Designing Schemas

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor
Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Identify the anomalies that a bad schema can cause.
- State the key issues for schema design.

Is This Schema Good?

- Schema: Students(id, name, major, birthdate)
- Instance:

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

How If Double (or Triple) Majors?

- Schema: Students(id, name, major, birthdate)
- Instance:

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

Anomalies

- Redundancy

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

- Update anomaly
  - What is Bugs Bunny's birthday?

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2006-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

How If No Major (Undeclared) Yet?

- Schema: Students(id, name, major, birthdate)
- Instance:

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
~~4~~	~~Mickey Mouse~~	~~CS~~	~~1995-04-01~~

Example Students table

Problems: Storing Birthday with Major

- When a student has exactly one major
  - The schema is good.

- When a student can have multiple or no majors
  - The same schema is bad.
  - Problems:
    - Redundancy
    - Loss of facts
    - Update anomaly

- The goodness of a schema depends on the domain characteristics.

Questions We Must Ask

- How do we consider a schema good?
  - A good schema is called a **normal form**.
  - There are various normal forms.
  - We will discuss Boyce-Codd Normal Form (BCNF).

- How do we transform a bad schema into a good one?
  - That is, to **normalize** a schema to a normal form.

Functional Dependencies

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define what functional dependency means.
- Give examples of functional dependencies.

Functional Dependencies

- A form of constraint, and therefore part of a schema.

- A key factor for determining and organizing good schemas.

About Major and Birthday

- How many majors do you have?
  - None, one, or multiple
  - **id ⟶ major?**
- How many birthdays do you have?
  - One
  - **id ⟶ birthday?**

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

Functional Dependency (FD)

- Notation: $A_1, \ldots, A_m \longrightarrow B_1, \ldots, B_n$
- We say: $A_1, \ldots, A_m$ **functionally determines** $B_1, \ldots, B_n$.
- Meaning:
  - If any tuples agree on $A_1, \ldots, A_m$ values, then they must also agree on $B_1, \ldots, B_n$.
  - I.e., the mapping from $A_1, \ldots, A_m$ to $B_1, \ldots, B_n$ is **functional** (many-one).
- Whether FDs hold is your **knowledge/assumption** of the domain.
  - id $\longrightarrow$ birthday
  - id, course $\longrightarrow$ grade

A " **Functionally Determines" B**

- Given a value of A, there exists at most one value of B-- no ambiguity.

- It does not mean: B can be computed from A by a formula.
  - For id → birthday, you cannot compute birthday from id.

- It does not mean: B can be easily found by A.
  - For id → birthday, you may not be able to identify the birthday if it is not disclosed to you.

FD: Your Domain Knowledge/Assumption

- Understand the domain and make assumptions accordingly.

- id ⟶ major
  - Possibly true, if a student can have only one major
- id, course ⟶ grade
  - Possibly false, if a student can take a course multiple times

Keys

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define key in terms of functional dependencies.
- Distinguish the notions of key and superkey.

What Is a Key?

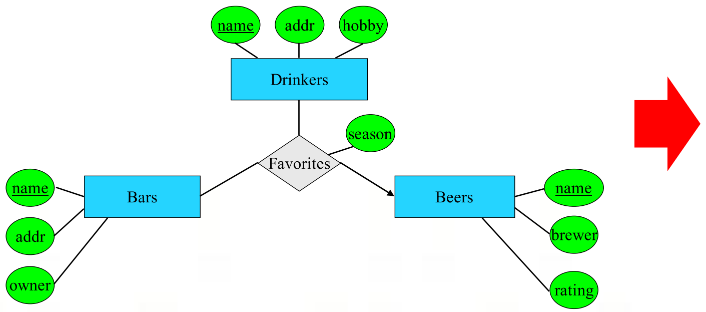
- After defining FDs, we can now define keys formally.
- **Key** of a relation $R(A_1, \ldots, A_n)$ is a set of attributes K that
  - Functionally determines all attributes of R, $K \longrightarrow A_1, \ldots, A_n$, and
  - None of its subsets does
- Superkey
  - A set of attributes that contains a key

Explaining the Key Rule in ER Translation

- Recall this rule:

Relationship → Relation

Rule: Translating relationship X of E_1 … , E_n to relation R
- Attributes of R = key attributes of E_1, … , E_n plus attributes of X
- Key of R = key of E_1, … , E_n except those "arrowed" entities



Favorites(<u>drinker</u>, <u>bar</u>, beer, season)

Translating relationship into relation

- Why?
  - {drinker, bar, beer} is not a key, since {drinker, bar} already is.
  - {drinker, bar, beer} is a superkey.

