
MONTANA
STATE UNIVERSITY

Subqueries

Putting queries in your queries

SQL - Last Lecture: The Select Statement

- General form is

  SELECT <exprs>
  FROM <tables>
  WHERE <predicates>

- Today we are going to talk about Subqueries (aka Sub-Selects)
- Select expressions in your Select statement, for fun and for profit

Basic Subqueries

- A Subquery is a query inside another query
- Allows you to base the results of one query on the results of another
- A simple example

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID = (SELECT AlbumID
                   FROM albums
                   WHERE Title="Machine Head")
```

Basic Subqueries

- *Give me the name of all tracks whose AlbumID is the same as the AlbumID of the album with the title “Machine Head”*

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID = (SELECT AlbumID
                   FROM albums
                   WHERE Title="Machine Head")
```

Basic Subqueries

- What if we wanted the names of tracks that are on albums whose title starts with “A”?
- Let’s use a LIKE...

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID = (SELECT AlbumID
                   FROM albums
                   WHERE Title LIKE "A%")
```

Subqueries

- Seems legit…
- But it's *wrong*
- Since we are using the = operator, SQLite assumes a single value is returned

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID = (SELECT AlbumID
                   FROM albums
                   WHERE Title LIKE "A%")
```

Subqueries

- *Give me the name of all tracks on the FIRST album whose name starts with “A”*

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID = (SELECT AlbumID
                   FROM albums
                   WHERE Title LIKE "A%")
```

Subqueries

- To do what we want, we need to switch this to an IN operator
- Now we are getting the correct answer for
  Give me the name all tracks on albums that start with an "A"

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID IN (SELECT AlbumID
                    FROM albums
                    WHERE Title LIKE "A%")
```

Subqueries & Joins

- This subquery example is closely related to a concept of JOINs
- JOINs correlate tables via Foreign Keys
- We will discuss Joins in the next lecture

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID IN (SELECT AlbumID
                    FROM albums
                    WHERE Title LIKE "A%")
```

Correlated Subqueries

- So far we have looked at subqueries that are independent of the outer query
- The outer query depends on the inner query, but not vice versa
- Is it possible for the inner query to depend on the outer?

```
SELECT
    name
FROM
    tracks
WHERE
        AlbumID IN (SELECT AlbumID
                    FROM albums
                    WHERE Title LIKE "A%")
```

Correlated Subqueries

- Yes! It certainly is!
- Here is an example
- Note that the outer query albums is referenced in the inner query selecting tracks
- The sum() function is used to compute the total number of bytes of all the tracks
    - More on sum() later...

```
SELECT title
FROM albums
WHERE 10000000 > (
    SELECT sum(bytes)
    FROM tracks
    WHERE tracks.AlbumId = albums.AlbumId)
```

Correlated Subqueries

- What is this saying?

  Show me all titles of albums whose tracks size sum to less than 1M bytes

```sql
SELECT title
FROM albums
WHERE 10000000 > (
    SELECT sum(bytes)
    FROM tracks
    WHERE tracks.AlbumId = albums.AlbumId)
```

Correlated Subqueries

- Pretty cool!
- But there is a catch…
- This approach causes N + 1 queries:
  - 1 query to select all album rows
  - N queries to select all tracks per album
  - Performance issue!
- How can we get around it?
  - Denormalize the album size
  - Move condition into subquery

```
SELECT title
FROM albums
WHERE 10000000 > (
    SELECT sum(bytes)
    FROM tracks
    WHERE tracks.AlbumId = albums.AlbumId)
```

Correlated Subqueries

- Denormalize the data into the album table
  - Pros?
  - Cons?

```sql
-- Denormalized query
SELECT title
FROM albums
WHERE 10000000 > albums.bytes;
```

Correlated Subqueries

- Move condition into the subquery using a GROUP BY statement
  - More on GROUP BY in a few lectures
- This eliminates the correlation and makes it a 2 query job

