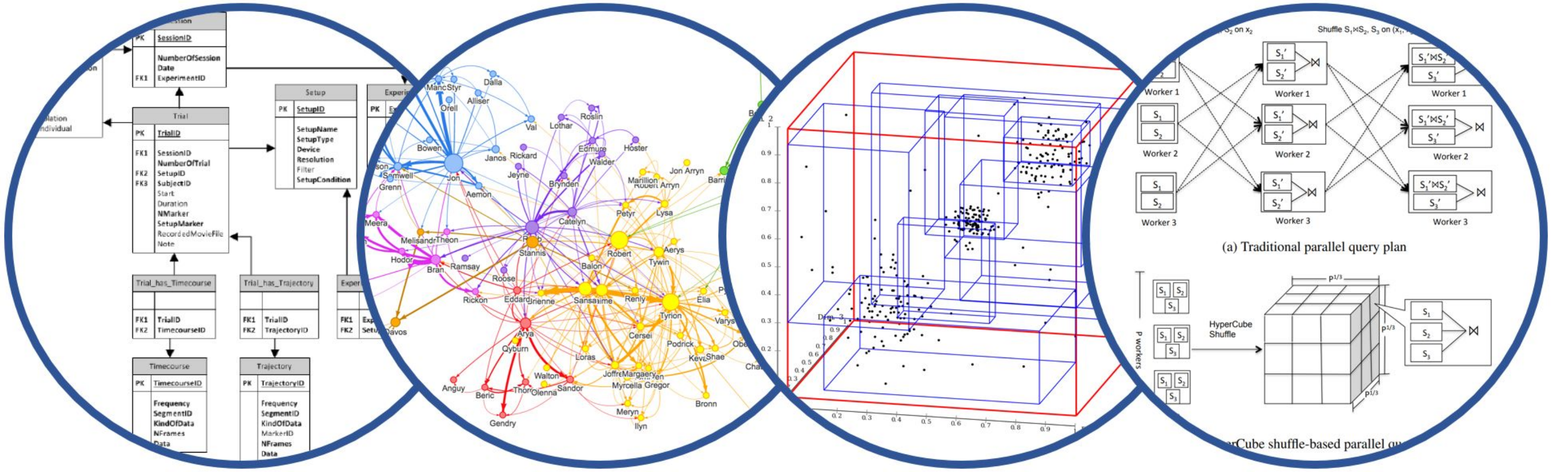


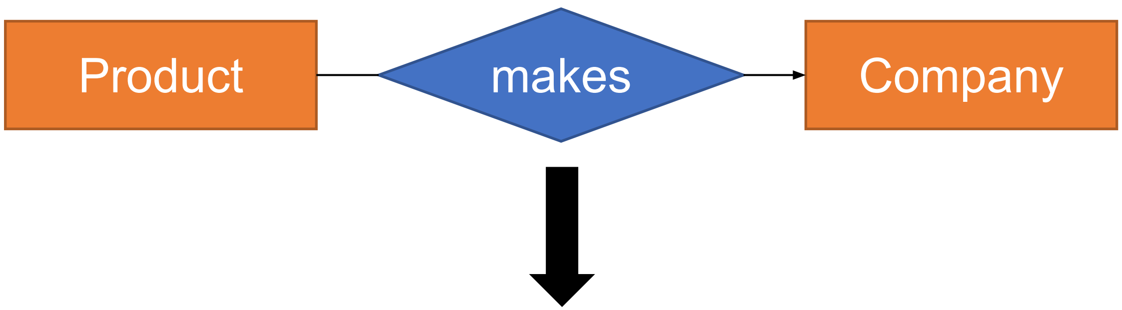
Introduction to Data Management

Design Theory

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Recap

- ER Diagrams
  - Conceptual modeling
  - Rules of thumb for converting diagram into schema




```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY
KEY,
    ...);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY
KEY,
    cname VARCHAR(100)
      REFERENCES Company
    ...);
```