Reasoning with FDs

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

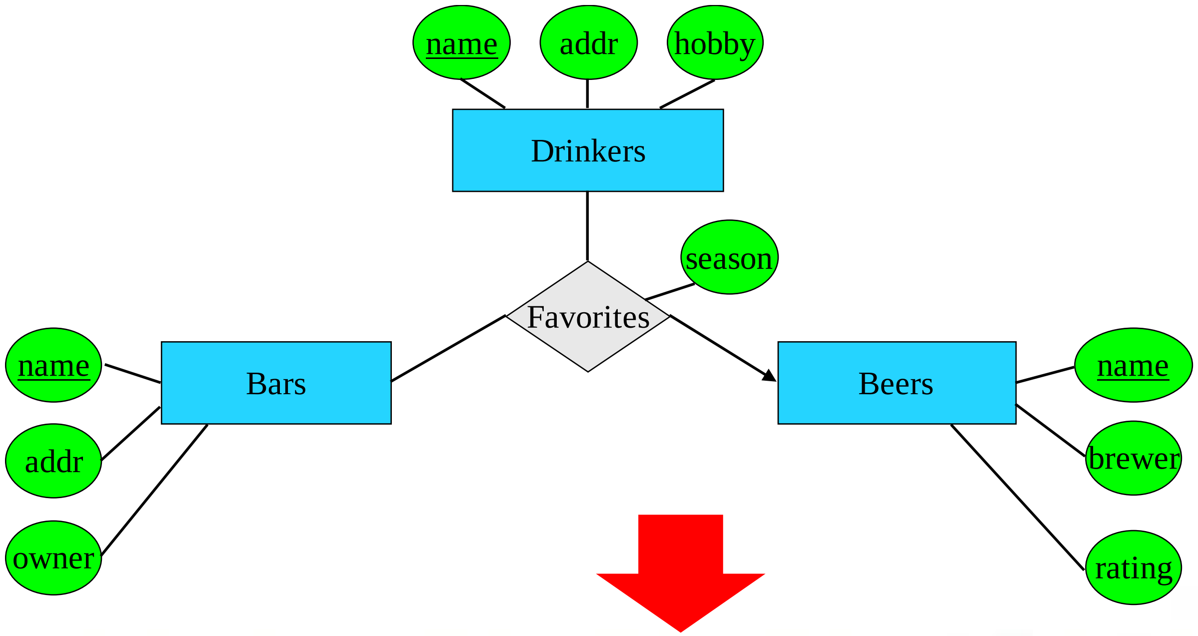
Learning Objectives

By the end of this video, you will be able to:

- Describe why reasoning with FDs is useful.
- State Armstrong's Axioms and their derived rules.
- Determine if an FD holds based on a given set of FDs.

Why Do I Need to Reason?

- Recall the Favorites relation:
- How to know {drinker, bar} is a key?
- We are given
  - drinker, bar, beer ⟶ season
  - drinker, bar ⟶ beer



Favorites(**<u>drinker</u>**, **<u>bar</u>**, beer, season)

Translating the Favorites relationship to a relation

- Can we reason that
  - drinker, bar ⟶ drinker, beer, bar, season?
  - And {drinker} or {bar} does not.
- If so, then we know {drinker, bar} is key!

Basic Rules: Armstrong's Axioms

- Reflexivity rule
  - If $B \subseteq A$, then $A \longrightarrow B$.
- Augmentation rule
  - If $A \longrightarrow B$, then $AC \longrightarrow BC$.
- Transitivity rule
  - If $A \longrightarrow B$ and $B \longrightarrow C$, then $A \longrightarrow C$.

Deriving More Rules

- Splitting rule
  - If $A \longrightarrow BC$, then $A \longrightarrow B$, and $A \longrightarrow C$.
  - Derivation
    - $\because A \longrightarrow BC$ and $BC \longrightarrow B$ (by reflexivity)
    - $\therefore A \longrightarrow B$ (by transitivity)
    - Similarly for $A \longrightarrow C$

- Combining rule
  - If $A \longrightarrow B$, and $A \longrightarrow C$, then $A \longrightarrow BC$.
  - Derivation
    - $\because A \longrightarrow AB$ (by augmentation) and $AB \longrightarrow BC$ (by augmentation)
    - $\therefore A \longrightarrow BC$ (by transitivity)

Reasoning: Is {drinker, bar} a Superkey?

- Suppose Favorites(drinker, bar, beer, season, price)
- **Given:**
  - **drinker, bar, beer ⟶ season**
  - **drinker, bar ⟶ beer**
  - **bar, beer ⟶ price**
- **Decide: Is {drinker, bar} a superkey?**

- To determine, can we reason:
- Is **drinker, bar ⟶ drinker, bar, beer, season, price** an FD?
- If it is an FD, then YES.

Reasoning: *Is* **drinker, bar ⟶ all** *an FD?*

- 1. drinker, bar ⟶ drinker, bar (reflexivity)
- 2. drinker, bar ⟶ beer (given)
- 3. drinker, bar ⟶ season
  - drinker, bar ⟶ drinker, bar, beer (augmenting 2)
  - drinker, bar, beer ⟶ season (given)
  - drinker, bar ⟶ season (transitivity)
- 4. drinker, bar ⟶ price
  - How to derive?

