.

Example: Dependency Preservation

Consider relation $\texttt{Bookings}(\texttt{title}, \texttt{theater}, \texttt{city})$ with FDs $\{\texttt{theater} \rightarrow \texttt{city}, \texttt{title city} \rightarrow \texttt{theater}\}$. Decompose into BCNF.

- Keys: $\{\texttt{title}, \texttt{city}\}$ and $\{\texttt{theater}, \texttt{title}\}$.
- BCNF Violation: $\texttt{theater} \rightarrow \texttt{city}$.
- Decomposition: $S_1 = \{\texttt{theater}, \texttt{city}\}, S_2 = \{\texttt{theater}, \texttt{title}\}$.
- Problem: $\texttt{title city} \rightarrow \texttt{theater}$.

theater	city
Guild	NYC
Park	NYC

$\pi_{S_1}(\texttt{Bookings})$

theater	title
Guild	Antz
Park	Antz

$\pi_{S_2}(\texttt{Bookings})$

theater	city	title
Guild	NYC	Antz
Park	NYC	Antz

$\pi_{S_1}(\texttt{Bookings}) \bowtie \pi_{S_2}(\texttt{Bookings})$

- $\pi_{S_1}(\texttt{Bookings})$ and $\pi_{S_2}(\texttt{Bookings})$ are valid instances.
- But $\pi_{S_1}(\texttt{Bookings}) \bowtie \pi_{S_2}(\texttt{Bookings})$ violates $\texttt{title city} \rightarrow \texttt{theater}$.

Dependency Preservation

Unfortunately, it is not always possible to decompose into BCNF and preserve functional dependencies.

Required

- Go over example 3.25.
- Go over problems 3.4.1, 3.4.2.
- Read section 3.4 from chapter #3 and go over the remaining problems.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

Third Normal Form

Definition: Third Normal Form (BCNF)

Consider a relation `R` and two sets of attributes `A` and `B`. Relation `R` is in Third Normal Form (3NF) if and only if:

- Whenever there exists a non-trivial functional dependency $\texttt{A} \rightarrow \texttt{B}$ in `R`, `A` is a superkey of `R`, **OR**
- Every attribute in $\texttt{B} \setminus \texttt{A}$ is a member of some key (need not be the same).

An attribute that is a member of some key is called prime. Thus, 3NF states that the `lhs` of every non-trivial FD is a superkey, or all attributes on the `rhs` are prime.

Third Normal Form

Algorithm: 3NF Synthesis

INPUT: Relation $\mathtt{R}$ with a set of FDs $\mathcal{F}$.

OUTPUT: Decomposition of $\mathtt{R}$ into a collection of relations in 3NF.

1. Find a minimal basis for $\mathcal{F}$ (call it $\mathcal{G}$).
2. For each FD $\mathtt{X} \rightarrow \mathtt{A}$ in $\mathcal{G}$, use $\mathtt{X} \cup \mathtt{A}$ as attributes of one of the relations in the decomposition.
3. If none of the relations obtained in (2) contain a superkey of $\mathtt{R}$, add another relation whose schema is a key of $\mathtt{R}$.

The 3NF synthesis algorithm gurantees that: a) all decomposed relations are in 3NF, b) the decomposition has a lossless join, and c) the decomposition is dependency-preserving.

Required

- Go over example 3.27.
- Go over problems 3.5.1, 3.5.4.
- Read section 3.5 from chapter #3 and go over the remaining problems.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

Attribute Independence

In certain cases, a BCNF schema may still show some redundancy. A common source is having two or more set-valued properties of the key into a single relation.

Example: Attribute Independence

name	street	city	title	year
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

Table: Sets of addresses independent from movies.

- Stars may have multiple addresses and act in multiple movies.
- No reason to associate one address with one movie and not the other.
- This relation has no non-trivial FD, therefore, it is in BCNF!

Multivalued Dependency

Definition: Multivalued Dependency

A multivalued dependency (MVD) on a relation R is a statement of the form:

$$A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$$

where $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ are attributes of R. This MVD holds if, for each pair of tuples t and u of R that agree on their $A_1, A_2, \ldots, A_n$ values, there exists some tuple v in R that agrees:

1. with the $A_1, A_2, \ldots, A_n$ values of t and u,
2. with the $B_1, B_2, \ldots, B_m$ values of t, and
3. with u on the values of all attributes of R not among the A's or B's.

Observation: swapping t and u guarantees the existence of another tuple w. Thus, for any fixed values of the A's, the B's and the other attributes appear in all possible combinations in different tuples (hence their independence).

Example: Multivalued Dependency

name	street	city	title	year
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

Table: Sets of addresses independent from movies.

- Now we can express the following MVD: $\texttt{name} \twoheadrightarrow \texttt{street city}$.
- Tha is, each star name appears with their set of addresses.
- Let t and u be the first and second blue rows. Then, v is the red row.
- If we swap t and u, then v is the yellow row.

