Final Review

Zaki Malik

November 20, 2008

Basic Operators Covered

- Remove parts of a single relation:
  - projection: $\pi_{A,B}(R)$ and `SELECT A, B FROM R.`
  - selection: $\sigma_C(R)$ and `SELECT * FROM R WHERE C.`
  - combining projection and selection:
    - $\pi_{A,B}(\sigma_C(R))$
    - `SELECT A, B FROM R WHERE C.`
- Set operations (R and S must have the same attributes, same attribute tyes, and same order of attributes):
  - union: $R \cup S$ and `(R) UNION (S).`
  - intersection: $R \cap S$ and `(R) INTERSECT (S).`
  - difference: $R - S$ and `(R) EXCEPT (S).`
- Combine the tuples of two relations:
  - Cartesian product: $R \times S$ and `... FROM R, S ....`
  - Theta-join: $R \underset{C}{\bowtie} S$ and `... FROM R, S WHERE C.`
  - Natural join: $R \bowtie S$; in SQL, list the conditions that the common attributes be equal in the `WHERE` clause.

Renaming

- If two relations have the same attribute, disambiguate the attributes by prefixing the attribute with the name of the relation it belongs to.
- How do we answer the query "Name pairs of students who live at the same address"? `Students(Name, Address)`
  - We need to take the cross-product of Students with itself?
  - How do we refer to the two "copies" of Students?
  - Use the rename operator.

RA: $\rho_{S\,(A1,A2,\,...\,An)}(R)$: give R the name S; R has n attributes, which are called A1,A2, . . . ,An in S

SQL: Use the **AS** keyword in the **FROM** clause: Students AS Students1 renames Students to Students1.

SQL: Use the **AS** keyword in the **SELECT** clause to rename attributes.

Q5: Find the names of sailors who have reserved a red <u>or</u> a green boat

Reserves(sid, bid, day) *Sailors(sid, sname, rating, age)*
Boats(bid, bname, color)

- Solution:

$$\pi_{sname}(\sigma_{color=\text{'red'} \text{ or } color = \text{'green'}} \; Boats \infty Reserves \infty Sailors)$$

Q6: Find the names of sailors who have reserved a red <u>and</u> a green boat

Reserves(sid, bid, day) *Sailors(sid, sname, rating, age)*
Boats(bid, bname, color)

- Solution:

$$\cancel{\pi_{sname}(\sigma_{color='red' \text{ and } color = 'green'} Boats \infty Reserves \infty Sailors)}$$

A ship cannot have TWO colors at the same time

$$\pi_{sname}(\sigma_{color='red'} Boats \infty Reserves \infty Sailors)$$
$$\cap$$
$$\pi_{sname}(\sigma_{color = 'green'} Boats \infty Reserves \infty Sailors)$$

Basic SQL Query

SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification;

- Relation-list: A list of relation names (possibly with range-variable after each name).
- Target-list: A list of attributes of relations in relation-list
- Qualification: conditions on attributes
- DISTINCT: optional keyword for duplicate removal.
  - Default = no duplicate removal!

Representing "Multiplicity"

- Show a many-one relationship by an arrow entering the "one" side. Many $\longrightarrow$ One
- Show a one-one relationship by arrows entering both entity sets. One $\longleftrightarrow$ One
- In some situations, we can also assert "exactly one," i.e., each entity of one set must be related to exactly one entity of the other set. To do so, we use a rounded arrow. $\longrightarrow$ Exactly One

1. (3 points)
Assume the schema consists of two relations R(A,B,C) and S(D,E). Consider the following expressions:

(a) $\prod_{A,C}(\sigma_{D=2}\,(R \infty_{A=E}\,(S)))$

(b) $(\prod_{A,C}(R)) \infty_{A=E}\,(\sigma_{D=2}(S))$

Are algebraic expressions (a) and (b) equivalent? Use no more than two sentences to explain your answer.

