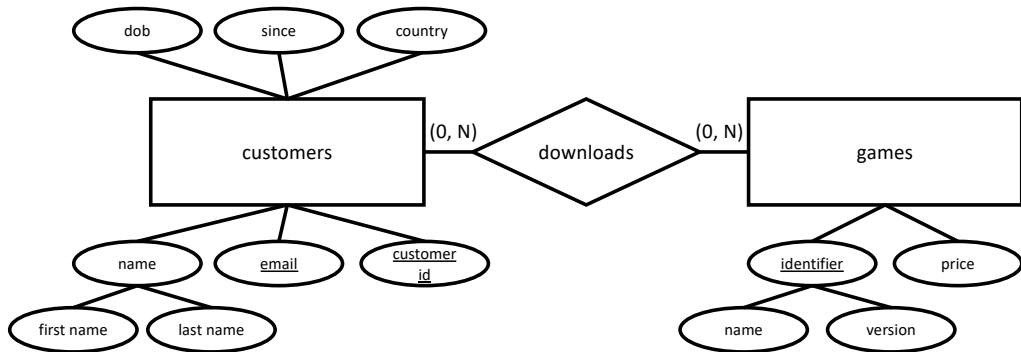


Nested Queries in SQL

Stéphane Bressan



Entity-relationship Diagram



This is the complete schema for our example

```
1 CREATE TABLE IF NOT EXISTS customers (  
2   first_name VARCHAR(64) NOT NULL,  
3   last_name  VARCHAR(64) NOT NULL,  
4   email     VARCHAR(64) UNIQUE NOT NULL,  
5   dob       DATE NOT NULL,  
6   since     DATE NOT NULL,  
7   customerid VARCHAR(16) PRIMARY KEY,  
8   country   VARCHAR(16) NOT NULL);  
9  
10 CREATE TABLE IF NOT EXISTS games(  
11   name VARCHAR(32),  
12   version CHAR(3),  
13   price NUMERIC NOT NULL,  
14   PRIMARY KEY (name, version));  
15  
16 CREATE TABLE downloads(  
17   customerid VARCHAR(16) REFERENCES customers(customerid)  
18     ON UPDATE CASCADE ON DELETE CASCADE  
19     DEFERRABLE INITIALLY DEFERRED,  
20   name VARCHAR(32),  
21   version CHAR(3),  
22   PRIMARY KEY (customerid, name, version),  
23   FOREIGN KEY (name, version) REFERENCES games(name, version)  
24     ON UPDATE CASCADE ON DELETE CASCADE  
25     DEFERRABLE INITIALLY DEFERRED);
```

We can use **subqueries** in the FROM clause.

```
1 SELECT cs.last_name , d.name
2 FROM   (SELECT *
3         FROM customers c
4         WHERE c.country = 'Singapore') AS cs , downloads d
5 WHERE cs.customerid = d.customerid;
```

last_name	name
"Green"	"Biodex"
"Green"	"Domainer"
"Green"	"Subin"
...	

It is rarely a good idea. Most of the time the same query can be written as a simple query.

```
1 SELECT c.last_name , d.name
2 FROM   customers c , downloads d
3 WHERE  c.country = 'Singapore'
4 AND    c.customerid = d.customerid;
```

We can use **subqueries** in the WHERE clause.

```
1 SELECT d.name
2 FROM downloads d
3 WHERE d.customerid IN (
4     SELECT c.customerid
5     FROM customers c
6     WHERE c.country = 'Singapore');
```

name
"Biodex"
"Domainer"
"Subin"
...

Most of the time the same query can be written as a simple query.

```
1 SELECT d.name
2 FROM customers c, downloads d
3 WHERE c.country = 'Singapore'
4 AND c.customerid = d.customerid;
```

The queries below are the same.

```
1 SELECT d.name
2 FROM downloads d
3 WHERE d.customerid IN (
4     SELECT c.customerid
5     FROM customers c
6     WHERE c.country = 'Singapore');
```

```
1 SELECT d.name
2 FROM downloads d
3 WHERE d.customerid = ANY (
4     SELECT c.customerid
5     FROM customers c
6     WHERE c.country = 'Singapore');
```

Never use a comparison to a subquery without specifying the quantifier ALL or ANY.

ANY and ALL

The query below finds the names and versions and the price of the most expensive games.

```
1 SELECT g1.name, g1.version, g1.price
2 FROM games g1
3 WHERE g1.price >= ALL (
4     SELECT g2.price
5     FROM games g2);
```

name	version	price
"Aerified"	"1.0"	12
"Aerified"	"2.1"	12
"Alpha"	"1.0"	12
...		

