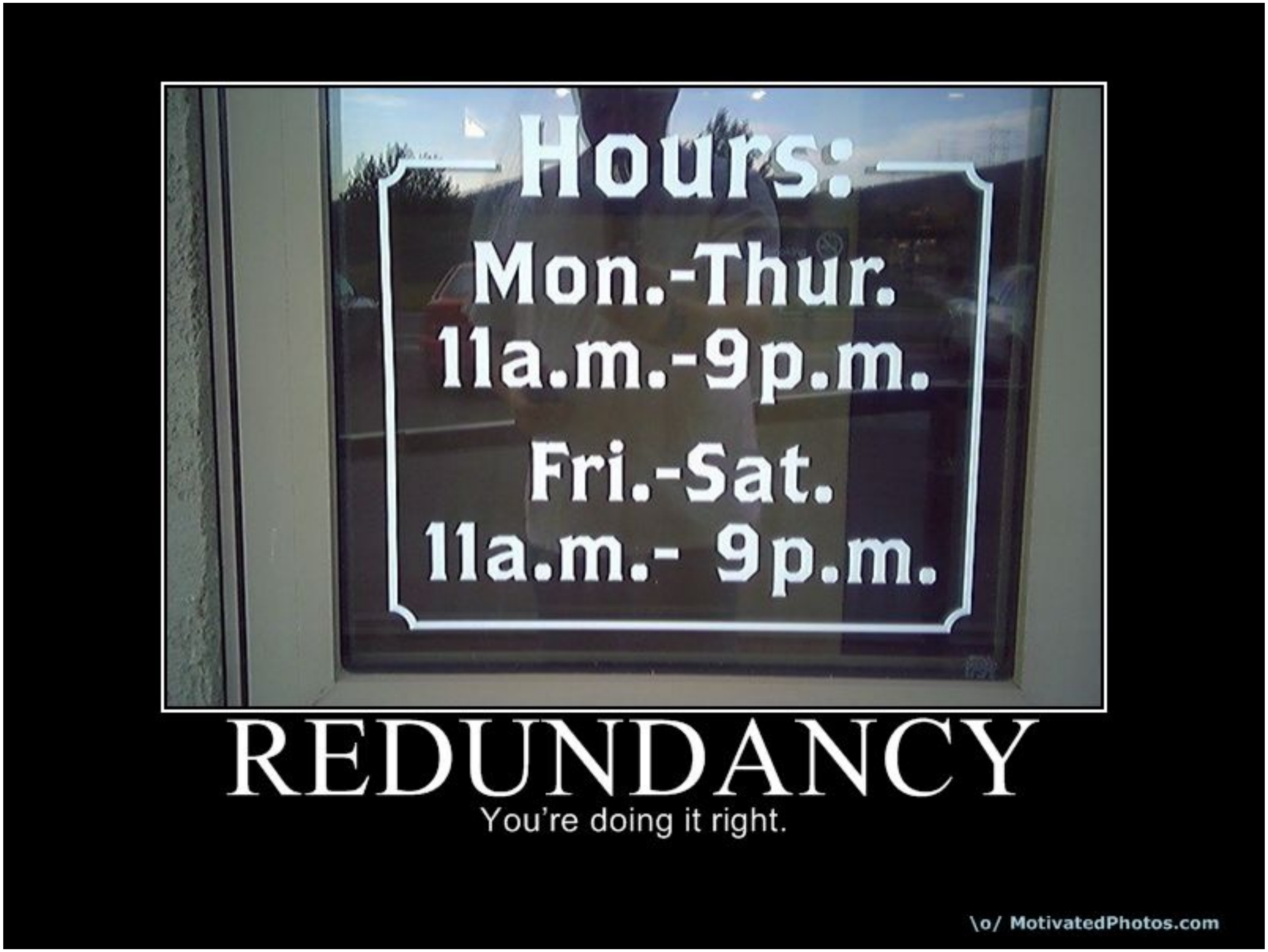