SOLUTION

No: Note that the arity (number of attributes) of the relation output of expression (a) is 2 while the arity of the relation output of expression (b) is 4

5. (22 points: 11 + 11)

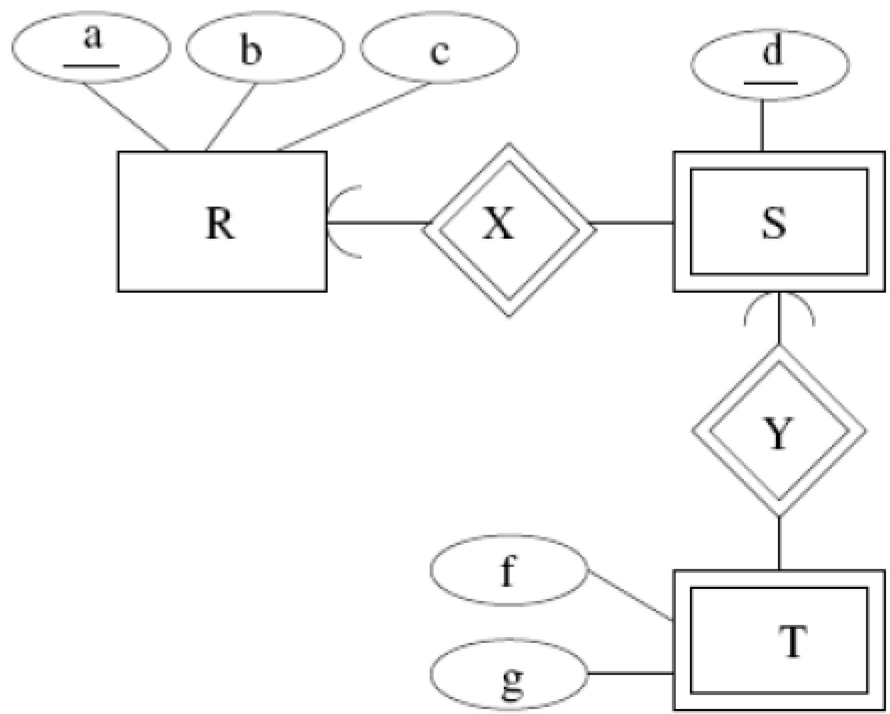
This question tests how well you understand the algorithm for converting E/R diagrams to relational schemas. An E/R diagram when converted to relations (using the mechanical construction that we know and love) gives rise to the following relations:

R(a,b,c)

S(a,d)

T(a,d,f,g)

You may assume that the same symbols refer to the same attribute and different symbols refer to different attributes (e.g., the attributes a in the relations R, S, and T are the same). Your task is to reverse-engineer the E/R diagram from these relations; in other words, what E/R diagram could have produced these relations? For full credit, give two different E/R diagrams that could have produced these relations.



5. (22 points: 11 + 11)

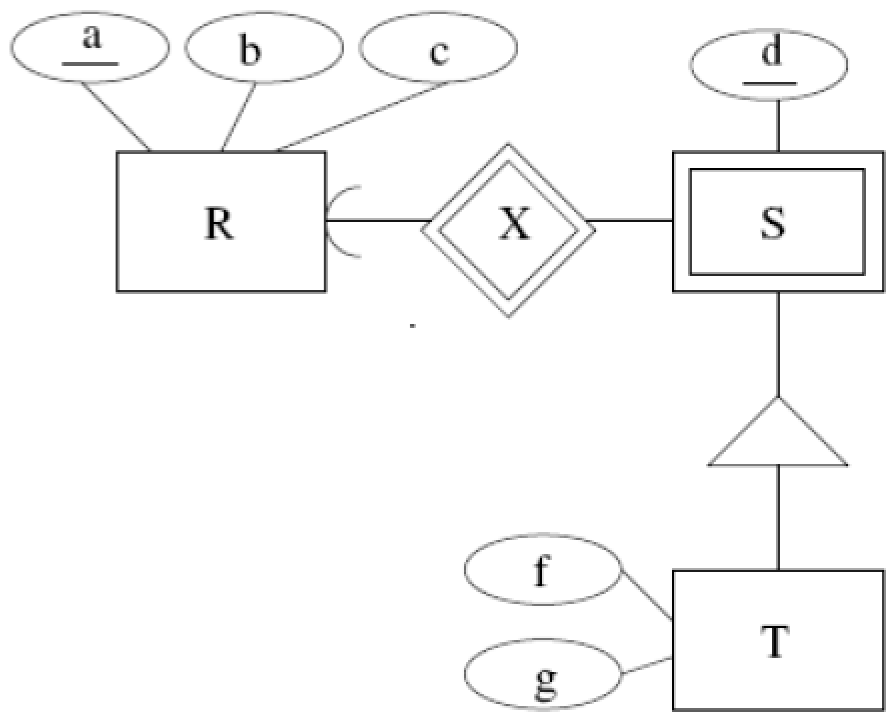
This question tests how well you understand the algorithm for converting E/R diagrams to relational schemas. An E/R diagram when converted to relations (using the mechanical construction that we know and love) gives rise to the following relations:

R(a,b,c)

S(a,d)

T(a,d,f,g)

You may assume that the same symbols refer to the same attribute and different symbols refer to different attributes (e.g., the attributes a in the relations R, S, and T are the same). Your task is to reverse-engineer the E/R diagram from these relations; in other words, what E/R diagram could have produced these relations? For full credit, give two different E/R diagrams that could have produced these relations.



Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- *trivial* if the B's are a subset of the A's,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$
- *non-trivial* if at least one B is not among the A's,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$
- *completely non-trivial* if none of the B's are among the A's, i.e.,
  $\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset$.

- *Trivial dependency rule*: The FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is equivalent to the FD $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$, where the C's are those B's that are not A's, i.e.,
  $\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}$.

- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.

Boyce-Codd Normal Form

- Condition on the FDs in a relation that guarantees that anomalies do not exist.
- A relation R is in *Boyce-Codd Normal Form* (BCNF) if and only if for every non-trivial FD $A_1A_2 \ldots A_n \rightarrow B$ for R, $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R.
- Informally, the left side of every non-trivial FD must be a superkey.
- A relation R *violates* BCNF if it has an FD such that the attributes of the left side of an FD do not form a superkey.

Closures of FDs vs. Closures of Attributes

- Both algorithms take as input a relation R and a set of FDs F.
- Closure of FDs:
  - Computes $\{F\}^+$, the set of all FDs that follow from F.
  - Output is a set of FDs.
  - Output may contain an exponential number of FDs.
- Closure of attributes:
  - In addition, takes a set $\{A_1, A_2, \ldots, A_n\}$ of attributes as input.
  - Computes $\{A_1, A_2, \ldots, A_n\}^+$, the set of all attributes B such that the $A_1 A_2 \ldots A_n \rightarrow B$ follows from F.
  - Output is a set of attributes.
  - Output may contain at most the number of attributes in R.

Checking for BCNF Violations

- List all FDs.
- Ensure that left hand side of each FD is a superkey.
- We have to first find all the keys!
- Is `Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address)` in BCNF?
- FDs are

$$\texttt{Number DepartmentName} \longrightarrow \texttt{CourseName}$$
$$\texttt{Number DepartmentName} \longrightarrow \texttt{Classroom}$$
$$\texttt{Number DepartmentName} \longrightarrow \texttt{Enrollment}$$

- What is $\{\texttt{Number}, \texttt{DepartmentName}\}^+$?

$\{$`Number, DepartmentName, Coursename, Classroom, Enrollment`$\}$.

- Therefore, the key is $\{$`Number, DepartmentName, StudentName, Address`$\}$
- The relation is not in BCNF.

Decomposition into BCNF

- Suppose R is a relation schema that violates BCNF.
- We can decompose R into a set S of new relations such that
  1. each relation in S is in BCNF and
  2. we can "recover" R from the relations in S, i.e., the relations in S "faithfully" represent the data in R.
- Let X be the set of all attributes of R.
- Suppose the FD $A_1A_2 \ldots A_m \rightarrow B$ violates BCNF.
- Decomposition algorithm:
  1. Compute $\{A_1A_2 \ldots, A_m\}^+$ and augment the FD to $A_1A_2 \ldots A_m \rightarrow \{A_1, A_2 \ldots, A_m\}^+$.
  2. Decompose R into two relations containing
     2.1 all the attributes in $\{A_1, A_2 \ldots, A_m\}^+$
     2.2 all the attributes on the left side of the FD and all the attributes of R not on the right side of the FD, i.e., $X - \{A_1, A_2 \ldots, A_m\}^+ \cup \{A_1, A_2 \ldots, A_m\}$.
  3. Find FDs in the new relations and decompose them if they are not in BCNF.

Decomposing Courses

- Schema is Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address).
- BCNF-violating FD is
  Number DepartmentName $\rightarrow$ CourseName Classroom Enrollment.
  - What is $\{$Number, DepartmentName$\}^+$?

{Number, DepartmentName, Coursename, Classroom, Enrollment}.

- Decompose Courses into
  Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment) and
  Courses2(Number, DepartmentName, StudentName, Address).

Decomposing `Courses`

- Decompose `Courses` into

  `Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment)` and

  `Courses2(Number, DepartmentName, StudentName, Address)`.