Goals for Today

- Figure out the fundamentals of what makes a good schema

Outline

- Background
  - Anomalies, i.e. things we want to avoid
  - Functional Dependencies (FDs)
  - Closures and formal definitions of keys
- Normalization: BCNF Decomposition

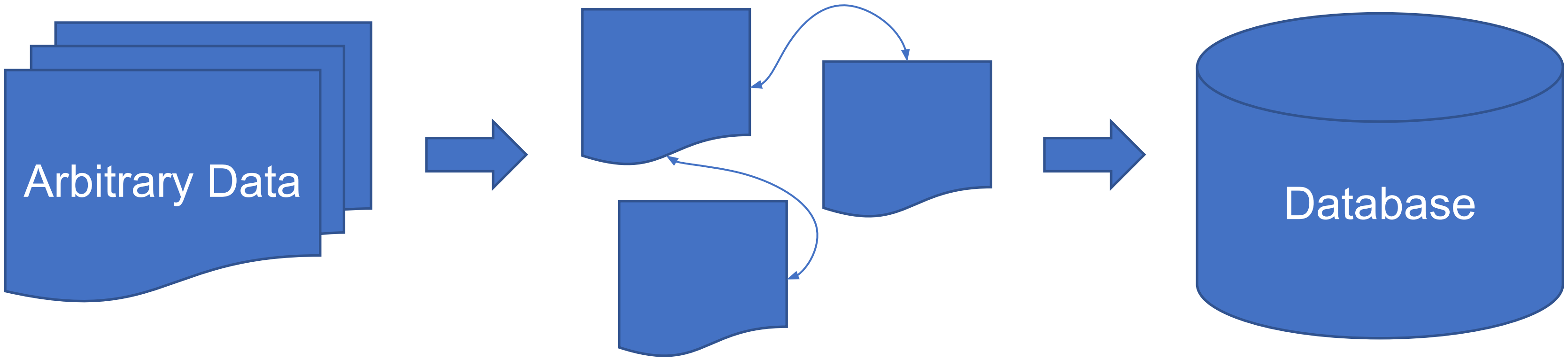
Informal Design Guidelines

- Semantics of attributes should be self-evident
- Avoid redundant information in tuples
- Avoid NULL values in tuples
- Disallow the generation of “spurious” tuples
  - If certain tuples shouldn’t exist, don’t allow them

Database Design

Database Design

Database Design or **Logical Design** or **Relational Schema Design** is the process of organizing data into a database model. This is done by considering what data needs to be stored and the interrelationship of the data.



Database Design

Database Design is about

(1) characterizing data and (2) organizing data

Database Design

Database Design is about

(1) characterizing data and (2) organizing data

How to talk about properties

we know or see in the data

Data Interrelationships

How do we start talking about data interrelationships?

- What rules govern our data?
  - Domain knowledge
    - Dimension vs measure
  - Pattern analysis

Data Interrelationships

How do we start talking about data interrelationships?

- What rules govern our data?
  - Domain knowledge
    - Dimension vs measure
  - Pattern analysis

The rules that are known to us since we **made them up** or they correlate to **things in the real world**

[ex] An engineer knows that a plane model determines the plane's wingspan

Think About This

Make a simple directory that can:

- Hold information about name, SSN, phone, and city
- Associate **people** with the **city** they live in
- Associate **people** with any **phone numbers** they have

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

The above instance does the job, but are there issues?

Think About This

Make a simple directory that can:

- Hold information about name, SSN, phone, and city
- Associate **people** with the **city** they live in
- Associate **people** with any **phone numbers** they have

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

Anomalies:

- **Redundancy ➡ Slow Update**
  - Change Fred’s city to Bellevue (two rows!)
- **Deletion Anomalies**
  - How to delete Joe’s phone without deleting Joe?

Think About This

Make a simple directory that can:

- Hold information about name, SSN, phone, and city
- Associate **people** with the **city** they live in
- Associate **people** with any **phone numbers** they have

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

Anomalies:

- **Redundancy ➡ Slow Update**
  - Change Fred’s city to Bellevue (two rows!)
- **Deletion Anomalies**
  - How to delete Joe’s phone without deleting Joe?