- **Given:**
  - **drinker, bar, beer ⟶ season**
  - **drinker, bar ⟶ beer**
  - **bar, beer ⟶ price**
- **Decide: Is {drinker, bar} a Superkey?**

- **5. drinker, bar ⟶ drinker, bar, beer, season, price** (combining all above)

Attribute Closures

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define the concept of attribute closure.
- Describe how and why attribute closure is useful.
- Find the closure of a set of attributes given a set of FDs.
- Determine keys of a relation using attribute closures.

Reasoning: Is {drinker, bar} a Superkey?

- We are asking:
  - Can {drinker, bar} determine all attributes?
- We may ask, more fundamentally:
  - What attributes can {drinker, bar} determine?
  - This is called the **closure** of {drinker, bar}.

- **Given:**
  - **drinker, bar, beer $\longrightarrow$ season**
  - **drinker, bar $\longrightarrow$ beer**
  - **bar, beer $\longrightarrow$ price**
- **Decide: Is {drinker, bar} a Superkey?**

Closure of Attributes

- Problem: *What can these attributes determine?*
  - Given a set of attributes $\{A_1, \ldots, A_n\}$ and a set of dependencies F
  - Find all attributes $B_1, \ldots, B_m$ such that any relation that satisfies F also satisfies:

    $$A_1, \ldots, A_n \longrightarrow B_1, \ldots, B_m$$
  - The closure of $\{A_1, \ldots, A_n\}$ is $B_1, \ldots, B_m$, i.e.,

    $$\{A_1, \ldots, A_n\}^+ = \{B_1, \ldots, B_m\}$$

- **Given:**
  - **drinker, bar, beer ⟶ season**
  - **drinker, bar ⟶ beer**
  - **bar, beer ⟶ price**
- **Decide: Is {drinker, bar} a Superkey?**

- Ex: What can {drinker, bar} determine with the given FDs?
  - $\{\text{drinker, bar}\}^+$

    $= \{\text{drinker, beer, beer, season, price}\}$

Finding Attribute Closures

- Given a set of attributes $\{A_1, \ldots, A_n\}$ and a set of dependencies F
- $C = \{A_1, \ldots, A_n\}$
- Repeat until C does not change:
  - If $X_1, \ldots, X_m \longrightarrow Y$ is in F, and $X_1, \ldots, X_m$ are all in C, and Y not in C:
    - $C := C + Y$

- Ex: {drinker, bar }$^+$ = ?
  - C = {drinker, bar}
  - Add beer, ∵ drinker, bar ⟶ beer
  - Add season, ∵ drinker, bar, beer ⟶ season
  - Add price, ∵ bar, beer ⟶ price
  - C = {drinker, bar, beer, season, price}

- **Given:**
  - **drinker, bar, beer ⟶ season**
  - **drinker, bar ⟶ beer**
  - **bar, beer ⟶ price**
- **Decide: Is {drinker, bar} a Superkey?**

Reasoning: {drinker, bar} Is a Key

- $\{\text{drinker, bar}\}^{+} =$ {drinker, bar, beer, season, price}
  - So, {drinker, bar} is a superkey.

- $\{\text{drinker}\}^{+} =$ {drinker}
  - So, {drinker} is not a superkey.

- $\{\text{bar}\}^{+} =$ {bar}
  - So, {bar} is not a superkey.

- So, {drinker, bar} is a key!

Food for Thought

Can you use attribute closure to determine if an FD $A \rightarrow B$ holds? How?

Normal Forms

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Describe what normal forms mean.
- Explain why the concept of normal forms is useful for designing schemas.

Recall the Problematic Schema

- Why is it bad?
- A student’s name and birthday are repeated for each major.
- So? We should not store name and birthday with major.

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

A Better Design

Students

id	name	birthday
1	Bugs Bunny	2004-11-06
2	Donald Duck	1997-02-01
3	Peter Pan	1998-10-01
4	Mickey Mouse	1995-04-01

Majors

id	major
1	CS
1	Music
2	Bio
3	Econ
3	Social
3	ME
4	CS

Example Students and Majors tables

- Redundancy?
- Update anomaly?
- Delete anomaly?

Schema Refinement: Desired Properties

- Minimize redundancy.
- Avoid info loss.
- Preserve dependency.
- Ensure good query performance.