ALL adds expressive power similar to that of OUTER JOIN, EXCEPT and aggregate functions.

If we change ALL to ANY, we print all the games.

```
1  SELECT g1.name, g1.version, g1.price
2  FROM   games g1
3  WHERE  g1.price >= ANY (
4  SELECT  g2.price
5  FROM    games g2);
6
```

The following queries do not work.

```
1 SELECT g.name, g.version, g.price
2 FROM games g
3 HAVING g.price = MAX(g.price);
```

```
1 ERROR: column "g.name" must appear in the GROUP BY clause or be used in an aggregate function
2 LINE 1: SELECT g.name, g.version, g.price
3           ^
4 SQL state: 42803
5 Character: 8
```

```
1 SELECT g1.name, g1.version, g1.price
2 FROM games g1
3 WHERE g1.price = MAX(SELECT g2.price FROM games g2);
```

```
1 ERROR: syntax error at or near "SELECT"
2 LINE 3: WHERE g1.price = MAX(SELECT g2.price FROM game g2);
3                               ^
4 SQL state: 42601
5 Character: 73
```

But we could write the following nested query.

```
1 SELECT g1.name, g1.version, g1.price
2 FROM games g1
3 WHERE g1.price = ALL (
4     SELECT MAX(g2.price)
5     FROM games g2);
```

EXISTS

EXISTS evaluates to **true** if the subquery has **some results**.
It evaluates to **false** if the subquery has **no result**.

```
1 SELECT d.name
2 FROM downloads d
3 WHERE EXISTS (
4     SELECT c.customerid
5     FROM customers c
6     WHERE d.customerid = c.customerid
7     AND c.country = 'Singapore');
```

name
"Biodex"
"Domainer"
"Subin"
...

The subquery is **correlated** to the query.

The column d.customerid of the customer table of the outer query appears in the WHERE clause of the inner query.

All subqueries can be correlated.

```
1 SELECT g1.name, g1.version, g1.price
2 FROM games g1
3 WHERE g1.price >= ALL (
4     SELECT g2.price
5     FROM games g2
6     WHERE g1.name = g2.name);
```

name	version	price
...		
"Fintone"	"1.0"	12
"Fintone"	"1.2"	12
"Fix San"	"1.1"	5.0
...		

The above query finds the names, versions and prices of the games that are the most expensive among the games of the same name.

You can always use a column from an outer table in an inner query but not the other way around.

```
1 SELECT c.customerid, d.name
2 FROM downloads d
3 WHERE d.customerid IN (
4     SELECT c.customerid
5     FROM customers c
6     WHERE c.country = 'Singapore');
```

```
1 ERROR: missing FROM-clause entry for table "c"
2 LINE 1: SELECT c.customerid, d.name
3           ^
4 SQL state: 42P01
5 Character: 8
```

Nested queries are most powerful when combined with negation.

```
1 SELECT c.customerid
2 FROM customers c
3 WHERE c.customerid NOT IN (
4     SELECT d.customerid
5     FROM downloads d);
```

```
1 SELECT c.customerid
2 FROM customers c
3 WHERE c.customerid <> ALL (
4     SELECT d.customerid
5     FROM downloads d);
```

```
1 SELECT c.customerid
2 FROM customers c
3 WHERE NOT EXISTS (
4     SELECT d.customerid
5     FROM downloads d
6     WHERE c.customerid = d.customerid);
```

The three queries above find the 22 customers who never downloaded a game.

The following query finds the countries with the largest number of customers.

```
1 SELECT c1.country
2 FROM customers c1
3 GROUP BY c1.country
4 HAVING COUNT(*) >= ALL (
5     SELECT COUNT(*)
6     FROM customers c2
7     GROUP BY c2.country);
```

country

"Singapore"

Who are our best customers in each country (those who spend the most money among all the customers in their country)?

customerid	country
"TheresaB81"	"Thailand"
"Ashley1991"	"Vietnam"
"Amy1990"	"Malaysia"
"Helen1992"	"Singapore"
"RandyK86"	"Indonesia"

What does this query find?

```
1 SELECT c.first_name , c.last_name
2 FROM customers c
3 WHERE NOT EXISTS(
4     SELECT *
5     FROM games g
6     WHERE g.name = 'Aerified '
7     AND NOT EXISTS (
8         SELECT *
9         FROM downloads d
10        WHERE d.customerid = c.customerid
11        AND d.name = g.name
12        AND d.version = g.version));
```



Copyright 2023 Stéphane Bressan. All rights reserved.