Think About This

Make a simple directory that can:

- Hold information about name, SSN, phone, and city
- Associate **people** with the **city** they live in
- Associate **people** with any **phone numbers** they have

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

Anomalies:

- **Redundancy ➡ Slow Update**
  - Change Fred’s city to Bellevue (two rows!)
- **Deletion Anomalies**
  - How to delete Joe’s phone without deleting Joe?

Think About This

We can solve the anomalies by converting this

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

into this

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	San Francisco

SSN	Phone
123-45-6789	206-555-9999
123-45-6789	206-555-8888
987-65-4321	415-555-7777

How can we systematically avoid anomalies?

Functional Dependencies (FDs)

Definition

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

$A_1 \ldots A_n$ **determines** $B_1 .. B_m$

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Data Interrelationships

Functional Dependency

A ***Functional Dependency*** $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ holds in the relation R if:

$$\forall t, t' \in R, (t.A_1 = t'.A_1 \land \ldots \land t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \land \ldots \land t.B_n = t'.B_n)$$

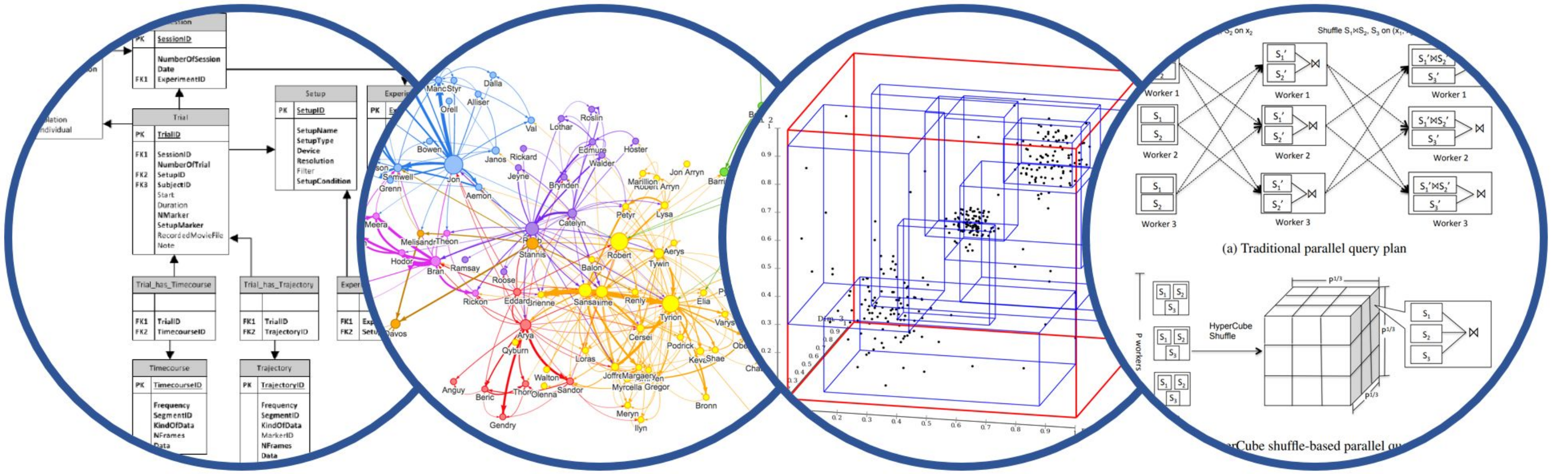
Informally, some attributes determine other attributes.

$$A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$$

This is the antecedent

This is the consequent

Warning! Dependency does not imply causation!



Introduction to Data Management
Design Theory

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Think About This

We can solve the anomalies by converting this

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

into this

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	San Francisco

SSN	Phone
123-45-6789	206-555-9999
123-45-6789	206-555-8888
987-65-4321	415-555-7777

How can we systematically avoid anomalies?

