Triggers: Correction

Mutating Tables (Explanation)

- The problems with mutating tables are mainly with FOR EACH ROW triggers
- STATEMENT triggers can query/update/delete/etc the table that they are running on since they are run only at a point that processing is "complete" and the table is no longer mutating

Mutating Tables, Cont.

- So, the trigger of slide 13 in the trigger lesson is actually ok. It would not work if it was a FOR EACH ROW trigger
- Can you think of an action that could not be done with a trigger because of the mutating table problem? Or, can we always use a STATEMENT trigger to solve our problems?

Design Theory

Overview

- **Starting Point:** Set of function dependencies that describe real-world constraints
- **Goal:** Create tables that do not contain redundancies, so that
  - there is less wasted space
  - there is less of a chance to introduce errors in the the database

From Start to Goal (1)

- Armstrong's axioms defined, so that we can derive "implicit" functional dependencies
- Need to identify a key:
  - find a single key (algorithm from homework)
  - find all keys (algorithm taught in tirgul class)
- Both algorithms use as a subroutine an algorithm that computes the closure. In class a polynomial algorithm was given. Later today, a linear algorithm will be shown

From Start to Goal (2)

- Given a decomposition of a schema, need to be able to determine functional dependencies that hold on the sub-schemas.
- Two important characteristics of a decomposition:
  - lossless join (necessary, otherwise original relation cannot be recreated, even if tables are not modified)
  - dependency preserving: allows us to check that inserts/updates are correct without joining the relations

7

From Start to Goal (3)

- Check for a lossless join using the algorithm from class (with the a-s and b-s)
- Check for dependency preserving using an algorithm shown today
- Normal Forms:
  - 3NF: Every dependency X->A must be (1) trivial, (2) X is a superkey or (3) A is an attribute of a key
  - BCNF: Every dependency X->A must be (1) trivial or (2) X is a key

8

From Start to Goal (3)

- Algorithm for decomposition to 3NF that has a lossless join and is dependency preserving uses a minimal cover (algorithm for minimal cover shown in class)
- Polynomial algorithm for decomposition to BCNF that has a lossless join not taught
- **Question:** Can you find a trivial decomposition to BCNF of any relation?

9

Compute Closure in Linear Time

10

Closure of a Set of Attributes

- Let U be a set of attributes and F be a set of functional dependencies on U.
- Suppose that $X \subseteq U$ is a set of attributes.
- **Definition:** $X^+ = \{ A \mid F \models X \rightarrow A\}$
- We would like to compute X^+
- Note: We use the $\models$ symbol, not the $\vdash$ symbol. Is there a difference?

11

Algorithm From Class

Compute Closure(X, F)

1. $X^+ := X$
2. While there if a $V \rightarrow W$ in F such that ($V \subseteq X^+$)and ($W \not\subseteq X^+$) do
   $X^+ := X^+ \cup W$
3. Return X^+

Complexity: $|U|*|F|$

12

A More Efficient Algorithm

- We start by creating a table, with a row for each FD and a column for each attribute. The table will have 2 additional columns called size and tail.
- In the row for a dependency $X \rightarrow Y$, there will be the value true in each column corresponding to an attribute in X. The size column will contain the size of the set X. The tail column will contain Y.

Example Table

$F = \{A \rightarrow C, B \rightarrow D, AD \rightarrow E\}$

	A	B	C	D	E	Size	Tail
$A \rightarrow C$	✓					1	C
$B \rightarrow D$		✓				1	D
$AD \rightarrow E$	✓			✓		2	E


```
Compute Closure(X, F, T)  /* T is the table */
X+ := X
Q := X
While Q is not empty
  A := Q.dequeue()
  for i=1..|F|
      if T[i, A]=true then
          T[i,size] := T[i, size]  1
       if T[i,size]=0, then
           X+ := X + ∪ T[i,tail]
           Q := Q ∪ T[i,tail]
```


Computing AB⁺

Start: $X^+ = \{A,B\}$, $Q = \{A, B\}$

	A	B	C	D	E	Size	Tail
$A \rightarrow C$	✓					1	C
$B \rightarrow D$		✓				1	D
$AD \rightarrow E$	✓			✓		2	E

Computing AB⁺

Iteration of A: $X^+ = \{A,B,C\}$, $Q = \{B,C\}$

	A	B	C	D	E	Size	Tail
$A \rightarrow C$	✓					0	C
$B \rightarrow D$		✓				1	D
$AD \rightarrow E$	✓			✓		1	E

Computing AB⁺

Iteration of B: $X^+ = \{A,B,C,D\}$, $Q = \{C,D\}$

	A	B	C	D	E	Size	Tail
$A \rightarrow C$	✓					0	C
$B \rightarrow D$		✓				0	D
$AD \rightarrow E$	✓			✓		1	E

Computing AB^+

Iteration of C: X^+ = {A,B,C,D}, Q = {D}

	A	B	C	D	E	Size	Tail
$A \to C$	✓					0	C
$B \to D$		✓				0	D
$AD \to E$	✓			✓		1	E

19

Computing AB^+

Iteration of D: X^+ = {A,B,C,D,E}, Q = {E}

	A	B	C	D	E	Size	Tail
$A \to C$	✓					0	C
$B \to D$		✓				0	D
$AD \to E$	✓			✓		0	E

20

Computing AB^+

Iteration of E: X^+ = {A,B,C,D,E}, Q = {}

	A	B	C	D	E	Size	Tail
$A \to C$	✓					0	C
$B \to D$		✓				0	D
$AD \to E$	✓			✓		0	E

21

Complexity?

- To get an efficient algorithm, we assume that there are pointers from each "true" box in the table to the next "true" box in the same column. Complexity:O(|X| + |F|)

	A	B	C	D	E	Size	Tail
$A \to C$	✓					1	C
$B \to D$		✓				1	D
$AD \to E$	✓			✓		2	E

22

Decompositions that Preserve Dependencies

23

Decompositions that Preserve Dependencies

- **Problem:** Suppose that we decompose R and then insert rows into the decomposition. Is it possible that the join of these rows will contradict a FD?
- **Example:** R = CSZ (city, street, zip-code) then, CS→Z, Z→C hold in R. Suppose we decompose into SZ and CZ. This is lossless. However, we can contradict CS→Z

24

Definitions

- We define π_S (F) to be the set of dependencies X→Y in F^+ such that X and Y are in S.
- We say that a decomposition $R_1 \ldots R_n$ of R is **dependency preserving** if for all instances r of R that satisfy the FDs of R:

$$\pi_{R_1}(F) \cup \ldots \cup \pi_{R_n}(F) \text{ implies } F$$

- Note that the other direction of implication clearly holds always.
- This definition implies and exponential algorithm to check if a decomposition is dependency preserving

25

Testing Dependency Preservation

- To check if the decomposition preserves X→Y:

```
Z:=X
while changes to Z occur do
  for i:=1..n do
    Z:=Z U ((Z ∩ Ri)+ ∩ Ri)
    /* closure w.r.t. F */
Return true if Y is contained in Z
Otherwise return false
```

26

Example

- Suppose R=ABCD and we have a decomposition {AB, BC, CD}, and dependencies {A→B, B→C, C→D, D→A}.
- Does this decomposition preserve D→A?

27