Unit 6: Conceptual Design
E/R Diagrams
Integrity Constraints
BCNF

(3 lectures)
Introduction to Data Management
CSE 414

E/R Diagrams
Announcements

• HW6 due on Friday. Turn instances off!!!

• WebQuiz 6 due on Saturday
Class Overview

- Unit 1: Intro
- Unit 2: Relational Data Models and Query Languages
- Unit 3: Non-relational data
- Unit 4: RDMBS internals and query optimization
- Unit 5: Parallel query processing
- Unit 6: DBMS usability, conceptual design
  - E/R diagrams
  - Constraints
  - Schema normalization
- Unit 7: Transactions
- Unit 8: Advanced topics (time permitting)
Database Design

What it is:
• Starting from scratch, design the database schema: relation, attributes, keys, foreign keys, constraints etc

Why it’s hard
• The database will be in operation for a very long time (years). Updating the schema while in production is very expensive (why?)
Database Design

• Consider issues such as:
  – What entities to model
  – How entities are related
  – What constraints exist in the domain

• Several formalisms exist
  – We discuss E/R diagrams
  – UML, model-driven architecture

• Reading: Sec. 4.1-4.6
Database Design Process

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details
Physical Schema
Entity / Relationship Diagrams

- Entity set = a class
  - An entity = an object

- Attribute

- Relationship
Keys in E/R Diagrams

• Every entity set must have a key
What is a Relation?

- A mathematical definition:
  - if A, B are sets, then a relation R is a subset of A × B

A={1,2,3}, B={a,b,c,d},
A × B = {(1,a),(1,b),(1,c),(1,d),
(2,a),(2,b),(2,c),(2,d),
(3,a),(3,b),(3,c),(3,d)}

R = {(1,a), (1,c), (3,b)}

- makes is a subset of Product × Company:
Multiplicity of E/R Relations

- one-one:

- many-one

- many-many
What does this say?
Attributes on Relationships

What does this say?
Multi-way Relationships

How do we model a purchase relationship between buyers, products and stores?

Can still model as a mathematical set (How?)

As a set of triples $\subseteq \text{Product} \times \text{Person} \times \text{Store}$
Q: What does the arrow mean?

A: Any person buys a given product from at most one store

[Fine print: Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]

CSE 414 - 2019sp
Q: What does the arrow mean?

A: Any person buys a given product from at most one store AND every store sells to every person at most one product.
Converting Multi-way Relationships to Binary

Arrows go in which direction?
Converting Multi-way Relationships to Binary

Make sure you understand why!
3. Design Principles

What’s wrong?

Product — Purchase — Person

Country — President — Person

Moral: Be faithful to the specifications of the application!
Design Principles: What’s Wrong?

Moral: pick the right kind of entities.
Design Principles: What’s Wrong?

Moral: don’t complicate life more than it already is.
From E/R Diagrams to Relational Schema

- Entity set $\rightarrow$ relation
- Relationship $\rightarrow$ relation
**Product** (*prod-ID, category, price*)

<table>
<thead>
<tr>
<th>prod-ID</th>
<th>category</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo55</td>
<td>Camera</td>
<td>99.99</td>
</tr>
<tr>
<td>Pokemn19</td>
<td>Toy</td>
<td>29.99</td>
</tr>
</tbody>
</table>
Represent this in relations
N-N Relationships to Relations

Orders(\text{prod-ID}, \text{cust-ID}, \text{date})

Shipment(\text{prod-ID}, \text{cust-ID}, \text{name}, \text{date})

Shipping-Co(\text{name}, \text{address})

<table>
<thead>
<tr>
<th>prod-ID</th>
<th>cust-ID</th>
<th>name</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo55</td>
<td>Joe12</td>
<td>UPS</td>
<td>4/10/2011</td>
</tr>
<tr>
<td>Gizmo55</td>
<td>Joe12</td>
<td>FEDEX</td>
<td>4/9/2011</td>
</tr>
</tbody>
</table>
N-1 Relationships to Relations

Represent this in relations
Orders\((prod-ID, cust-ID, date1, name, date2)\)
Shipping-Co\((name, address)\)

Remember: no separate relations for many-one relationship
Multi-way Relationships to Relations

Product

| prod-ID | price |

Purchase

| prod-ID | ssn | name |

Store

| name | address |

Person

| ssn | name |

Purchase(prod-ID, ssn, name)
Modeling Subclasses

Some objects in a class may be special
  • define a new class
  • better: define a subclass