Multivalued Dependency

Rules involving MVDs:

- Triviality. (analogous to FDs)
- Transitivity. (analogous to FDs)
- Promotion. Every FD is an MVD.
- Complementation. If $A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$ is an MVD for a relation R then R also satisfies $A_1 A_2 \ldots A_n \twoheadrightarrow C_1 C_2 \ldots C_k$, where the Cs are the attributes of R not among the As and Bs (swapping Bs in a tuple that agrees on the As has the same effect as swapping the Cs).
- More Triviality. If $\{A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m\}$ are all the attributes of R, then $A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$.

Observation: the combining and splitting rules do not apply to MVDs.

Fourth Normal Form

Definition: Fourth Normal Form

Consider a relation $\mathtt{R}$ and two sets of attributes $\mathtt{A}$ and $\mathtt{B}$. Relation $\mathtt{R}$ is in Fourth Normal Form (4NF) if and only if:

- Whenever there exists a non-trivial multivalued dependency $\mathtt{A} \twoheadrightarrow \mathtt{B}$ in $\mathtt{R}$, $\mathtt{A}$ is a superkey of $\mathtt{R}$.

Observation: 4NF is a generalization of BCNF, thus, every relation in 4NF is also in BCNF and every BCNF violation is also a 4NF violation.

Fourth Normal Form

Algorithm

INPUT: Relation $\mathtt{R}_0$ with a set of FDs+MVDs $\mathcal{S}_0$.

OUTPUT: Decomposition of $\mathtt{R}_0$ into a collection of relations in 4NF.

PROCEDURE: Run the steps below recursively for any relation $\mathtt{R}$ and set of FDs+MVDs $\mathcal{S}$ that must be decomposed into 4NF. Start with $\mathtt{R} = \mathtt{R}_0$ and $\mathcal{S} = \mathcal{S}_0$.

1. If $\mathtt{R}$ already is in 4NF, return $\{\mathtt{R}\}$.
2. Let $\mathtt{X} \twoheadrightarrow \mathtt{Y}$ be a 4NF (true MVD or FD) violation. Let $\mathtt{R}_1$ have schema $\mathtt{X} \cup \mathtt{Y}$ and $\mathtt{R}_2$ schema $\mathtt{X} \cup (\mathtt{R} - \mathtt{X} - \mathtt{Y})$.
3. Compute the FDs+MVDs $\mathcal{S}_1$ and $\mathcal{S}_2$ for $\mathtt{R}_1$ and $\mathtt{R}_2$, respectively.
4. Recursively decompose $\mathtt{R}_1$ and $\mathtt{R}_2$.
5. Return the union of all decompositions.

Required

- Go over example 3.34.
- Go over problems 3.6.2, 3.6.3.
- Read section 3.6 from chapter #3 and go over the remaining problems.

Next...

1. Design Theory
   - Functional Dependencies
   - FD Rules
   - Boyce-Codd Normal Form
   - Decomposition
   - Third Normal Form
   - Multivalued Dependencies
   - Discovering MVDs

The Chase: Checking if an FD holds

The Chase.

- First application: checking for lossless join decomposition.
- New application: checking if a FD holds in a relation.
- Consider a relation $\mathtt{R}$ and a set of FDs $\mathcal{F}$ that hold in $\mathtt{R}$.
- To know whether the FD $\mathtt{X} \rightarrow \mathtt{Y}$ follows from $\mathcal{F}$, proceed as follows:
  - Create a tableau with two rows agreeing on their $\mathtt{X}$ values only.
  - Chase the tableau using the FDs in $\mathcal{F}$.
  - If the final tableau agrees in all columns of $\mathtt{Y}$ then $\mathtt{X} \rightarrow \mathtt{Y}$ holds in $\mathtt{R}$.

Example: The Chase for FDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F})$ with FDs $\mathcal{F} = \{\mathtt{AB} \rightarrow \mathtt{C}, \mathtt{BC} \rightarrow \mathtt{AD}, \mathtt{D} \rightarrow \mathtt{E}, \mathtt{CF} \rightarrow \mathtt{B}\}$ and say we wish to know whether $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

A	B	C	D	E	F
a	b	c_1	d_1	e_1	f_1
a	b	c_2	d_2	e_2	f_2

Step #1: Tableau setup based on the FD $\{\mathtt{AB} \rightarrow \mathtt{D}\}$.

Example: The Chase for FDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F})$ with FDs $\mathcal{F} = \{\mathtt{AB} \rightarrow \mathtt{C}, \mathtt{BC} \rightarrow \mathtt{AD}, \mathtt{D} \rightarrow \mathtt{E}, \mathtt{CF} \rightarrow \mathtt{B}\}$ and say we wish to know whether $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

A	B	C	D	E	F
a	b	c_1	d_1	e_1	f_1
a	b	c_1	d_2	e_2	f_2

Step #2: Apply FD $\{\mathtt{AB} \rightarrow \mathtt{C}\}$.

