Subqueries

Jaroslav Porubän, Miroslav Biňas, Milan Nosáľ (c) 2011 - 2016

Introduction

- **Inner query** - query in a query (subquery)
  - `SELECT` statement inside of another statement
- **Outer query** - query containing subquery
- Not always irreplacible
  - Often can be replaced with inner/outer join, what can be more effective (exception is subquery with EXISTS, which uses shortcircuiting making it even faster than joins)
  - Multiple subqueries can reduce readability

Usage

- Subquery in `WHERE` clause
  - Part of filtering condition - value/values needed for condition are calculated from current state of the database
  - Usable also in `INSERT/UPDATE/DELETE` statements
- In `SELECT` or `FROM` clause
  - Result of a `SELECT` is a table, therefore it can be used in other `SELECT`

Subquery example

```
SELECT name, surname
FROM FBUser
WHERE username IN
        (SELECT follower
         FROM follows
         WHERE followee='ivan');
```

- Order of processing:
  1. Subquery is evaluated (its result is needed for evaluation of outer query)
  2. Result of the subquery is ‘inlined’ to the outer query
  3. Outer query is evaluated

Subquery types

- Categorized according to result type:
  - Subqueries returning **a single value**
  - Subqueries returning **list of values (column)**
  - Subqueries returning **table**
- Result of a subquery has to fit into outer query!

Subqueries returning a single value

- Scalar query
- Condition in outer query `WHERE` can use one of the following to compare with subquery: `=`, `<>`, `<`, `<=`, `>`, `>=`, `IN`, `NOT IN`, `BETWEEN`
- Example (oldest post):

```
SELECT text
FROM Post
WHERE dateOfPost =
       (SELECT min(dateOfPost)
        FROM Post);
```

Subqueries returning list of values (a column)

- To make it usable in the `WHERE` clause of the outer query, it has to be compared with one of the following:
  - `(NOT) IN` - existence test in a set
  - `op ALL` - compared value has to be related by `op` with each value returned from subquery
  - `op ANY` - compared value has to be related by `op` with at least one of the values returned by subquery, e.g.:
    - `< ANY` - less than at least one of the results of the subquery

Subqueries returning list of values - example with **`IN`**

- Followers of ivan:

```
SELECT name, surname
FROM FBUser
WHERE username IN
        (SELECT follower
         FROM follows
         WHERE followee='ivan');
```

Subqueries returning list of values - example with **`ALL`**

- Oldest post:

```
SELECT text
FROM Post
WHERE dateOfPost <= ALL
       (SELECT dateOfPost
        FROM Post);
```

Subqueries returning list of values - example with **`ANY`**

- Post that was commented:

```
SELECT text
FROM Post
WHERE id_post = ANY
       (SELECT id_post
        FROM PostComment);
```

Correlated subqueries

- Special case of subquery
- Subquery that contains in its `WHERE` clause reference to a column of a table from outer query
    - Subquery cannot be evaluated before the outer query (it is dependent on the outer query - it is correlated to outer query)
    - Correlated subquery is evaluated once for each tuple of the outer query - it usually has a poor performance

Correlated subqueries - example

```
SELECT name, surname,
    (SELECT COUNT(*)
    FROM Post p
    WHERE p.author = fbu.username)
FROM FBUser fbu;

SELECT name, surname
FROM FBUser
WHERE (SELECT COUNT(*)
      FROM Post
      WHERE author = username)=3;
```

Operator **EXISTS** in correlated subqueries

- Tests, whether the correlated subquery (or subquery in general) returns at least one tuple
- Negation - `NOT EXISTS`
- Example:

```
SELECT *
FROM FBUser
WHERE
    EXISTS
        (SELECT *
         FROM follows
         WHERE followee=username);
```

Questions?