SQL: Recursion
(and review for midterm)

Introduction to Databases
CompSci 316 Spring 2017

Announcements (Mon., Feb. 20)

• Midterm next Wednesday 02/22 in class
  • Up to lecture 9 included
  • We will start at 4:40 pm, come early!
  • Open book, open notes, open written material
  • No electronic devices are allowed
  • No collaboration allowed

• Homework #2 Problem 6 (Gradiance) and Problem X1 (non-Gradiance) : due on Thursday 02/23

• Project milestone 1 due next Monday 02/27

Today

• Finish recursion from Lecture 10
  • Negation and recursion
  • All lecture slides on recursion can be found from Lecture 10

• Practice a few problems for midterm

Practice Problems

• Student(sid, name)
• Enrollment(sid, cid, points)
• Find “names” of all students who never scored less than 90 points in any course

• Write query in RA
• Write query in SQL
  • EXCEPT
  • NOT EXISTS
  • ALL
  • NOT IN

RA (in class)

• Student(sid, name)
• Enrollment(sid, cid, points)
• Find names of all students who never scored less than 90 points in any course

SQL (in class) : EXCEPT

• Student(sid, name)
• Enrollment(sid, cid, points)
• Find names of all students who never scored less than 90 points in any course

SELECT name
FROM Student
EXCEPT
SELECT name
FROM Enrollment E, Student S
WHERE S.sid = E.sid
    AND points < 90
SQL (in class) : NOT EXISTS

- Student(sid, name)
- Enrollment(sid, cid, points)

Find names of all students who never scored less than 90 points in any course

SELECT name
FROM Student S
WHERE NOT EXISTS
  (SELECT *
   FROM Enrollment E
   WHERE S.sid = E.sid
   AND points < 90)

SQL (in class) : NOT IN

- Student(sid, name)
- Enrollment(sid, cid, points)

Find names of all students who never scored less than 90 points in any course

SELECT name
FROM Student S
WHERE sid NOT IN
  (SELECT sid
   FROM Enrollment
   WHERE points < 90)

SQL (in class) : ALL

- Student(sid, name)
- Enrollment(sid, cid, points)

Find names of all students who never scored less than 90 points in any course

SELECT name
FROM Student S
WHERE 90 <= ALL
  (SELECT points
   FROM Enrollment E
   WHERE S.sid = E.sid)

BCNF

- Are the following relations in BCNF given the dependencies? If not, decompose into BCNF.
  - R1(A,C,B,D,E), A → B, C → D

  Step 1: Non-trivial A → B violates the condition as A is not a key. Decompose into AB and ACDE.

  Step 2: Non-trivial C → D violates the condition as C is not a key of ACDE. Decompose into CD and ACE. No more violation.

Chase

- Let R(ABCD) be a relation with functional dependencies A → B, C → D, AD → C, BC → A
- Is the following a lossless-join decomposition of R into Boyce-Codd Normal Form (BCNF)?
  - {AB, AC, CD}?

  Consider a tuple (a,b,c,d) after joining these three relations – need to show belonged to R

  Could only come from some tuples in R of the form (a,b,c1,d1), (a,b2,c2,d2), and (a3,b3,c3,d3), which project as (a,b) in AB, (a,c) in AC, and (c,d) in CD.

  A → B tells us b2=b, and C → D tells us d2=d

  Thus, the tuple (a,b2,c,d2) is really (a,b,c,d). Since we know the former is in R, the latter is in R.

  Hence lossless – also check in BCNF – two-attribute relations are always in BCNF.

Chase

- Let R(ABCD) be a relation with functional dependencies A → B, C → D, AD → C, BC → A
- Is the following a lossless-join decomposition of R into Boyce-Codd Normal Form (BCNF)?
  - {AB, AD, BC, CD}?

  Consider a tuple (a,b,c,d) after joining these three relations – need to show belonged to R

  (a,b,c,d) must have come from tuples (a,b,c1,d1), (a,b2,c2,d2), (a3,b3,c3,d3), and (a4,b4,c4,d) in R.

  To prove (a,b,c,d) is in R, need to prove that it must be one of these tuples.

  However, the given FD’s only allow us to infer b=b2 and d=d3.

  Not lossless.

  We can come up with a counterexample by substituting constants for the variables (or just keep at b2=b, e.g., a relation R consisting of tuples (1,2,9,6), (1,2,7,4), (2,3,2,4), and (3,10,3,4). These, projected and joined, yield (1,2,3,4), which is not in R.
Remember: NULL and 3-valued logic

- A = NULL: wrong
- A IS NULL: correct

- NULL = ½, OR = max, AND = min, TRUE = 1, False = 0

- Given A = UNKNOWN, B = TRUE, C = FALSE
- (A AND B) OR C = (unknown AND True) OR False = Unknown OR False = Unknown

- Where clause selects only "True" evaluations