Products

Software products

Educational products

So --- we define subclasses in E/R
Subclasses

Product
  isa
    Software Product
    platforms
  isa
    Educational Product
    Age Group

name
category
price
Subclasses to Relations

![Diagram of subclass relations]

Product

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>99</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>49</td>
<td>photo</td>
</tr>
<tr>
<td>Toy</td>
<td>39</td>
<td>gadget</td>
</tr>
</tbody>
</table>

Software Product

<table>
<thead>
<tr>
<th>Name</th>
<th>platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>unix</td>
</tr>
</tbody>
</table>

Educational Product

<table>
<thead>
<tr>
<th>Name</th>
<th>Age Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>toddler</td>
</tr>
<tr>
<td>Toy</td>
<td>retired</td>
</tr>
</tbody>
</table>

Other ways to convert are possible

CSE 414 - 2019sp
Modeling Union Types with Subclasses

Say: each piece of furniture is owned either by a person or by a company
Modeling Union Types with Subclasses

Say: each piece of furniture is owned either by a person or by a company

Solution 1. Acceptable but imperfect (What’s wrong ?)
Modeling Union Types with Subclasses

Solution 2: better, more laborious
Weak Entity Sets

Entity sets are weak when their key comes from other classes to which they are related.

Team(sport, number, universityName)
University(name)
What Are the Keys of R?

A
B
R
S
T
V
C
E
W
U
D
Q
F
G
Z
K
Introduction to Data Management
CSE 414

Integrity Constraints
An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

Why?

How?
Integrity Constraints Motivation

An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

Why? Because we want application data to be consistent

How?
An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

Why? Because we want application data to be consistent.

How? The DBMS checks and enforces IC during updates.
Constraints in E/R Diagrams

- Keys
- Single-value constraints
- Referential integrity constraints
- General constraints
Keys in E/R Diagrams

Underline:

No formal way to specify multiple keys in E/R diagrams
Single Value Constraints

makes

vs.

makes
Referential Integrity Constraints

Each product made by at most one company. Some products made by no company

Each product made by **exactly** one company.
A Company entity is connected to at most 99 Product entities.
Constraints in SQL

- Keys
- Attribute-level, tuple-level constraints
- General (complex) constraints

The more complex the constraint, the harder it is to check and to enforce
Key Constraints

Product(name, category)

```
CREATE TABLE Product (  
name CHAR(30) PRIMARY KEY,  
category VARCHAR(20))
```

OR:

```
CREATE TABLE Product (  
name CHAR(30),  
category VARCHAR(20),  
PRIMAR KEY (name))
```
Keys with Multiple Attributes

Product(name, category, price)

CREATE TABLE Product (  
  name CHAR(30),  
  category VARCHAR(20),  
  price INT,  
  PRIMARY KEY (name, category))

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>10</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>20</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Photo</td>
<td>30</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>40</td>
</tr>
</tbody>
</table>
Other Keys

CREATE TABLE Product (  
productID CHAR(10),  
name CHAR(30),  
category VARCHAR(20),  
price INT,  
PRIMARY KEY (productID),  
UNIQUE (name, category))

There is at most one PRIMARY KEY; there can be many UNIQUE
CREATE TABLE Purchase (prodName CHAR(30)
REFERENCES Product(name),
date DATETIME)

prodName is a foreign key to Product(name)
name must be a key in Product

Referential integrity constraints

May write just Product if name is PK
Foreign Key Constraints

• Example with multi-attribute primary key

```
CREATE TABLE Purchase (  
  prodName CHAR(30),  
  category VARCHAR(20),  
  date DATETIME,  
  FOREIGN KEY (prodName, category)  
    REFERENCES Product(name, category)
```

• (name, category) must be a KEY in Product
What happens when data changes?

Types of updates:
• In Purchase: insert/update
• In Product: delete/update
What happens when data changes?

SQL policies for maintaining referential integrity:

- **NO ACTION** reject modifications (default)
- **CASCADE** after delete/update do delete/update
- **SET NULL** set foreign-key field to NULL
- **SET DEFAULT**

```sql
CREATE TABLE ... (pid int DEFAULT 42 REFERENCES...)
```
Constraints on Attributes and Tuples

• Constraints on attributes:
  NOT NULL -- obvious meaning...
  CHECK condition -- any condition!