Example: The Chase for FDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F})$ with FDs $\mathcal{F} = \{\mathtt{AB} \rightarrow \mathtt{C}, \mathtt{BC} \rightarrow \mathtt{AD}, \mathtt{D} \rightarrow \mathtt{E}, \mathtt{CF} \rightarrow \mathtt{B}\}$ and say we wish to know whether $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

A	B	C	D	E	F
a	b	c_1	d_1	e_1	f_1
a	b	c_1	d_1	e_2	f_2

Step #3: Apply FD $\{\mathtt{BC} \rightarrow \mathtt{AD}\}$.

Example: The Chase for FDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F})$ with FDs $\mathcal{F} = \{\mathtt{AB} \rightarrow \mathtt{C}, \mathtt{BC} \rightarrow \mathtt{AD}, \mathtt{D} \rightarrow \mathtt{E}, \mathtt{CF} \rightarrow \mathtt{B}\}$ and say we wish to know whether $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

A	B	C	D	E	F
a	b	c_1	d_1	e_1	f_1
a	b	c_1	d_1	e_1	f_2

Step #4: Apply FD $\{\mathtt{D} \rightarrow \mathtt{E}\}$.

Example: The Chase for FDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F})$ with FDs $\mathcal{F} = \{\mathtt{AB} \rightarrow \mathtt{C}, \mathtt{BC} \rightarrow \mathtt{AD}, \mathtt{D} \rightarrow \mathtt{E}, \mathtt{CF} \rightarrow \mathtt{B}\}$ and say we wish to know whether $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

A	B	C	D	E	F
a	b	c_1	d_1	e_1	f_1
a	b	c_1	d_1	e_1	f_2

Conclusion: The tuples agree on $\mathtt{D}$ thus $\{\mathtt{AB} \rightarrow \mathtt{D}\}$ holds in $\mathtt{R}$.

The Chase: Checking if an MVD holds

Checking if a MVD holds in a relation.

- Consider a relation $\mathtt{R}$ and a set of FDs+MVDs $\mathcal{F}$ for $\mathtt{R}$.
- To know whether the MVD $\mathtt{X} \twoheadrightarrow \mathtt{Y}$ follows from $\mathcal{F}$, proceed as follows:
  - Create a tableau with two rows agreeing on the values of $\mathtt{X}$ only.
  - Chase the tableau using the FDs of $\mathcal{F}$.
  - Apply each MVD of $\mathcal{F}$ as follows: given a MVD $\mathtt{A} \twoheadrightarrow \mathtt{B}$, if two tuples in the tableau agree on $\mathtt{A}$, form two new tuples by swapping all their $\mathtt{B}$ values.
  - If we ever discover that one of the original tuples with its $\mathtt{Y}$ values replaced by those of the other original tuple is in the tableau, the MVD holds in $\mathtt{R}$.

Example: The Chase for MVDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \twoheadrightarrow \mathtt{C}\}$ and say we wish to know whether $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$ holds in $\mathtt{R}$.

A	B	C	D
a	b_1	c	d_1
a	b	c_2	d

Step #1: Tableau setup based on the MVD $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$.

Example: The Chase for MVDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \twoheadrightarrow \mathtt{C}\}$ and say we wish to know whether $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$ holds in $\mathtt{R}$.

A	B	C	D
a	b	c	d_1
a	b	c_2	d

Step #2: Apply FD $\mathtt{A} \rightarrow \mathtt{B}$.

Example: The Chase for MVDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \twoheadrightarrow \mathtt{C}\}$ and say we wish to know whether $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$ holds in $\mathtt{R}$.

A	B	C	D
a	b	c	d_1
a	b	c_2	d
a	b	c_2	d_1
a	b	c	d

Step #3: Apply MVD $\mathtt{B} \twoheadrightarrow \mathtt{C}$.

Example: The Chase for MVDs

Consider relation $\mathtt{R}(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D})$ with FDs $\mathcal{F} = \{\mathtt{A} \rightarrow \mathtt{B}, \mathtt{B} \twoheadrightarrow \mathtt{C}\}$ and say we wish to know whether $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$ holds in $\mathtt{R}$.

A	B	C	D
a	b	c	d_1
a	b	c_2	d
a	b	c_2	d_1
a	b	c	d

Conclusion: Last tuple is evidence that $\{\mathtt{A} \twoheadrightarrow \mathtt{C}\}$ holds in $\mathtt{R}$.

Projecting MVDs

Just like the projection of FDs is essential to the BCNF algorithm, the projection of MVDs is essential to the 4NF algorithm.

To verify whether an MVD holds in a projected relation (decomposition), we apply the chase on the full set of attributes of the original relation. The goal is to produce two rows in the tableau that satisfy the MVD condition for the decomposed relation.

As with FDs, there are shortcuts: do not check trivial FDs and MVDs; it suffices to look for FDs with a single attribute on the `rhs`; FDs or MVDs whose `lhs` does not contain the `lhs` of any MVD or FD can be ignored (the chase cannot start).

Required

- Go over example 3.38 and problems 3.7.1, 3.7.4.
- Read section 3.7 from chapter #3.
- Go over the remaining problems of section 3.7.