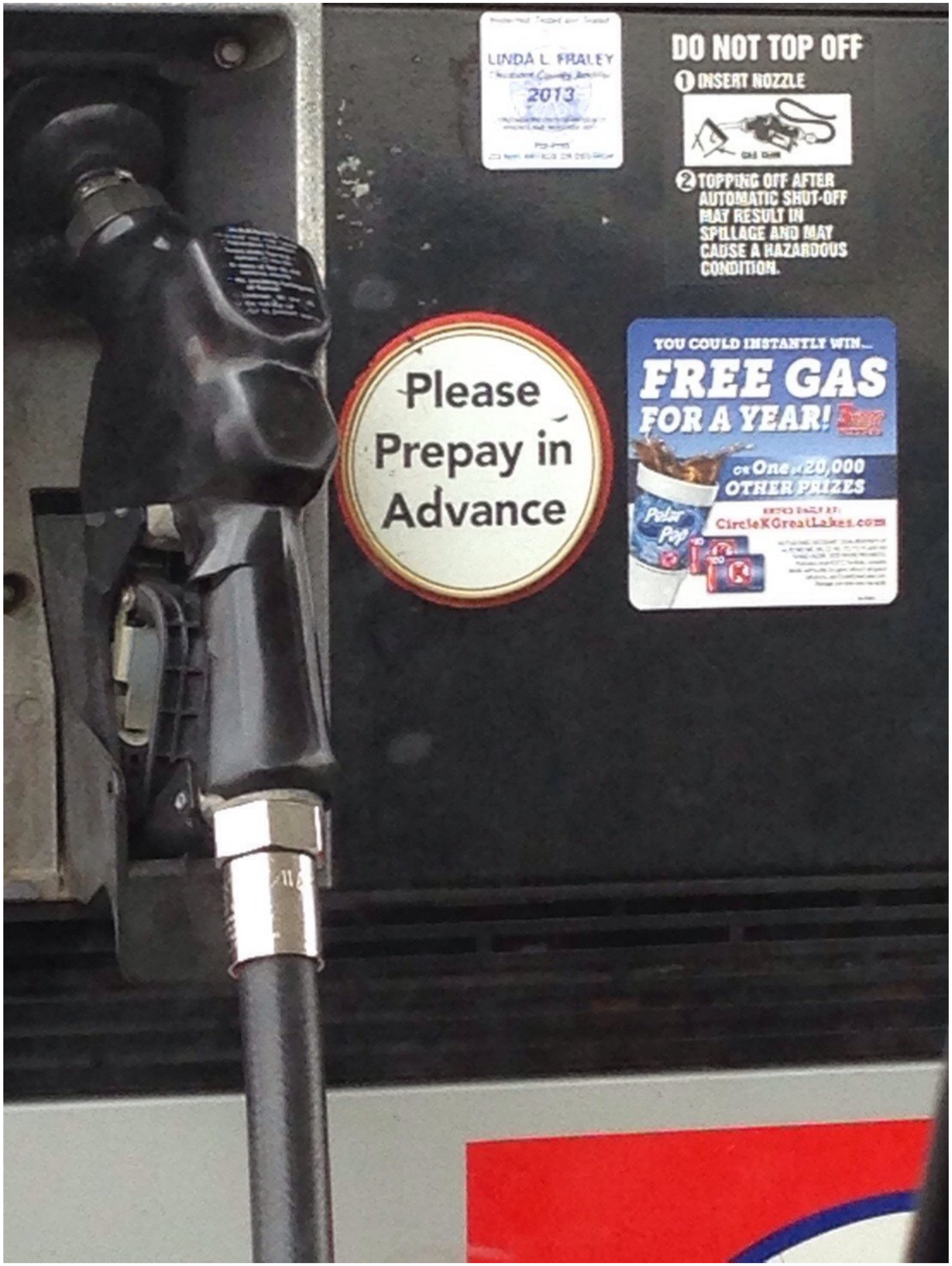
Functional Dependencies