Recap - Functional Dependencies (FDs)

Definition

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

$A_1 \ldots A_n$ **determines** $B_1..B_m$

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Example

An FD holds, or does not hold on an instance:

EmpID	**Name**	**Phone**	**Position**
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID ➡ Name, Phone, Position

Position ➡ Phone

but not Phone ➡ Position

Example

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876 ⬅	Salesrep
E1111	Smith	9876 ⬅	Salesrep
E9999	Mary	1234	Lawyer

Position ➡ Phone

Example

EmpID	Name	Phone	Position
E0045	Smith	1234 ➡	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 ➡	Lawyer

But not Phone ➡ Position

Checking with Queries

An FD <u>holds</u>, or <u>does not hold</u> on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID ➡ Name, Phone, Position

Position ➡ Phone

but not

Phone ➡ Position

```
SELECT *
  FROM R1, R2
 WHERE (R1.position = R2.position)
   AND (R1.Phone != R2.Phone)
```

```
SELECT *
  FROM R1, R2
 WHERE (R1.phone = R2.phone)
   AND (R1.position != R2.position)
```

Checking with Queries

An FD holds, or does not hold on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID ➡ Name, Phone, Position

Position ➡ Phone

but not

Phone ➡ Position

```
SELECT *
  FROM R1, R2
 WHERE (R1.position = R2.position)
   AND (R1.Phone != R2.Phone)
```

```
SELECT *
  FROM R1, R2
 WHERE (R1.phone = R2.phone)
   AND (R1.position != R2.position)
```

If no results, then the FD holds!

Example

name ➡ color
category ➡ department
color, category ➡ price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Do all the FDs hold on this instance?

Example

name ➡ color
category ➡ department
color, category ➡ price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Do all the FDs hold on this instance?

No!

Example

name ➡ color
category ➡ department
color, category ➡ price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	49

Do all the FDs hold on this instance?

Now they do!

Example

name ➡ color
category ➡ department
color, category ➡ price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	49
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

Buzzwords

- FD **holds** or **does not hold** on an instance
- If we can be sure that *every instance of R* will be one in which a given FD is true, then we say that **R satisfies the FD**
- If we say that R satisfies an FD, we are **stating a constraint on R**

An Interesting Observation

If all these FDs are true:

name ➡ color
category ➡ department
color, category ➡ price

Then this FD also holds:

name, category ➡ price

An Interesting Observation

If all these FDs are true:

name ➡ color
category ➡ department
color, category ➡ price

Then this FD also holds:

name, category ➡ price

An Interesting Observation

If all these FDs are true:

name ➡ color
category ➡ department
color, category ➡ price

Then this FD also holds:

name, category ➡ price

If we find out from application domain that a relation satisfies some FDs, it doesn't mean that we found all the FDs that it satisfies!
There could be more FDs implied by the ones we have.

Fundamentals of FDs

Armstrong's Axioms

- Axiom of ***Reflexivity*** (***Trivial FD***)

  If $B \subseteq A$ then $A \rightarrow B$

  [ex] $\{name\} \subseteq \{name, job\}$ so $\{name, job\} \rightarrow \{name\}$

- Axiom of ***Augmentation***

  If $A \rightarrow B$ then $\forall C, AC \rightarrow BC$

  [ex] $\{ID\} \rightarrow \{name\}$ so $\{ID, job\} \rightarrow \{name, job\}$

- Axiom of ***Transitivity***

  If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

  [ex] $\{ID\} \rightarrow \{name\}$ and $\{name\} \rightarrow \{initials\}$
  so $\{ID\} \rightarrow \{initials\}$

Fundamentals of FDs

Armstrong's Axioms

- Axiom of ***Reflexivity*** (***Trivial FD***)

  If $B \subseteq A$ then $A \to B$

  [ex] $\{name\} \subseteq \{name, job\}$ so $\{name, job\} \to \{name\}$

- Axiom of ***Augmentation***

  If $A \to B$ then $\forall C$, $AC \to BC$

  [ex] $\{ID\} \to \{name\}$ so $\{ID, job\}$

- Axiom of ***Transitivity***

  If $A \to B$ and $B \to C$ then $A \to C$

  [ex] $\{ID\} \to \{name\}$ and $\{name\} \to \{initials\}$
  so $\{ID\} \to \{initials\}$

We'll use transitivity all the time

Fundamentals of FDs

Interesting Secondary Rules

- **Pseudo Transitivity**

If $A \rightarrow BC$ and $C \rightarrow D$ then $A \rightarrow BD$

- **Extensivity**

If $A \rightarrow B$ then $A \rightarrow AB$

(Useful when connecting an AB → CD type FD to A via transitivity)

Fundamentals of FDs

Can I do this to FDs?