• Constraints on tuples
  CHECK condition
CREATE TABLE User (  
  uid int primary key,  
  firstName text,  
  lastName text NOT NULL,  
  age int CHECK (age > 12 and age < 120),  
  email text,  
  phone text,  
  CHECK (email is not NULL or phone is not NULL)  
)
CREATE TABLE Purchase (  prodName CHAR(30)  CHECK (prodName IN (SELECT Product.name FROM Product),  date DATETIME NOT NULL)
General Assertions

CREATE ASSERTION myAssert CHECK (NOT EXISTS (SELECT Product.name FROM Product, Purchase WHERE Product.name = Purchase.prodName GROUP BY Product.name HAVING count(*) > 200))

But most DBMSs do not implement assertions. Because it is hard to support them efficiently, instead, they provide triggers.
Introduction to Data Management
CSE 414

Design Theory and BCNF
Announcements

• Monday is Memorial day – no lecture

• Webquiz 6 is due tomorrow

• HW6 is due tonight

• HW7 is posted, due next Friday.
What makes good schemas?

WHY SO MANY DATABASE TABLES???

I UPDATED A SCHEMA ONCE

IT SUCKED
Relational Schema Design

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

One person may have multiple phones, but lives in only one city

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?
Relational Schema Design

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies:

- **Redundancy** = repeat data
- **Update anomalies** = what if Fred moves to “Bellevue”?
- **Deletion anomalies** = what if Joe deletes his phone number?
Relation Decomposition

Break the relation into two:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies have gone:
- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)
Relational Schema Design (or Logical Design)

How do we do this systematically?

- Start with some relational schema
- Find out its *functional dependencies* (FDs)
- Use FDs to *normalize* the relational schema

CSE 414 - 2019sp

65
**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:

\[ A_1, A_2, \ldots, A_n \Rightarrow B_1, B_2, \ldots, B_m \]
Functional Dependencies (FDs)

**Definition** \( A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n \) holds in \( R \) if:

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

<table>
<thead>
<tr>
<th></th>
<th>( A_1 )</th>
<th>( \ldots )</th>
<th>( A_m )</th>
<th>( B_1 )</th>
<th>( \ldots )</th>
<th>( B_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t' )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

if \( t, t' \) agree here then \( t, t' \) agree here
Example

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

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

EmpID → Name, Phone, Position
Position → Phone
but not Phone → Position
### Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

**Position → Phone**
Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

But not Phone → Position
Example

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Red</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Stationary</td>
<td>Green</td>
<td>Office-supp.</td>
<td>59</td>
</tr>
</tbody>
</table>

Which FD’s hold?

name → color
category → department
color, category → price
department → price
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
Why bother with FDs?

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies:
- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?
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.
Closure of a set of Attributes

**Given** a set of attributes $A_1, \ldots, A_n$

The **closure** is the set of attributes $B$, notated $\{A_1, \ldots, A_n\}^+$, s.t. $A_1, \ldots, A_n \rightarrow B$

**Example:**
1. name $\rightarrow$ color
2. category $\rightarrow$ department
3. color, category $\rightarrow$ price

**Closures:**

$name^+ = \{\text{name, color}\}$

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

$\text{color}^+ = \{\text{color}\}$
Closure Algorithm

X={A1, ..., An}.

Repeat until X doesn’t change do:
  if B₁, ..., Bₙ → C is a FD and
  B₁, ..., Bₙ are all in X
  then add C to X.

Example:

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

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

Hence: name, category → color, department, price
Why do we care?

• The closure allows us to compute all FDs implied by a given FD; Here is how:

• To check if the FD implies $A \rightarrow B$
  – Compute $A^+$
  – Check if $B \subseteq A^+$
Example

In class:

\[ R(A,B,C,D,E,F) \]

- \[ A, B \rightarrow C \]
- \[ A, D \rightarrow E \]
- \[ B \rightarrow D \]
- \[ A, F \rightarrow B \]

Compute \( \{A,B\}^+ \)  \( X = \{A, B, \} \)

Compute \( \{A, F\}^+ \)  \( X = \{A, F, \} \)
Example

In class:

\( R(A,B,C,D,E,F) \)

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow E \\
B & \rightarrow D \\
A, F & \rightarrow B
\end{align*}
\]

Compute \( \{A,B\}^+ \) \( X = \{A, B, C, D, E\} \)

Compute \( \{A, F\}^+ \) \( X = \{A, F, \} \)
Example

In class:

\[ R(A,B,C,D,E,F) \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compute \( \{ A, B \}^+ \) \( X = \{ A, B, C, D, E \} \)