Chapter 3

Halfway done!

- So far:
  - Relational algebra (theory)
  - SQL (practice)
  - E/R modeling (theory)
  - HTML/Flask/Python (practice)
  - Midterm (super fun!)
- Next up:
  - Database design theory (builds on E/R modeling, more theory)
  - Data structures (practice)
  - Query optimization, transactions (theory + practice!)
  - Non-relational DB models (NoSQL)


Hours:
Mon.-Thur.
11a.m.-9p.m.
Fri.-Sat.
11a.m.- 9p.m.
REDUNDANCY
You’re doing it right.
\o/ MotivatedPhotos.com


LINDA L. FRALEY
2013
DO NOT TOP OFF
1 INSERT NOZZLE
2 TOPPING OFF AFTER AUTOMATIC SHUT-OFF MAY RESULT IN SPILLAGE AND MAY CAUSE A HAZARDOUS CONDITION.
Please Prepay in Advance
YOU COULD INSTANTLY WIN...
FREE GAS FOR A YEAR!
OR One of 20,000 OTHER PRIZES
Polar Pop
CircleKGreatLakes.com

More DB design theory

- E/R diagrams can still leave you with ***redundancies*** in your schemas.

- Redundancy: Storing something twice in the DB when you only need to store it once.

Often, our first attempts at DB schemas can be improved, especially by eliminating ***redundancy****.*

Functional dependencies help us do this.

Why study FDs?

- Formal framework for making decisions about the structure of database tables.
- We can use them to algorithmically check if
  - The (primary) key we pick for a table is a "good" key.
  - There will be any redundant (duplicate) data in our table.
  - When we join our tables together, it will be done in an efficient manner.
  - (next week for all three of these)

What is a FD?

- Statement of the form:
  - *If two tuples of relation R agree on attributes A_1…A_n, then they must agree on tuples B_1…B_m.*
  - We say A_1 through A_n ***functionally determine*** B_1 through B_m.
  - i.e., if you have two rows in a table, and the two rows all have the same values for A_1…A_n, then the rows ***must*** have the same values for B_1…B_m.

What is a FD?

- Write as **X -> Y**
  - X and Y are sets of attributes from a relation.
  - Read: "X functionally determines Y"
- Intuitive definitions:
  - "If you know X, you can determine Y."
  - "For each X, there can be only one Y."
- Guidelines:
  - Often Y is a single attribute, though it doesn't have to be.

What is a FD?

- If every instance of a relation will make a FD true, then the relation ***satisfies*** the FD.
  - Determined from real-world knowledge, not DB.
- An FD is a constraint on a single relational schema (one table).
  - It must hold on every instance of the relation.
  - Therefore, you cannot deduce an FD from a relational instance.

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

What are the FDs?

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

StudentR# -> StudentName

CRN -> ProfName

CRN -> Title

StudentR# CRN -> Grade

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

Is CRN -> Grade a FD?

Where do FDs come from?

- "Key-ness" of attributes
- Domain and application constraints
- Real world constraints

Definition of Keys

- FDs allow us to formally define keys
- A set of attributes $\{A_1, A_2, …, A_n\}$ is a ***key*** for relation R if it satisfies:

Uniqueness: $\{A_1, A_2, …, A_n\}$ functionally determine all the other attributes of R

Minimality: no proper subset of $\{A_1, A_2, …, A_n\}$ functionally determines all other attributes of R.

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

What are the keys?

