Queries Involving More than one Relation

Friday, April 5, 2019 9:56 AM

- Disambiguating Attributes

  relation_name.attribute

  - must be used if there're two same attribute name

  e.g. SELECT Movie.name

- join predicate (when don't use foreign key)

  e.g. company (cname, profit_mof), produce (name, manufacturer)

  - manufacturer is foreign key

  WHERE manufacturer = cname -- join predicate.

  - Cross join : x rows, y rows. If joined, result a x·y rows table.

  (Match each row in table A with that of table B)

  table1 CROSS JOIN table2

  - Inner join - compare each row and join if they satisfy join predicate.

  SELECT ... FROM table1 (INNER) JOIN table2 ON ...

  (optional)

NATURAL JOIN (automatically tests for equality) USING (column, ...) (avoid redundancy)

- Notice that NULL will be ignored.

- Outer Join (LEFT, RIGHT, FULL)

Same condition with inner join. But insert values to left, right won't ignore null ← SQLite only support LEFT OUTER JOIN: after joined, go to left to see if there's anything that doesn't appear on the joined table.

- Tuple variable: for each appearance of R in the FROM clause, we shall refer to as a tuple variable.
  - can be followed by AS

e.g.
```
SELECT star1.name, star2.name
FROM MovieStar star1, MovieStar star2;
```

  - If R appeared only once in the FROM clause, we can use R as its tuple name.

- Nested Loop

```
SELECT element
FROM R1, ..., Rn
WHERE condition;
```

⇐ for each element R1 {

⋮

```
for each element Rn{
    if(condition){
        select ...
    }
}
:
}
```

- therefore, if any of relation has no element, nothing will be select! (execute 0 times)

- ∪ : UNION ; ∩ : INTERSECT ; − : EXCEPT;

e.g. (SELECT name
FROM star
WHERE gender = 'F')
INTERSET
(SELECT name
FROM star
WHERE netWorth > 10000000);