Compute \( \{ A, F \}^+ \) \( X = \{ A, F, B, C, D, E \} \)
Example

In class:

R(A, B, C, D, E, F)

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow E \\
B & \rightarrow D \\
A, F & \rightarrow B \\
\end{align*}
\]

Compute \( \{A, B\}^+ \), \( X = \{A, B, C, D, E\} \)

Compute \( \{A, F\}^+ \), \( X = \{A, F, B, C, D, E\} \)

What is the key of R?
Practice at Home

Find all FD’s implied by:

A, B → C
A, D → B
B → D
Practice at Home

Find all FD’s implied by:

- A, B → C
- A, D → B
- B → D

Step 1: Compute \(X^+\), for every \(X\):

- \(A^+ = A\), \(B^+ = BD\), \(C^+ = C\), \(D^+ = D\)
- \(AB^+ = ABCD\), \(AC^+ = AC\), \(AD^+ = ABCD\), \(BC^+ = BCD\), \(BD^+ = BD\), \(CD^+ = CD\)
- \(ABC^+ = ABD^+ = ACD^+ = ABCD\) (no need to compute—why?)
- \(BCD^+ = BCD\), \(ABCD^+ = ABCD\)
Find all FD’s implied by:

A, B → C
A, D → B
B → D

Step 1: Compute $X^+$, for every $X$:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$
$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$
$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute— why ?)
$BCD^+ = BCD$, $ABCD^+ = ABCD$

Step 2: Enumerate all FD’s $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

AB → CD, AD → BC, ABC → D, ABD → C, ACD → B
Keys

- A **superkey** is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute $B$, we have $A_1, \ldots, A_n \rightarrow B$

- A **key** is a minimal superkey
  - A superkey and for which no subset is a superkey
Computing (Super)Keys

• For all sets $X$, compute $X^+$

• If $X^+ = [\text{all attributes}]$, then $X$ is a superkey

• Try reducing to the minimal $X$’s to get the key
Example

Product(name, price, category, color)

name, category $\rightarrow$ price

category $\rightarrow$ color

What is the key?
Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key?

(name, category) + = { name, category, price, color }

Hence (name, category) is a key
Key or Keys?

We can we have more than one key!

What are the keys here?

A → B
B → C
C → A

A → BC
B → AC
Eliminating Anomalies

Main idea:

• $X \rightarrow A$ is OK if $X$ is a (super)key

• $X \rightarrow A$ is not OK otherwise
  – Need to decompose the table, but how?
There are no “bad” FDs:

**Definition.** A relation R is in BCNF if:

Whenever $X \rightarrow B$ is a non-trivial dependency, then $X$ is a superkey.

Equivalently:

**Definition.** A relation R is in BCNF if:

$\forall X$, either $X^+ = X$ (i.e., $X$ is not in any FDs) 
or $X^+ = [\text{all attributes}]$ (computed using FDs)
BCNF Decomposition Algorithm

Normalize(R)

find X s.t.: X ≠ X⁺ and X⁺ ≠ [all attributes]
if (not found) then “R is in BCNF”
let Y = X⁺ - X; Z = [all attributes] - X⁺
decompose R into R1(X ∪ Y) and R2(X ∪ Z)
Normalize(R1); Normalize(R2);
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

SSN → Name, City

The only key is: \{SSN, PhoneNumber\}
Hence SSN → Name, City is a “bad” dependency

In other words:
SSN+ = SSN, Name, City and is neither SSN nor All Attributes
Example BCNF Decomposition

Let’s check anomalies:
- Redundancy?
- Update?
- Delete?
Find $X$ s.t.: $X \neq X^+$ and $X^+ \neq \{\text{all attributes}\}$

Example BCNF Decomposition

$\text{Person(name, SSN, age, hairColor, phoneNumber)}$

- SSN $\rightarrow$ name, age
- age $\rightarrow$ hairColor
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age
age → hairColor

Iteration 1: 

Person: SSN+ = SSN, name, age, hairColor

Decompose into:

P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN \to name, age
age \to hairColor

Iteration 1:

Person: \quad SSN^+ = SSN, name, age, hairColor

Decompose into:
P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2:
P: age^+ = age, hairColor

Decompose:
People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

What are the keys?