First Normal Form

- From the very beginning, when relational model was defined
- **First Normal Form (1NF)**: Each attribute contains only single atomic values.
- As proposed by E. F. Codd:

  E. F. Codd (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM, 13*(6): 377-387

employee (*man#*, name, birthdate, jobhistory, children)
jobhistory (*jobdate*, title, salaryhistory)
salaryhistory (*salarydate*, salary)
children (*childname*, birthyear)

FIG. 3(a). Unnormalized set

employee′ (*man#*, name, birthdate)
jobhistory′ (*man#*, *jobdate*, title)
salaryhistory′ (*man#*, *jobdate*, *salarydate*, salary)
children′ (*man#*, *childname*, birthyear)

FIG. 3(b). Normalized set

Examples used in Codd’s paper on relational model

Normal Forms

- **First Normal Form (1NF)**: Each attribute contains only single atomic values.
- Many normal forms were proposed, as more desired properties were discovered.
  - 2NF, 3NF, BCNF, 4NF, …
- We will study BCNF – Boyce Codd Normal Forms (and 3NF, 4NF).

Boyce-Codd Normal Form

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

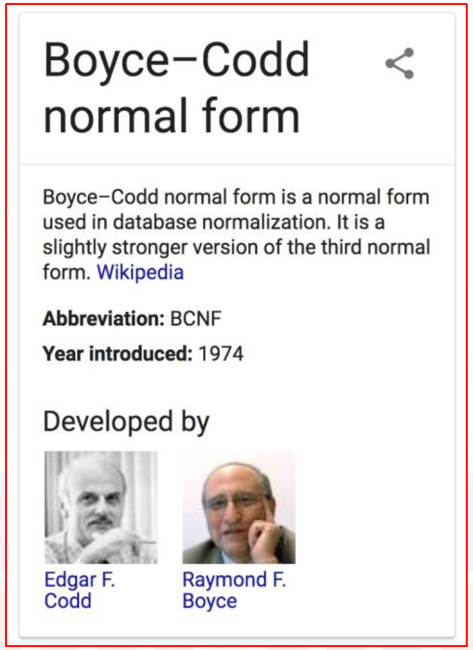
Learning Objectives

By the end of this video, you will be able to:

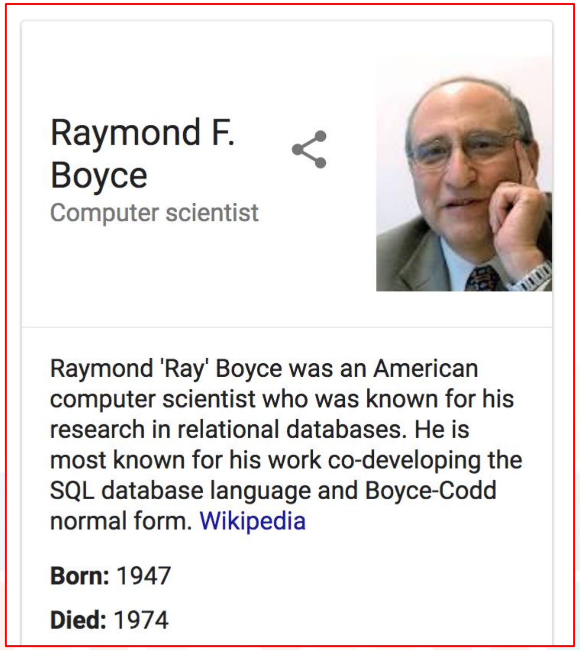
- Define what BCNF is.
- Determine whether a given relation is in BCNF.
- Transform a relation into BCNF with BCNF decomposition.

Towards BCNF: Boyce and Codd

- **Codd** invented the relational model, created the era of declarative (in contrast to procedural) data management.
- **Boyce** -- in addition to BCNF, he has a major contribution to make declarative data management a reality. *Stay tuned!!*



Google search result of "Boyce Codd Normal Form"



Google search result of "Raymond F. Boyce"

Recall the Problematic Schema

- Why is it bad?
- A student’s name and birthday are repeated for each major.
- So? We should not store name and birthday with major.

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

What Is the Culprit?

- Why would name and birthday repeat, but major does not?

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

- id $\longrightarrow$ name, birthday
- id $\not\longrightarrow$ major

- id determines some attributes A.
- id does not determine other attributes; i.e., it is not a superkey.
- Thus, A will repeat!
- Lesson: *An FD by non-key attributes can cause redundancy.*

Boyce-Codd Normal Form

- A relation R is in BCNF if and only if:

  Whenever there is a **nontrivial FD** for *R*,
  $$A \longrightarrow B$$
  then A is a superkey for *R*.

- Whenever a set of attributes of R determines another attribute, it should determine all attributes of R.

- That is, no bad FDs!

BCNF or Not?