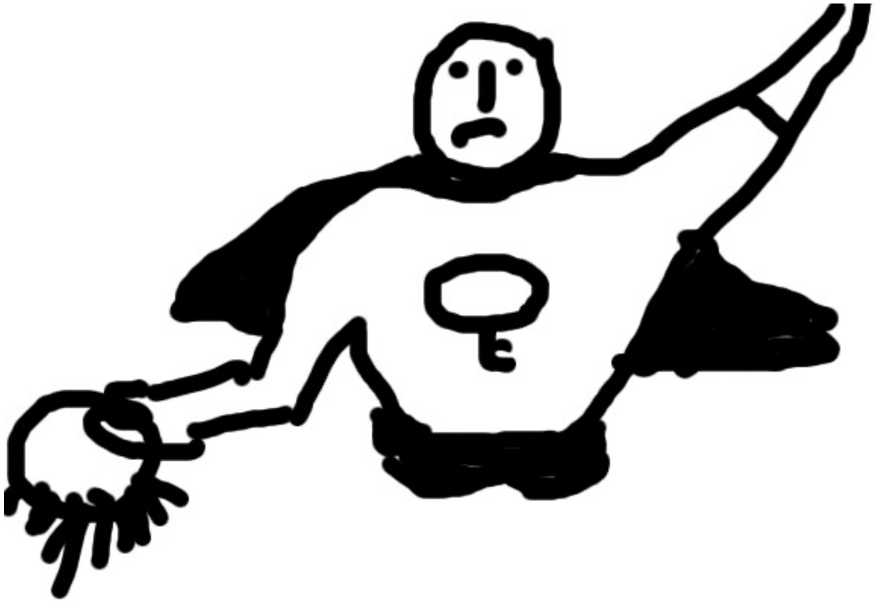
Two things you already know and one thing you don't:

- A relation can have more than one key.
- Usually one key is known as the primary key.
- FDs have nothing to do with primary keys, just keys.

Another way to think about FDs

- For a FD A_1 A_2 … A_n -> B_1 B_2 … B_m:
  - Equivalent to the set of FDs
    - A_1 A_2 … A_n -> B_1
    - A_1 A_2 … A_n -> B_2 *(etc, through)*
    - A_1 A_2 … A_n -> B_m:
  - You should be able to imagine a function f(A_1, A_2,…,A_n) that computes a unique B_1 (or B_2…).

Superkeys



Superkeys

- A ***superkey*** (superset of a key) is a set of attributes that contains a key.
- In other words, a superkey satisfies the uniqueness part of the key definition, but may not satisfy the minimality part.

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

What are the keys and superkeys?

With a partner

- Consider a relation about people in the USA, including name, SSN, street address, city, state, zip code, area code, and 7-digit phone number.
- What FDs would you expect to hold?
- What are the keys for this relation?
- Hints: Can an area code straddle two states? Can a zip code straddle two area codes?

Rules for Manipulating FDs

- Learn how to reason about FDs
- Define rules for deriving new FDs from a given set of FDs
- Example: R(A, B, C) satisfies FDs A->B, B->C.
  - What others does it satisfy?
  - A -> C
  - What is the key for R?
  - A (because A->B and A->C)

Why study FDs?

- Formal framework for making decisions about the structure of database tables.
- We can use them to algorithmically check if
  - The (primary) key we pick for a table is a "good" key.
  - There will be any redundant (duplicate) data in our table.
  - When we join our tables together, it will be done in an efficient manner.
  - (next week for all three of these)

Review

- **FD: X -> Y**: for each X, there is only one Y.
  - Knowing the value of X tells you the value of Y.
- **Superkey**: a set of attributes that functionally determines all of the other attributes of a relation.
- **Key**: a superkey that is also minimal (can't remove any attributes from it and still functionally determine all the other attributes).

Equivalence of FDs

- Why?
  - To derive new FDs from a set of FDs
- An FD F ***follows*** from a set of FDs T if every relation instance that satisfies all the FDs in T also satisfies F
  - A → C follows from T = {A→B, B→C}
- Two sets of FDs S and T are ***equivalent*** if each FD in S follows from T and each FD in T follows from S
  - S = {A→B, B→C, A→C} and T = {A→B, B→C} are equivalent

Splitting and Combining FDs

- The set of FDs
  - A1 A2 A3…An → B1
  - A1 A2 A3…An → B2
  - …

  is equivalent to the FD
  - A1 A2 A3…An → B1 B2 B3 …
- This equivalence implies two rules:
  - Splitting rule
  - Combining rule
  - These rules work because all the FDs in S and T have identical left hand sides

Splitting and Combining FDs

- Can we split and combine left hand sides of FDs?
- Consider a relation:
  Flights(airline, flightNum, source, dest)
- What are FDs?
- Does the FD “flightNum -> source” follow from “airline flightNum -> source”?
  - No!

