Week 6 Querying Databases: The Relational Way (1)

Assignment Solutions

1. The basic form of SQL queries is SELECT-FROM-WHERE. What is the corresponding relational algebra operation for SELECT, FROM, and WHERE, respectively?

- ○ Projection, Natural join, Selection
- ○ Projection, Cartesian product, Selection
- ○ Selection, Cartesian product, Projection

Answer: B

Explanation: From the lecture we know that **SELECT** is equivalent to a Projection (π), **FROM** is equivalent to a Cartesian product (×), and **WHERE** is equivalent to a selection (σ) in relational algebra.

*Note: Do not confuse the **SELECT** clause in SQL with the selection (σ) operator in relational algebra.

2. To handle comparison with Null values in relational databases, we have introduced 3-valued logic with the extra truth level UNKNOWN. If we map FALSE to integer 0, UNKNOWN to 1, and TRUE to 2, how can you encode the logic operations a AND b, a OR b, NOT a respectively, using integer operations?

- ○ $min(a, b), max(a, b), 3 - a$
- ○ $min(a, b), max(a, b), 2 - a$
- ○ $max(a, b), min(a, b), 1 - a$

Answer: B

Explanation: The best method to solve this problem is to take each given solution and test it out.

We know the following from lecture:

• False AND Unknown = False
• True AND Unknown = Unknown
• Unknown AND Unknown = Unknown
• False AND True = False
• True OR Unknown = True
• Unknown OR Unknown = Unknown
• False OR Unknown = Unknown
• True OR False = True
• NOT Unknown = Unknown
• NOT True = False
• NOT False = True

By process of elimination:

Option `max(a, b), min(a, b), 1 − a` gives 2 (True) for ”a = Unknown AND b = True”, which is incorrect.

Option `min(a, b), max(a, b), 3 − a` gives 2 (True) for ”NOT Unknown”, which is also incorrect.

Hence, we are left with option `min(a, b), max(a, b), 2 − a`, which indeed gives correct integer-based operations.

3. Consider the "Academic World" database. Say for those who are currently taking a course, the in-progress status of such students' grade is indicated using the Null value. Now you are asked to estimate the total number of students who have taken or are still taking CS 411 with a grade of at least 3.0. How will you get the lower and upper bound of this count, respectively? Your SQL query's WHERE clause should start with **WHERE number = "CS411" AND ___**. Complete the rest of the WHERE clause to get the lower and upper bound.

- [ ] Lower bound: **grade >= 3.0**. Upper bound: **grade >= 3.0 OR grade IS NULL**
- [ ] Lower bound: **grade >= 3.0**. Upper bound: **grade >= 3.0 AND grade IS NOT NULL**
- [ ] Lower bound: **grade IS NOT NULL AND grade >= 3.0**. Upper bound: **grade >= 3.0 OR grade IS NULL**

Answer: AC

Explanation: For our lower bound we are looking for at least the students who have taken CS 411. Therefore we have grade $\geq$ 3.0. In the last option, *IS NOT NULL* is redundant but preserves the truth value.

For our upper bound we are looking for all students that have taken CS 411 or are currently enrolled, which is indicated by a NULL value for grade. Hence we choose the first and last options.

4. Consider the two slightly modified "Academic World" relations: **Students(id)** and **Enrolls(id,number,term, instructor, grade)**. To retrieve a table of IDs of the students who take at least one course taught by Prof. Chang, we propose the following two SQL queries:

   SELECT id FROM Enrolls WHERE instructor = "kcchang" SELECT id FROM Students WHERE id IN (SELECT id FROM Enrolls WHERE instructor = "kcchang")

   Do you expect that these two queries *always* return the same relation? Hint: think about how SELECT handles duplicate tuples by default?

   ○ Yes

   ○ No

Answer: B

Explanation: A student can take more than one course taught by the instructor, and hence the first query can have duplicate Enrolls.id tuples (by default SELECT does not eliminate duplicates).

The second query will guarantee uniqueness in the result, because in the Students table, id is the key (since it is the only attribute).

5. It is possible to test for NULL values using comparison operators, such as =, <, or <>. Is this statement **True** or **False**?

- ◯ True
- ◯ True under certain special scenarios.
- ◯ False

Answer: B

Explanation: It is not possible to test for NULL values using comparison operators, such as =, <, or <>, as these operators cannot be applied to Null values. Therefore, SQL provides the *IS NULL* and *IS NOT NULL* operators for this purpose.