Find X s.t.: X \neq X^+ and X^+ \neq [all attributes]
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age
age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
               Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor
Decompose: People(SSN, name, age)
           Hair(age, hairColor)
           Phone(SSN, phoneNumber)

Find X s.t.: X ≠ X+ and X+ ≠ [all attributes]

Note the keys!
Example: BCNF

R(A,B,C,D) → A
B → C
Example: BCNF

Recall: find $X$ s.t. $X \subset X^+ \subset [\text{all-attrs}]$
Example: BCNF

\( R(A,B,C,D) \)

\[ A^+ = ABC \neq ABCD \]
Example: BCNF

$R(A, B, C, D)$

$R(A, B, C, D) \quad A^+ = ABC \neq ABCD$

$R_1(A, B, C)$

$R_2(A, D)$
Example: BCNF

R(A,B,C,D)

A⁺ = ABC ≠ ABCD

R₁(A,B,C)
B⁺ = BC ≠ ABC

R₂(A,D)

A → B
B → C
Example: BCNF

R(A,B,C,D)

A^+ = ABC ≠ ABCD

R_1(A,B,C)

B^+ = BC ≠ ABC

R_{11}(B,C)

R_{12}(A,B)

R_2(A,D)

What are the keys?

What happens if in R we first pick B^+? Or AB^+?
Decompositions in General

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]

\[ S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]

\[ S_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\( S_1 = \) projection of \( R \) on \( A_1, \ldots, A_n, B_1, \ldots, B_m \)

\( S_2 = \) projection of \( R \) on \( A_1, \ldots, A_n, C_1, \ldots, C_p \)
Lossless Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

What is lossy here?

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Decomposition in General

$R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p)$

$S_1(A_1, ..., A_n, B_1, ..., B_m)$  $S_2(A_1, ..., A_n, C_1, ..., C_p)$

Let: $S_1 = \text{projection of } R \text{ on } A_1, ..., A_n, B_1, ..., B_m$
$S_2 = \text{projection of } R \text{ on } A_1, ..., A_n, C_1, ..., C_p$

The decomposition is called \textit{lossless} if $R = S_1 \bowtie S_2$

Fact: If $A_1, ..., A_n \rightarrow B_1, ..., B_m$ then the decomposition is lossless

It follows that every BCNF decomposition is lossless
Testing for Lossless Join

If we decompose $R$ into $\Pi_{S_1}(R)$, $\Pi_{S_2}(R)$, $\Pi_{S_3}(R)$, ... Is it true that $S_1 \bowtie S_2 \bowtie S_3 \bowtie ... = R$?

To check “=“ we need to check “$\subseteq$” and “$\supseteq$”

$R \subseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie ...$ always holds (why?)

$R \supseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie ...$ need to check
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

\[ R(A, B, C, D) = S_1(A, D) \bowtie S_2(A, C) \bowtie S_3(B, C, D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

\[ S_1 = \Pi_{AD}(R), \quad S_2 = \Pi_{AC}(R), \quad S_3 = \Pi_{BCD}(R) \]

Lossless?
The Chase Test for Lossless Join

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\[ S1 = \Pi_{AD}(R), \ S2 = \Pi_{AC}(R), \ S3 = \Pi_{BCD}(R) \]

\( R \subseteq S1 \bowtie S2 \bowtie S3 \)

To check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Lossless?
The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) \Join S2(A,C) \Join S3(B,C,D)
R satisfies: A\rightarrow B, B\rightarrow C, CD\rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R)
R \subseteq S1 \Join S2 \Join S3
To check: R \supseteq S1 \Join S2 \Join S3
Suppose (a,b,c,d) \in S1 \Join S2 \Join S3 Is it also in R?
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)
R satisfies: A \rightarrow B, B \rightarrow C, CD \rightarrow A
S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R)
R \subseteq S1 \bowtie S2 \bowtie S3
To check: R \supseteq S1 \bowtie S2 \bowtie S3
Suppose (a,b,c,d) \in S1 \bowtie S2 \bowtie S3 Is it also in R?
R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
</tbody>
</table>

Why?
(a,d) \in S1 = \Pi_{AD}(R)

Lossless?
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)
R satisfies: A \rightarrow B, B \rightarrow C, CD \rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R)
R \subseteq S1 \bowtie S2 \bowtie S3
To check: R \supseteq S1 \bowtie S2 \bowtie S3
Suppose (a,b,c,d) \in S1 \bowtie S2 \bowtie S3 Is it also in R?
R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
</tbody>
</table>

Lossless?
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)
R satisfies: A \rightarrow B, B \rightarrow C, CD \rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R)
R \subseteq S1 \bowtie S2 \bowtie S3
To check: R \supseteq S1 \bowtie S2 \bowtie S3
Suppose (a,b,c,d) \in S1 \bowtie S2 \bowtie S3  Is it also in R?
R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Why?