- Likes(name, addr, likeBeer)
  - name $\longrightarrow$ addr
  - name $\not\longrightarrow$ likeBeer
- BCNF?

name	addr	likeBeer
Alex	100 Green St	Sam Adams
Bob	300 Purple St	Sam Adams
Carissa	200 Green St	Bud Light
Alex	100 Green St	Coors

Example Likes table

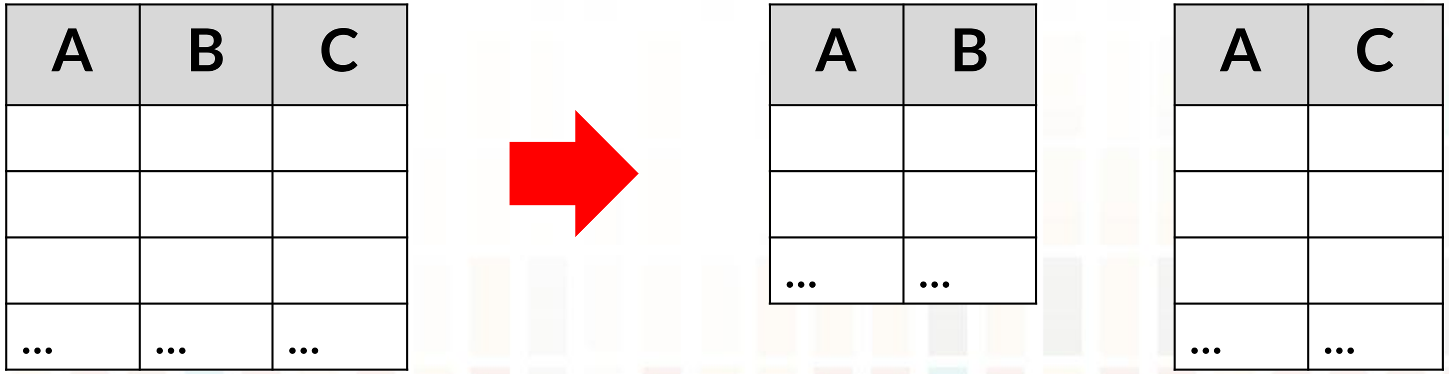
- Favorites(name, addr, favoriteBeer)
  - name $\longrightarrow$ addr
  - name $\longrightarrow$ favoriteBeer
- BCNF?

name	addr	favoriteBeer
Alex	100 Green St	Sam Adams
Carissa	200 Green St	Bud Light
Alex	100 Green St	Coors

Example Favorites table

BCNF Decomposition

- **Algorithm BCNF**
- Input: Relation R, FDs F
- If (exists an FD $A \longrightarrow B$ that violates the BCNF condition)
  - Decompose $R(A, B, C)$ into $R_1(A, B)$ and $R_2(A, C)$.
  - Compute FDs for R_1 and R_2 as F_1 and F_2.
  - Return BCNF$(R_1, F_1) \cup$ BCNF(R_2, F_2).
- Else
  - Return R.



BCNF decomposition

Decomposing into BCNF

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students table

- R = (id, name, major, birthday)
- F = {id ⟶ name, id ⟶ birthday}

- FD f: id ⟶ name, birthday violates BCNF
- Decompose into:
  - R_1 = (id, name, birthday), F_1 = {id ⟶ name, id ⟶ birthday}
  - R_2 = (id, major), $F_2 = \emptyset$
- Done?

Decomposing into BCNF

- $R =$ (id, name, major, birthday, adviser)
- $F =$ {id $\longrightarrow$ name, id $\longrightarrow$ birthday, major $\longrightarrow$ adviser}

- FD f: id $\longrightarrow$ name, birthday violates BCNF
- Decompose into:
  - $R_1 =$ (id, name, birthday), $F_1 =$ {id $\longrightarrow$ name, id $\longrightarrow$ birthday}
  - $R_2 =$ (id, major, adviser), $F_2 =$ {major $\longrightarrow$ adviser}
- Done?

Food for Thought

Is BCNF unique? I.e., given a relation R and FDs F, does BCNF decomposition results in a unique BCNF?

- Students(id, name, phone)
- Suppose: id → name, name → id

- What BCNF exists?

Lossless Decomposition

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define lossless decomposition, and explain why it is desired.
- Explain why BCNF decomposition is lossless.

A Decomposition Can Be Lossy

- Favorites(drinker, bar, beer)