I only know $\{ID\} \rightarrow \{name\}$

So $\{ID, hair\ color\} \rightarrow \{name\}$

Fundamentals of FDs

Can I do this to FDs?

I only know $\{ID\} \rightarrow \{name\}$

So $\{ID, hair\ color\} \rightarrow \{name\}$

Yes! If tuples already agree on ID and name, partitioning the left side by hair color changes nothing.

Fundamentals of FDs

Can I do this to FDs?

I only know $\{ID\} \rightarrow \{name\}$

So $\{ID, hair\ color\} \rightarrow \{name\}$

Yes! If tuples already agree on ID and name, partitioning the left side by hair color changes nothing.

Adding more attributes to the left side can never remove attributes in the right side.

Fundamentals of FDs

What about this?

I only know $\{ID\} \rightarrow \{name\}$

So $\{ID\} \rightarrow \{name, hair\ color\}$

Fundamentals of FDs

What about this?

I only know $\{ID\} \rightarrow \{name\}$

So $\{ID\} \rightarrow \{name, hair\ color\}$

No! E.g.

ID	Name	Hair color
001	Ryan	Brown
001	Ryan	Grey

Fundamentals of FDs

What about this?

I only know $\{ID\} \to \{name\}$

So $\{ID\} \to \{name, hair\ color\}$

ID	Name	Hair color
001	Ryan	Brown
001	Ryan	Grey

No! E.g.

No way to use the axioms to introduce hair color to the right side without also introducing it to the left side.

Finding Keys

All this talk about FDs sounds awfully similar to keys…

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

Closure (example)

Given the functional dependencies:

- $SSN \rightarrow Name$
- $Name \rightarrow Initials$

We can derive some closures:

- $Name^+ = \{Name, Initials\}$
- $SSN^+ = \{SSN, Name, Initials\}$
- $Initials^+ = \{Initials\}$
- $\{SSN, Initials\}^+ = \{SSN, Name, Initials\}$

Preview: A key is the minimal set of attributes such that its closure contains all the attributes in the table!

Closure

Closure Algorithm

Find the closure of $\{A_1, \ldots, A_m\}$

	$X = \{A_1, \ldots, A_m\}$ ***Repeat until*** X ***does not change***: ***if*** $B1, \ldots, Bn \rightarrow C$ is a FD ***and*** $B1, \ldots, Bn \in X$ ***then*** $X \leftarrow X \cup C$	

In practice:

Repeated use of transitivity

Closure

Closure Algorithm

Find the closure of $\{A_1, \ldots, A_m\}$

$X = \{A_1, \ldots, A_m\}$

Repeat until X ***does not change***:
  if $B1, \ldots, Bn \rightarrow C$ is a FD ***and*** $B1, \ldots, Bn \in X$
  then $X \leftarrow X \cup C$

If a FD applies, add the consequent to the answer

In practice:

Repeated use of transitivity

Closure

Closure Algorithm

Find the closure of $\{A_1, \ldots, A_m\}$

$X = \{A_1, \ldots, A_m\}$

Repeat until X ***does not change***:
  if $B1, \ldots, Bn \rightarrow C$ is a FD ***and*** $B1, \ldots, Bn \in X$
  then $X \leftarrow X \cup C$

If a FD applies, add the consequent to the answer

In practice:

Repeated use of transitivity

1. Find some attribute(s) C to add to right side
2. Add them
3. Look back at the FDs to find more C

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

$\{\text{name, category}\}^+ \quad \supseteq \quad ?$

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \to B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