(a,d) \in S1 = \Pi_{AD}(R)
(a,c) \in S2 = \Pi_{BD}(R)
(b,c,d) \in S3 = \Pi_{BCD}(R)
Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

\[ S1 = \Pi_{AD}(R), \quad S2 = \Pi_{AC}(R), \quad S3 = \Pi_{BCD}(R) \]

\( R \subseteq S1 \bowtie S2 \bowtie S3 \)

To check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in R?

R must contain the following tuples:

“Chase” them (apply FDs):

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c1 & d \\
(a,d) \in S1 = \Pi_{AD}(R) \\
(a,c) \in S2 = \Pi_{BD}(R) \\
(b,c,d) \in S3 = \Pi_{BCD}(R) \\
\end{array}
\]

Why?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Lossless?
Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

\[ R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]

R satisfies: \( A \rightarrow B, B \rightarrow C, CD \rightarrow A \)

\( S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R) \)

\( R \subseteq S1 \bowtie S2 \bowtie S3 \)

To check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in \( R \)?

R must contain the following tuples:

“Chase” them (apply FDs):

A\( \rightarrow \)B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

B\( \rightarrow \)C

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Why ?

\((a,d) \in S1 = \Pi_{AD}(R)\)

\((a,c) \in S2 = \Pi_{BD}(R)\)

\((b,c,d) \in S3 = \Pi_{BCD}(R)\)
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

$R(A, B, C, D) = S_1(A, D) \bowtie S_2(A, C) \bowtie S_3(B, C, D)$

$R$ satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S_1 = \Pi_{AD}(R)$, $S_2 = \Pi_{AC}(R)$, $S_3 = \Pi_{BCD}(R)$

$R \subseteq S_1 \bowtie S_2 \bowtie S_3$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a, b, c, d) \in S_1 \bowtie S_2 \bowtie S_3$ is it also in $R$?

$R$ must contain the following tuples:

“Chase” them (apply FDs):

Why?

$$(a, d) \in S_1 = \Pi_{AD}(R)$$
$$(a, c) \in S_2 = \Pi_{BD}(R)$$
$$(b, c, d) \in S_3 = \Pi_{BCD}(R)$$

Hence $R$ contains $(a, b, c, d)$
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

$R(A, B, C, D) = S_1(A, D) \bowtie S_2(A, C) \bowtie S_3(B, C, D)$

$R$ satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

$S_1 = \Pi_{AD}(R)$, $S_2 = \Pi_{AC}(R)$, $S_3 = \Pi_{BCD}(R)$

$R \subseteq S_1 \bowtie S_2 \bowtie S_3$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a, b, c, d) \in S_1 \bowtie S_2 \bowtie S_3$ Is it also in $R$?

$R$ must contain the following tuples:

```
   A  B  C  D
  |---|---|---|
 a1 | b1 | c1 | d  |
 a2 | b2 | c  | d2 |
 a3 | b  | c  | d  |
```

“Chase” them (apply FDs):

```
  A → B
  a  b1  c1  d
  a  b1  c  d2
 a3 b  c  d

  B → C
  a  b1  c  d
  a  b1  c  d2
 a3 b  c  d

  CD → A
  a  b1  c  d
  a  b1  c  d2
 a3 b  c  d
```

Why?

$(a,d) \in S_1 = \Pi_{AD}(R)$

$(a,c) \in S_2 = \Pi_{BC}(R)$

$(b,c,d) \in S_3 = \Pi_{BCD}(R)$

Hence $R$ contains $(a, b, c, d)$
Schema Refinements  
= Normal Forms

• 1st Normal Form = all tables are flat
• 2nd Normal Form = obsolete
• Boyce Codd Normal Form = no bad FDs
• 3rd Normal Form = see book
  – BCNF removes anomalies, but my lose some FDs (see book 3.4.4)
  – 3NF preserves all FD’s, but may still have some anomalies
Conclusion

• E/R diagrams are means to structurally visualize and design relational schemas

• Normalization is a principled way of converting schemas into a form that avoid such redundancies.

• BCNF and 3NF are the most widely used normalized form in practice