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

Example Favorites relation

- Decomposition: $R_1 =$

drinker	bar
Alex	John Bar
Carissa	Green Bar
Alex	Purple Bar

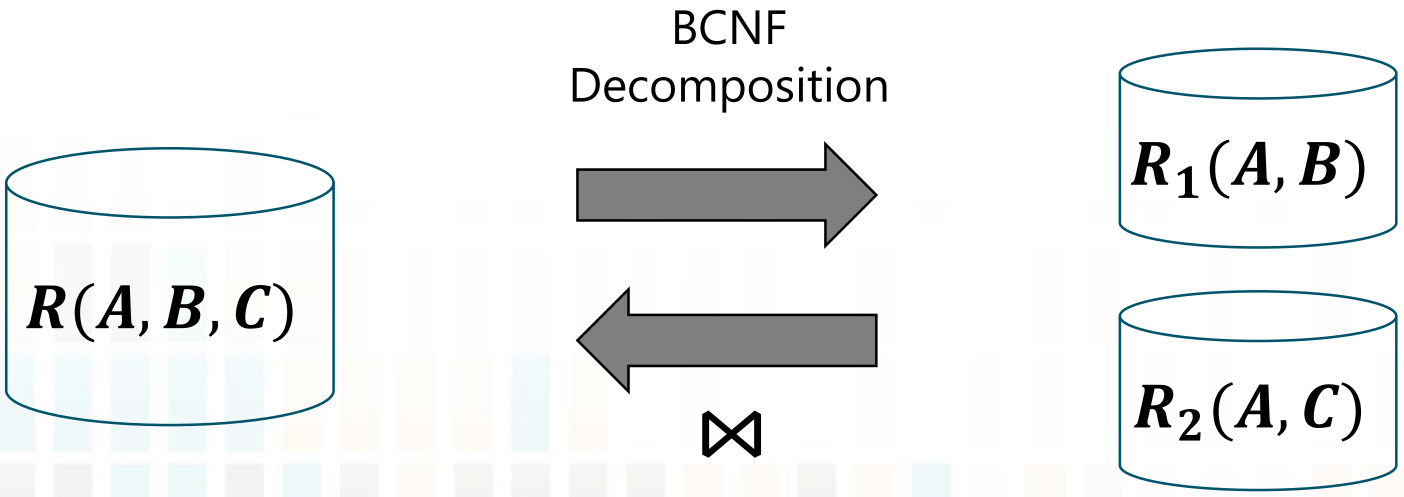
drinker	beer
Alex	Sam Adams
Carissa	Bud Light
Alex	Coors

Example decomposition of the Favorites relation

- Favorites $= R_1 \bowtie R_2$?
- No! We got extra tuples:
  - (Alex, John Bar, Coors)
  - (Alex, Purple Bar, Sam Adams)

BCNF: Lossless Decomposition

- That is, we reconstruct the original relation.
- If we decompose $R(A, B, C)$ into $R_1(A, B)$ and $R_2(A, C)$, due to $A \longrightarrow B$, then $R = R_1 \bowtie R_2$.



BCNF Decomposition is lossless

BCNF Decomposition Is Lossless

- Reconsider the lossy example: (drinker, bar, beer).
- Will BCNF decompose to (drinker, bar), (drinker, beer)?
  - Yes, but only when drinker → bar or drinker → beer.
- Suppose drinker → bar?
- Suppose drinker → beer?

- Favorites(drinker, bar, beer)

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

- Decomposition: $R_1 =$

drinker	bar
Alex	John Bar
Carissa	Green Bar
Alex	Purple Bar

$R_2 =$

drinker	beer
Alex	Sam Adams
Carissa	Bud Light
Alex	Coors

Example decomposition of the Favorites relation

- Favorites $= R_1 \bowtie R_2$?
- No! We got extra tuples:
  - (Alex, John Bar, Coors)
  - (Alex, Purple Bar, Sam Adams)

Dependency-Preserving Decomposition

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define dependency preserving, and explain why it is desired.
- Explain why BCNF may not be dependency preserving.

A Decomposition May Not Preserve Dependency

- A schema S is dependency preserving if, for every FD f of it:
  - f can be checked in a table T in S, or
  - f can be implied by those FDs that can be checked in single tables.

- Favorites(drinker, bar, beer)
- f_1: beer → bar, f_2: drinker, bar → beer
- Decomposition:
  $R_1 = $ (beer, bar), $R_2 = $ (beer, drinker)
- Preserve dependencies?
  - f_1: beer → bar
  - f_2: drinker, bar → beer

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

beer	bar
Sam Adams	John Bar
Bud Light	Green Bar
Coors	Purple Bar

beer	drinker
Sam Adams	Alex
Bud Light	Carissa
Coors	Alex

Example decomposition

BCNF May **Not** Preserve Dependency

- Favorites(drinker, bar, beer)
- f_1: beer → bar, f_2: drinker, bar → beer
- Decomposition:

  $R_1 = $ (beer, bar), $R_2 = $ (beer, drinker)

- Yes, this is lossless.
- But it does not preserve dependency!

drinker	**bar**	**beer**
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

beer	**bar**
Sam Adams	John Bar
Bud Light	Green Bar
Coors	Purple Bar

beer	**drinker**
Sam Adams	Alex
Bud Light	Carissa
Coors	Alex

Example decomposition

Food for Thought

For a relation R and FDs F, a dependency-preserving BCNF may not exist. Agree?

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

beer	bar
Sam Adams	John Bar
Bud Light	Green Bar
Coors	Purple Bar

beer	drinker
Sam Adams	Alex
Bud Light	Carissa
Coors	Alex

Example decomposition

Any BCNF that would preserve all dependencies?

f_1: beer → bar,
f_2: drinker, bar → beer

Third Normal Form

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define third normal form and describe when and why it is desired.
- Determine if a given relation is in 3NF.
- Settle for 3NF-- happily-- if your BCNF decomposition does not give you a dependency-preserving schema.

Since BCNF May Not Preserve FDs

- What happens if we cannot find a BCNF that preserves FDs?