Triviality of FDs

- A FD A1 A2…An → B1 B2…Bm is
  - **Trivial** if the B's are a subset of the A's

    $$\{B_1, B_2, \dots B_n\} \subseteq \{A_1, A_2, \dots A_n\}$$

  - **Non-trivial** if at least one B is not among the A's

    $$\{B_1, B_2, \dots B_n\} - \{A_1, A_2, \dots A_n\} \neq \emptyset$$

  - **Completely non-trivial** if none of the B's are among the A's

    $$\{B_1, B_2, \dots B_n\} \cap \{A_1, A_2, \dots A_n\} = \emptyset$$

  - Most real-world FDs are expressed completely non-trivially.

Triviality of FDs

- What good are trivial and non-trivial FDs?
  - Trivial dependencies are always true
  - They help simplify reasoning about FDs
- *Trivial dependency rule*: The FD A1 A2…An → B1 B2…Bm is equivalent to the FD A1 A2…An → C1 C2...Ck, where the C's are those B's that are not A's, i.e.

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

- Example: Suppose this FD holds: SSN -> birthday SSN
  Then this FD also holds: SSN -> birthday

- Find a trivial FD:

StudentR#	StudentName	CRN	ProfName	Title	Grade
101	Harry Potter	1	Snape	Potions	A
101	Harry Potter	2	McGonagall	Transfiguration	B
101	Harry Potter	3	Trelawney	Divination	C
102	Hermione Granger	1	Snape	Potions	B
102	Hermione Granger	2	McGonagall	Transfiguration	A
103	Ronald Weasley	1	Snape	Potions	B

Closure of a set of attributes

- Suppose you have a set of attributes {A1, …, An} and a set of FDs S.
- The closure of {A1, …, An} under S is the set of attributes B such that
  - every relation in S also satisfies A1…An -> B.
- Intuitive def'n: B is the largest set of attributes that we can deduce from knowing A1, …, An.
- Closure of {A1,…An} denoted by $\{A1,\ldots An\}^{+}$

Closure of Attributes: Algorithm

1. Use the splitting rule so that each FD in S has one attribute on the right (always possible).
2. Set X = {A1, A2 ..., An}
3. Find a FD B1 B2…Bk → C in S such that

$\{B1\ B2\ \ldots\ Bk\} \subseteq X$ but $C \notin X$

4. Add C to X
5. Repeat the last two steps until you can't find C

Why is the algorithm correct?
Read 3.2.5 in textbook

Closure of Attributes: Example

- Suppose a relation R(A, B, C, D, E, F) has FDs:
  - AB → C, BC → AD, D → E, CF → B
- Find the closures of:
  - {A, B}
  - {B, C, F}
  - {A, F}

  under the FDs above.

Note about closure

- The closure of a set of attributes will be different for differing sets of FDs.
- R(A, B, C, D); with FDs A -> B and C -> D.
  - What is $\{A, B\}^+$?
- R(A, B, C); with FDs A -> BC and C -> D
  - What is $\{A, B\}^+$?
- Takeaway: Closure of a set of attributes is meaningless without a set of FDs.

Why compute closures?

- Can test whether any FD follows from a set of other FDs.
  - Say we know a set of FDs S, and we want to check if a "new" FD A1…An -> B follows from S.
  - Simply check if B is in $\{A1, A2, \ldots, An\}^{+}$ under S.
- To prove the correctness of rules for manipulating FDs.
- Can compute keys algorithmically.

Algorithm for computing keys

- Recall a superkey is a set of attributes that functionally determines all the other attributes.
- The closure of a set of attributes A1…An under a set of FDs gives you all the other attributes in R that can be functionally determined from knowing A1…An.
- Let’s figure out the connection between superkeys and attribute closure.