```sql
-- Move condition into subquery
SELECT albums.title
FROM albums
WHERE albums.AlbumId IN
      (SELECT AlbumId FROM tracks
       GROUP BY AlbumId
    HAVING  10000000 > SUM(tracks.Bytes));
```

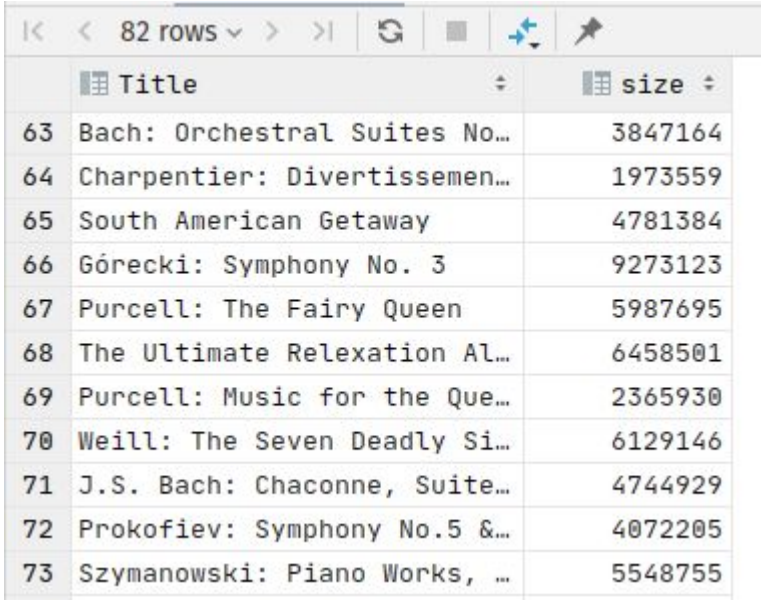
Correlated Subqueries

- Where else can we use a correlated subquery?
- In the select criterion!
- Note that we can refer to the synthetic column in the where clause
- Pretty advanced stuff...

```
SELECT title,
       (SELECT sum(bytes)
        FROM tracks
        WHERE tracks.AlbumId = albums.AlbumId)
           AS size
FROM albums
WHERE 10000000 > size
```

Correlated Subqueries

- And cool! Now we can see the size of the album in the results…
- Unfortunately this doesn't address the performance issues

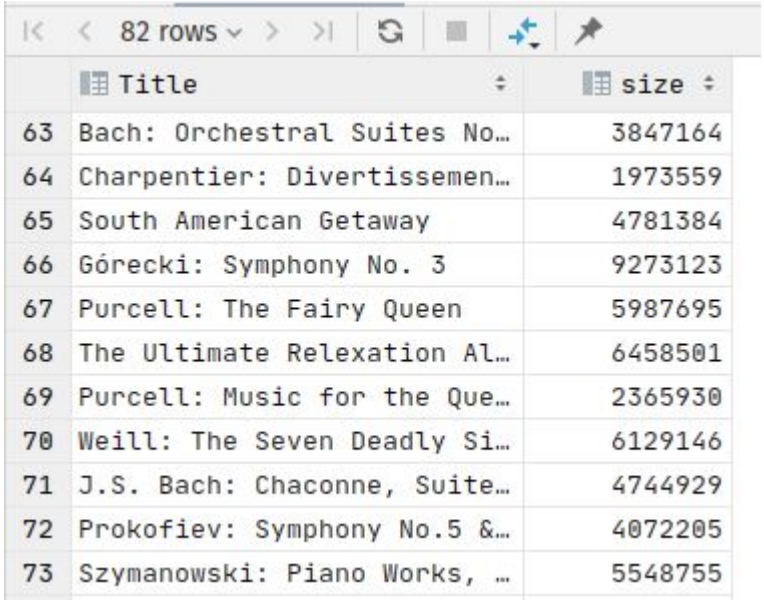


82 rows

	Title	size
63	Bach: Orchestral Suites No…	3847164
64	Charpentier: Divertissemen…	1973559
65	South American Getaway	4781384
66	Górecki: Symphony No. 3	9273123
67	Purcell: The Fairy Queen	5987695
68	The Ultimate Relexation Al…	6458501
69	Purcell: Music for the Que…	2365930
70	Weill: The Seven Deadly Si…	6129146
71	J.S. Bach: Chaconne, Suite…	4744929
72	Prokofiev: Symphony No.5 &…	4072205
73	Szymanowski: Piano Works, …	5548755

Correlated Subqueries

- Where else can you use correlated subqueries?
- Pretty much anywhere
- Using them in the WHERE and SELECT clauses are the most common



82 rows

	Title	size
63	Bach: Orchestral Suites No…	3847164
64	Charpentier: Divertissemen…	1973559
65	South American Getaway	4781384
66	Górecki: Symphony No. 3	9273123
67	Purcell: The Fairy Queen	5987695
68	The Ultimate Relexation Al…	6458501
69	Purcell: Music for the Que…	2365930
70	Weill: The Seven Deadly Si…	6129146
71	J.S. Bach: Chaconne, Suite…	4744929
72	Prokofiev: Symphony No.5 &…	4072205
73	Szymanowski: Piano Works, …	5548755

Subqueries Summary

- A SELECT within a SELECT
- Independent Subqueries
  - Usually involve an IN expression
  - Closely related to JOINs
  - Somewhat rare but very useful in real world applications
- Correlated Subqueries
  - The inner query refers to the outer queries data
  - Very powerful!
  - Performance issues (N+1) queries
  - Limited use in real world applications because of this


MONTANA
STATE UNIVERSITY