Food for Thought

For a relation R and FDs F, a dependency-preserving BCNF may not exist. Agree?

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

beer	bar
Sam Adams	John Bar
Bud Light	Green Bar
Coors	Purple Bar

beer	drinker
Sam Adams	Alex
Bud Light	Carissa
Coors	Alex

Example decomposition

Any BCNF that would preserve all dependencies?

f_1: beer → bar,
f_2: drinker, bar → beer

3NF: Making a Compromise from BCNF

- A relation R is in 3NF if and only if:

  Whenever there is a **nontrivial FD** for *R*,
  $$A \longrightarrow B$$
  then A is a superkey for *R*,
  or *B* is a prime attribute (i.e., a member of a key) for *R*.

Settling for 3NF

- Favorites(drinker, bar, beer)
- f_1: beer → bar, f_2: drinker, bar → beer

- It is in 3NF.
  - Key is {drinker, bar}.
  - f_1: beer → bar – Ok, because bar is part of a key.
  - f_2: drinker, bar → beer – Ok, because drinker, bar is a superkey.

- It is not in BCNF, though.
  - Since BCNF would not preserve FD, we will settle for 3NF.

Why Is 3NF Acceptable?

- 3NF decomposition is both:
  - Lossless decomposition
  - Dependency preserving

- It removes "bad FDs" mostly
  - Except those involving key attributes
  - I.e., will not split a key in two relations

- Favorites(drinker, bar, beer)
- f_1: beer → bar, f_2: drinker, bar → beer

drinker	**bar**	**beer**
Alex	Joe's Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Fancy Bar	Coors
Bob	Green Bar	Bud Light

Example Favorites relation

Multivalued Dependencies

Designing Schemas

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Kevin C.C. Chang, Professor

Computer Science @ Illinois

Learning Objectives

By the end of this video, you will be able to:

- Define multi-valued dependencies.
- Describe why MVD may cause redundancies.
- Explain why FD is a special case of MVD.

More Than Functional Dependencies

- There are other kinds of dependencies than FDs.
  - Multivalued dependencies (MVD)
  - Inclusion dependencies (IND)
  - Join dependencies (JD)

- We will take a look at MVD.
  - So you can say you know about (multiple kinds of) dependenc**ies**!

Consider Our Academic World

- We know that id determines birthday, id → birthday.

- But, doesn't id also determine majors?
  - Donald Duck? Majors = {Bio}.
  - Bugs Bunny? Majors = {CS, Music}.

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students relation

- So, id determines major – as **a set of** majors instead of a unique value.
- We call this determination a multivalued dependency: id ↠ major.

MVD Is “Tuple Generating”

- Students(id, name, major, gpa, hobby, level)
- Suppose id $\twoheadrightarrow$ major, gpa: Do you think this table is “complete”?

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
…	…	…	…	…	…

Example Students relation

Multivalued Dependency (MVD)

- Notation: $A_1, \ldots, A_m \twoheadrightarrow B_1, \ldots, B_n$
- We say: $A_1, \ldots, A_m$ **multidetermines** $B_1, \ldots, B_n$
- Meaning:
  - If two tuples agree on $A_1, \ldots, A_m$ values, then swapping their $B_1, \ldots, B_n$ values will result in two tuples that are also in the relation.
  - I.e., *B* depends only on *A*, and is independent of the remaining attributes.
- Ex: id ↠ major, gpa

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
1	Bugs Bunny	Music	3.5	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
…	…	…	…	…	…

Example Students relation

FD: Special Case of MVD

- FD is a special case of MVD.
- id → birthday ⟹ id ↠ birthday

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Example Students relation

FD/MVD Are Domain Knowledge

- What FDs/MVDs hold is your **knowledge** of the domain.
- id → birthday? id ↠ birthday?
- id → major? id ↠ major?
- age → major? age ↠ major?

Food for Thought

How do you suggest to normalize the problematic Students relation to eliminate MVDs?

Yes, you probably have (re-)invented 4NF!

Fourth normal form

Fourth normal form is a normal form used in database normalization. Introduced by Ronald Fagin in 1977, 4NF is the next level of normalization after Boyce–Codd normal form. Wikipedia

Abbreviation: 4NF

Developed by: Ronald Fagin

Year introduced: 1977

Google search result of 4NF

You need a magnifier to view it (Pixabay, 2017)

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
1	Bugs Bunny	Music	3.5	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
...	...	...	...	...	...

=

id	name
1	Bugs Bunny
2	Donald Duck
3	Peter Pan
...	...

⋈

id	hobby	level
1	Tennis	Beginner
1	Chess	Advanced
2	Basketball	Intermediate
3	Piano	Beginner
3	Reading	Advanced
3	Swimming	Advanced
...	...	...

⋈

id	major	gpa
1	CS	3.0
1	Music	3.5
2	Bio	3.2
3	Econ	2.8
3	Social	3.0
3	ME	3.6
...	...	...