Connection between closure and keys

- Suppose we have a relation R(A1, A2, …, An).
- A superkey is a set of attributes that functionally determines all the other attributes of a relation.
  - Set X is a superkey for R iff … ?
    - $X^+ = \{A1, A2, …, An\}$
- A key is a superkey that is also minimal (can't leave any attributes out of it):
  - Set X is a minimal superkey (a key) iff … ?
    - For any attribute A in X, $(X - \{A\})^+ \neq \{A1, A2, …, An\}$

(Brute-force) algorithm for computing keys

- Given:
  - A relation R (A1, A2, …, An)
  - The set of all FDs S that hold in R
- Find:
  - Compute all the keys of R

1. For every subset K of {A1, A2, …, An} compute its closure
2. If K^+ = {A1, A2, … An} and for every attribute A, $(K – \{A\})^+$ is not {A1, A2, … An}, then output K as a key

Students and Profs

- Suppose we have one single relation with attributes:
  - R#
  - Student Name
  - ProfID (ID of professor teaching a class with the student)
  - ProfName
  - AdvisorID
  - AdvisorName

Armstrong’s Axioms

- We can use closures of attributes to determine if any FD follows from a given set of FDs
- Armstrong's axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set.

Not the right W. W. Armstrong. This is Warwick Windridge Armstrong, an Australian cricketer. He did not invent these axioms. They were originated by William Ward Armstrong, who is Canadian, and does not have a picture on Wikipedia.

Armstrong’s Axioms

X, Y, and Z are sets of attributes.

- **Reflexivity**

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

  - E.g. ssn name → ssn
  - (always gives you a trivial FD)

- **Augmentation**

$$X \rightarrow Y \Rightarrow XW \rightarrow YW$$

  - E.g. ssn → name can give you
    ssn grade → name grade

Armstrong's Axioms

X, Y, and Z are sets of attributes.

- **Transitivity**

$$\left.\begin{array}{l} X \to Y \\ Y \to Z \end{array}\right\} \Rightarrow X \to Z$$

e.g. if ssn → address and address → tax-rate then

ssn → tax-rate

Note on notation

- Relation Schema: R(A1, A2, A3): parentheses surround attributes, attributes separated by commas.
- Set of attributes: {A1, A2, A3}: curly braces surround attributes, attributes separated by commas
- FD: A1 A2 → A3: no parentheses or curly braces, attributes separated by spaces, arrows separates left hand side and right hand side
- Set of FDs: {A1 A2 → A3, A2 → A1}: curly braces surround FDs, FDs separated by commas

Computing Closures of FDs

- Many times we are given a set of FDs and are interested in learning if there is a simpler set of FDs that has all the same implications that the original set does.
- To compute the closure of a set of FDs, repeatedly apply Armstrong's Axioms until you cannot find any new FDs.

Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C)
- F = {A→B, B→C}
- $\{F\}^+$ = ??

Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C)
- F = {A→B, B→C}
- $\{F\}^+$ = {A→B, B→C, A→C, AC→B, AB→C}

Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C)
- F = {AB→C, BC→A, AC→B}
- $\{F\}^{+}$ = ??

Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C)
- F = {AB→C, BC→A, AC→B}
- $\{F\}^+$ = {AB→C, BC→A, AC→B}

Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C, D)
- F = {$A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$}
- $\{F\}^+$ = ??