Number	DeptName	CourseName	Classroom	Enrollment
4604	CS	E-Business	211 McBryde	32
6722	CS	Advanced DB	210 McBryde	15
4322	Electrical	DB	220 McBryde	29
5722	CS	DB	311 Durham	34

Number	DeptName	StudentName	Address
4604	CS	Adam	71 Main Street
6722	CS	Adam	71 Main Street
4322	Electrical	Suri	54 Elm Street
5722	CS	Suri	54 Elm Street
5722	CS	Joe	33 Astoria Ave
6722	CS	Joe	33 Astoria Ave

- Are there any BCNF violations in the two new relations?

Third Normal Form (3NF)

- A relation R is in *Third Normal Form* (3NF) if and only if for every non-trivial FD $A_1\ A_2\ \ldots\ A_n \rightarrow B$ for R, one of the following two conditions is true:
  1. $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R or
  2. B is an attribute in some key.
- `Teach(C. D, P, S, Y)` has FDs `PSY` $\rightarrow$ `CD` and `CD` $\rightarrow$ `S`
- Keys are $\{P, S, Y\}$ and $\{C, D, P, Y\}$.
- `CD` $\rightarrow$ `S` violates BCNF.
- However, `Teach` is in 3NF because `S` is a part of a key.

Definition of MVD

- A *multivalued dependency* (MVD) *X* ->->*Y* is an assertion that if two tuples of a relation agree on all the attributes of *X*, then their components in the set of attributes *Y* may be swapped, and the result will be two tuples that are also in the relation.

Definition of MVD

- A multi-valued dependency (MVD or MD) is an assertion that two sets of attributes are independent of each other.
- The *multi-valued dependency* $A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$ holds in a relation R if in every instance of R,

  for every pair of tuples t and u in R that agree on all the A's, we can find a tuple v in R that agrees

  1. with both t and u on A's,
  2. with t on the B's, and
  3. with u on all those attributes of R that are not A's or B's.

Number	DeptName	Textbook	Professor
4604	CS	FCDB	Ullman
4604	CS	SQL Made Easy	Ullman
4604	CS	FCDB	Widom
4604	CS	SQL Made Easy	Widom

Example

	Number	DeptName	Textbook	Professor
	4604	CS	FCDB	Ullman
t	4604	CS	SQL Made Easy	Ullman
u	4604	CS	FCDB	Widom
v	4604	CS	SQL Made Easy	Widom
	2604	CS	Data Structures	Ullman
	2604	CS	Data Structures	Widom

- Number DeptName $\twoheadrightarrow$ Textbook is an MD. For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees
  1. with both t and u on Number and DeptName,
  2. with t on Textbook, and with u on Professor.
- Number DeptName $\twoheadrightarrow$ Professor is an MD. For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees
  1. with both t and u on Number and DeptName,
  2. with t on Professor, and with u on Textbook.

MVD Rules

- Every FD is an MVD
  - If *X* ->*Y*, then swapping *Y*'s between two tuples that agree on *X* doesn't change the tuples.
  - Therefore, the "new" tuples are surely in the relation, and we know *X* ->->*Y*.
- Definition of keys depend on FDs and not MDs

4NF Definition

- A relation *R* is in 4NF if whenever $X \text{->->} Y$ is a nontrivial MVD, then *X* is a superkey.
  - Nontrivial means that:
    1. *Y* is not a subset of *X*, and
    2. *X* and *Y* are not, together, all the attributes.
  - Note that the definition of "superkey" still depends on FD's only.

Decomposition and 4NF

- If X ->->Y is a 4NF violation for relation R, we can decompose R using the same technique as for BCNF.
  1. XY is one of the decomposed relations.
  2. All but $Y - X$ is the other.

Example

Drinkers(<u>name</u>, addr, <u>phones</u>, <u>beersLiked</u>)

FD: name -> addr

MVD's: name ->-> phones

name ->-> beersLiked

- Key is
  - {name, phones, beersLiked}.
- Which dependencies violate 4NF ?
  - All

Example, Continued

- Decompose using name -> addr:

1. Drinkers1(name, addr)
   - In 4NF, only dependency is name -> addr.
2. Drinkers2(name, phones, beersLiked)
   - Not in 4NF. MVD's name ->-> phones and name ->-> beersLiked apply.
   - Key ?
     - No FDs, so all three attributes form the key.

Example: Decompose Drinkers2

- Either MVD name ->-> phones or name ->-> beersLiked tells us to decompose to:
  - Drinkers3(<u>name</u>, <u>phones</u>)
  - Drinkers4(<u>name</u>, <u>beersLiked</u>)

Midterm Points Distribution

- In Order of Point Percentage
  - FDs, MDs, Normalization
  - Relational Algebra and SQL
  - ER Modeling