Sample answer for this FoT

References

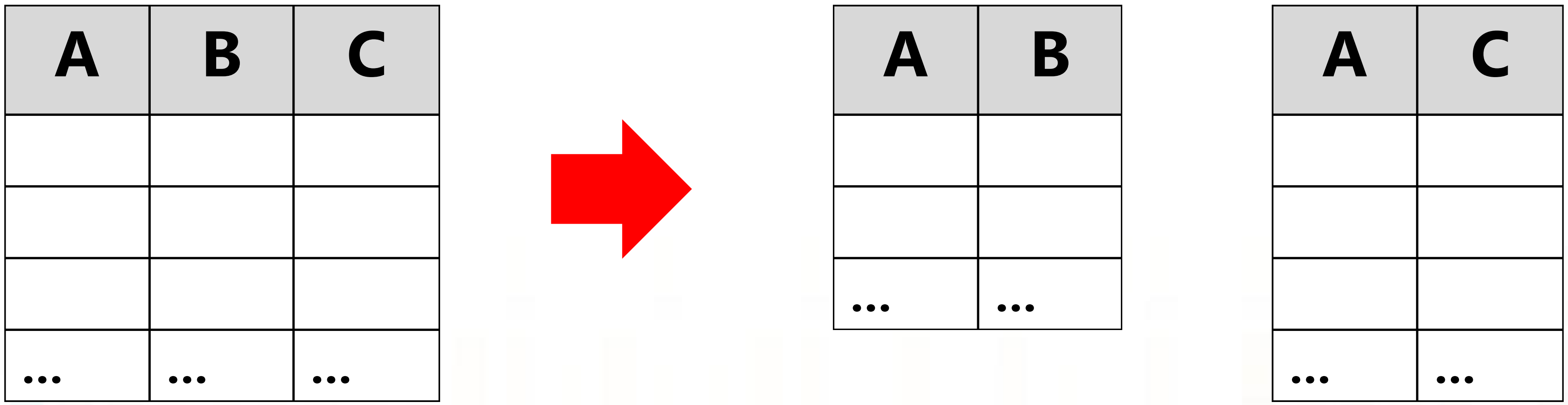
- Pixabay, 2017. *Image of a magnifier*[Online image]. Retrieved from https://pixabay.com/en/inspector-man-detective-male-160143/.

The End

Figure

id	name	major	birthday
1	Bugs Bunny	CS	2004-11-06
1	Bugs Bunny	Music	2004-11-06
2	Donald Duck	Bio	1997-02-01
3	Peter Pan	Econ	1998-10-01
3	Peter Pan	Social	1998-10-01
3	Peter Pan	ME	1998-10-01
4	Mickey Mouse	CS	1995-04-01

Figure



Figure

drinker	bar	beer
Alex	John Bar	Sam Adams
Carissa	Green Bar	Bud Light
Alex	Purple Bar	Coors

beer	bar
Sam Adams	John Bar
Bud Light	Green Bar
Coors	Purple Bar

beer	drinker
Sam Adams	Alex
Bud Light	Carissa
Coors	Alex

Figure

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
1	Bugs Bunny	Music	3.5	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
…	…	…	…	…	…

Figure

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
…	…	…	…	…	…

id	name
1	Bugs Bunny
2	Donald Duck
3	Peter Pan
...	...

Figure

id	major	gpa
1	CS	3.0
1	Music	3.5
2	Bio	3.2
3	Econ	2.8
3	Social	3.0
3	ME	3.6
…	…	…

Figure

id	hobby	level
1	Tennis	Beginner
1	Chess	Advanced
2	Basketball	Intermediate
3	Piano	Beginner
3	Reading	Advanced
3	Swimming	Advanced
…	…	…

Figure

id	name	major	gpa	hobby	level
1	Bugs Bunny	CS	3.0	Tennis	Beginner
1	Bugs Bunny	Music	3.5	Tennis	Beginner
1	Bugs Bunny	CS	3.0	Chess	Advanced
1	Bugs Bunny	Music	3.5	Chess	Advanced
2	Donald Duck	Bio	3.2	Basketball	Intermediate
3	Peter Pan	Econ	2.8	Piano	Beginner
3	Peter Pan	Social	3.0	Reading	Advanced
3	Peter Pan	ME	3.6	Swimming	Advanced
…	…	…	…	…	…

=

id	name
1	Bugs Bunny
2	Donald Duck
3	Peter Pan
…	…

$\bowtie$

id	hobby	level
1	Tennis	Beginner
1	Chess	Advanced
2	Basketball	Intermediate
3	Piano	Beginner
3	Reading	Advanced
3	Swimming	Advanced
…	…	…

$\bowtie$

id	major	gpa
1	CS	3.0
1	Music	3.5
2	Bio	3.2
3	Econ	2.8
3	Social	3.0
3	ME	3.6
…	…	…