Examples of Computing Closures of FDs

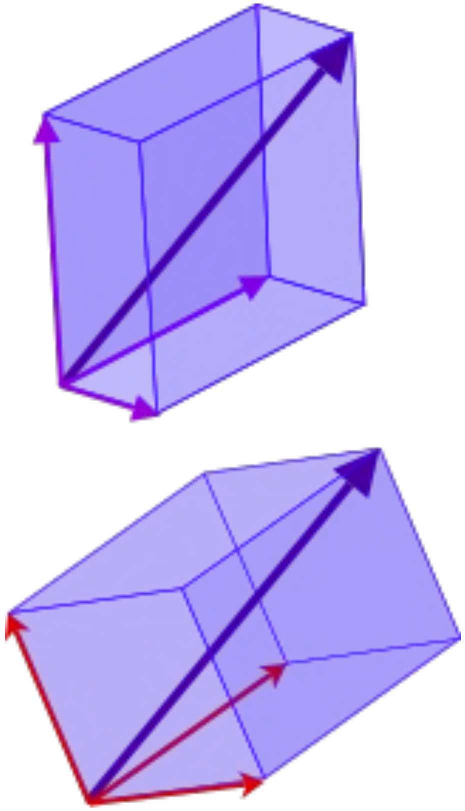
- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- Suppose we have the relation R(A, B, C, D)
- F = {A→B, B→C, C→D}
- $\{F\}^+$ = {A→B, B→C, C→D, A→C, A→D, B→D, …}

Closures of Attributes vs Closure of FDs

- Closure of attributes:
  - Takes a set of attributes A and a set of FDs S.
  - Produces a set of attributes (all the attribs that can be functionally determined from A, given S).
  - Used for computing keys, checking if an FD follows from a set of FDs.
- Closure of a set of FDs:
  - Takes a set of FDs.
  - Produces a set of FDs (all the FDs that follow from S).
  - Can be used for verifying a minimal basis, but also can verify by using closure of attributes.

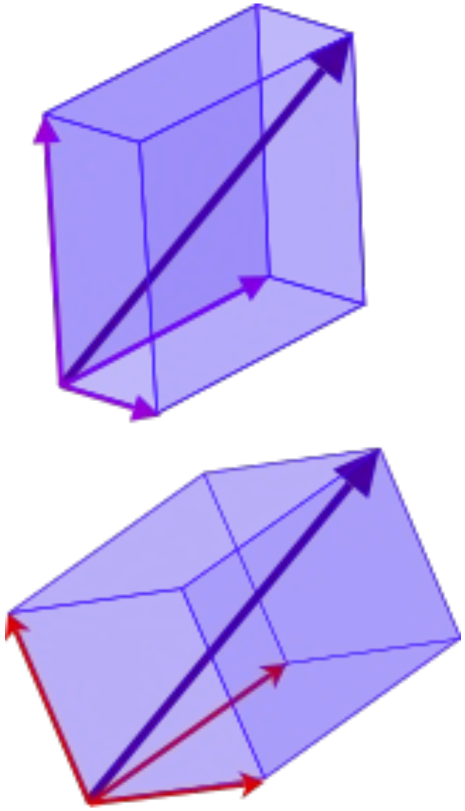
Basis Set of FDs

- In linear algebra, a ***basis*** is the smallest set of linearly independent vectors such that you can build any other vector out of the basis vectors.



Basis Set of FDs

- In databases, a ***(minimal) basis*** for a set of FDs S is the smallest set of FDs that is equivalent to S.
  - That is, all of the FDs in S follow from the basis set of FDs.



Minimal basis

- Given a set of FDs S, a minimal basis for S is another set of FDs B where:
  - All the FDs in B have singleton right sides.
  - If any FD is removed from B, the result is no longer a basis.
  - If we remove any attribute from the left side of any FD in B, the result is no longer a basis.
- Like in linear algebra, there can be multiple minimal bases for a set of FDs, though unlike in linear algebra, two minimal bases for a set of FDs may be different sizes.

Example of Minimal Basis

- R(A, B, C) is a relation such that each attribute functionally determines the other two attributes.
- What are the FDs that hold in R and what are the minimal bases?
  - (Assume only one attribute on the right-hand side, only non-trivial FDs)

Example of Minimal Basis

- R(A, B, C) is a relation such that each attribute functionally determines the other two attributes
- What are the FDs that hold in R and what are the minimal bases?
  - (Assume only one attribute on the right-hand side, only non-trivial FDs)
- FDs: A→B, A→C, B→A, B→C, C→A, C→B, AB→C, BC→A, AC→B
- Minimal Bases: {A→B, B→A, B→C, C→B},

  {A→B, B→C, C→A}, etc.