{name, category}$^+$ $\supseteq$ {name, category} [reflexivity]

Closure

Closure

The **Closure** of the set $\{A_1, \dots, A_m\}$, written as $\{A_1, \dots, A_m\}^+$, is the set of attributes B is such that $A_1, \dots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

We can think of this as X in our closure algorithm

{name, category}$^+$ $\supseteq$ {name, category} [reflexivity]

Closure

Closure

The **Closure** of the set $\{A_1, \dots, A_m\}$, written as $\{A_1, \dots, A_m\}^+$, is the set of attributes B is such that $A_1, \dots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

{name, category}$^+$ ⊇ {name, category} [reflexivity]

⊇ {name, category, color} [transitivity, name ➡ color]

Closure

Closure

The **Closure** of the set $\{A_1, \dots, A_m\}$, written as $\{A_1, \dots, A_m\}^+$, is the set of attributes B is such that $A_1, \dots, A_m \to B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

X iteratively grows

{name, category}$^+$ ⊇ {name, category} [reflexivity]

⊇ {name, category, color} [transitivity, name ➡ color]

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

{name, category}$^+$ ⊇ {name, category} [reflexivity]

⊇ {name, category, color} [transitivity, name ➡ color]

⊇ {name, category, color, department} [transitivity, category ➡ department]

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

$\{\text{name, category}\}^+$ ⊇ {name, category} [reflexivity]

⊇ {name, category, color} [transitivity, name ➡ color]

⊇ {name, category, color, department} [transitivity, category ➡ department]

= {name, category, color, department, price} [transitivity, color, category ➡ price]

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

{name, category}$^+$ ⊇ {name, category} [reflexivity]

⊇ {name, category, color} [transitivity, name ➡ color]

⊇ {name, category, color, department} [transitivity, category ➡ department]

= {name, category, color, department, price} [transitivity, color, category ➡ price]

Now X is equal to our closure

Closure

Closure

The **Closure** of the set $\{A_1, \ldots, A_m\}$, written as $\{A_1, \ldots, A_m\}^+$, is the set of attributes B is such that $A_1, \ldots, A_m \rightarrow B$.

A closure finds everything a set of attributes determines.

1. name ➡ color
2. category ➡ department
3. color, category ➡ price

{name, category}$^+$ = {name, category, color, department, price}

Note: If these are all the attributes, (name, category) is a key

Finding Keys

What do FDs and Closures do for us?

- Characterize the interrelationships of data
- Able to find keys

Finding Keys

Superkey

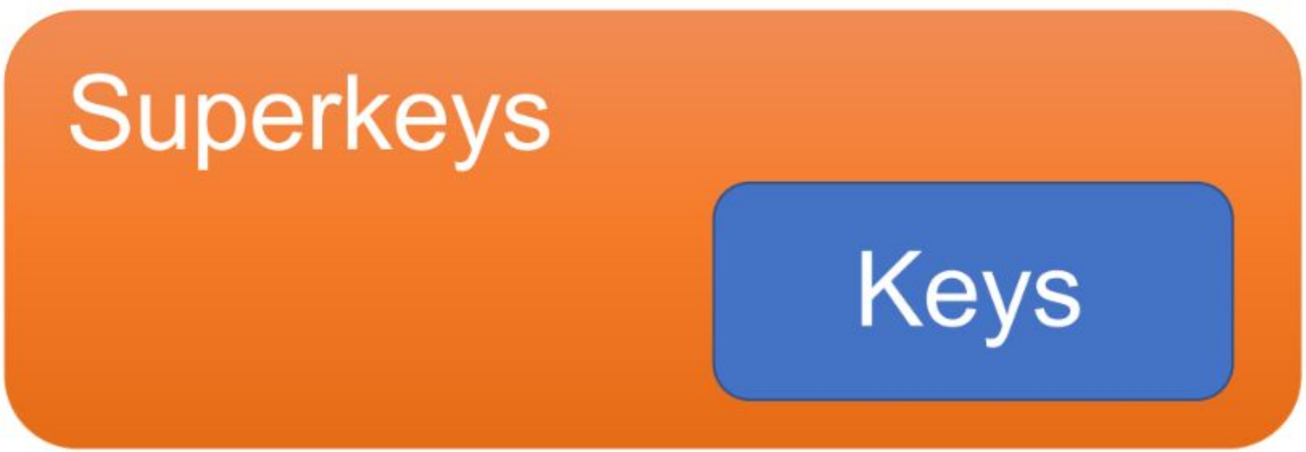
A **Superkey** is a set of attributes $A_1, \dots, A_n$ s.t. for any single attribute B:

$$A_1, \dots, A_n \rightarrow B$$

In other words, for the set of all attributes C in the relation R, the set $\{A_1, \dots, A_n\}$ is a superkey iff $\{A_1, \dots, A_n\}^+ = C$

Key

A **Key** is a minimal superkey, i.e. no subset of a key is a superkey.



Finding Keys

Candidate Key

When a relation has multiple keys, each key is a **Candidate Key**.

Usefulness of Keys in Design

What intuitions do we get from data interrelationships?

- FDs that are not superkeys hint at redundancy
  - If a FD antecedent <u>is **not** a superkey</u>, we can remove redundant information, i.e. the FD consequent

Usefulness of Keys in Design

What intuitions do we get from data interrelationships?

- FDs that are not superkeys hint at redundancy
  - If a FD antecedent <u>is **not** a superkey</u>, we can remove redundant information, i.e. the FD consequent
- Rephrased
  - $A \rightarrow B$ is fine if A is a superkey
  - Otherwise, we can extract B into a separate table

Usefulness of Keys in Design

What intuitions do we get from data interrelationships?

- FDs that are not superkeys hint at redundancy
  - If a FD antecedent is **not** a superkey, we can remove redundant information, i.e. the FD consequent
- Rephrased
  - $A \rightarrow B$ is fine if A is a superkey
  - Otherwise, we can extract B into a separate table

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

SSN is not a superkey!

Think About This

We can solve the anomalies by converting this

Name	SSN	Phone	City
Fred	123-45-6789	206-555-9999	Seattle
Fred	123-45-6789	206-555-8888	Seattle
Joe	987-65-4321	415-555-7777	San Francisco

into this

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	San Francisco

SSN	Phone
123-45-6789	206-555-9999
123-45-6789	206-555-8888
987-65-4321	415-555-7777

How can we systematically avoid anomalies?

Usefulness of Keys in Design

Restaurants(rid, name, rating, popularity)

rid ➡ name

rid ➡ rating

rating ➡ popularity

rid	name	rating	popularity
1	Mee Sum Pastry	3	Respectable
2	Café on the Ave	4	Poppin
3	Guanaco's Tacos	4	Poppin
4	Aladdin Gyro-Cery	5	Poppin

Usefulness of Keys in Design

Restaurants(rid, name, rating, popularity)

rid ➡ name

rid ➡ rating

Fine because rid is a superkey

rating ➡ popularity

rid	name	rating	popularity
1	Mee Sum Pastry	3	Respectable
2	Café on the Ave	4	Poppin
3	Guanaco's Tacos	4	Poppin
4	Aladdin Gyro-Cery	5	Poppin

Usefulness of Keys in Design

Restaurants(rid, name, rating, popularity)

rid ➡ name

rid ➡ rating

Fine because rid is a superkey

rating ➡ popularity

rid	name	rating	popularity
1	Mee Sum Pastry	3	Respectable
2	Café on the Ave	4	Poppin
3	Guanaco's Tacos	4	Poppin
4	Aladdin Gyro-Cery	5	Poppin

Usefulness of Keys in Design

Restaurants(rid, name, rating, popularity)

rid ➡ name

rid ➡ rating

Fine because rid is a superkey

rating ➡ popularity

Redundancy!

rid	name	rating	popularity
1	Mee Sum Pastry	3	Respectable
2	Café on the Ave	4	Poppin
3	Guanaco's Tacos	4	Poppin
4	Aladdin Gyro-Cery	5	Poppin