Data Management and Warehousing

SQL Queries to Multiple Tables

Stéphane Bressan

NUS
National University
of Singapore





We want to develop a sales analysis application for our online gaming store. We would like to store several items of information about our customers: their first name, last name, date of birth, e-mail, date and country of registration on our online sales service and the customer identifier that they have chosen . We also want to manage the list of our products, the games, their version and price. The price is fixed for each version of each game. Finally, our customers buy and download games. So we must remember which version of which game each customer has downloaded. It is not important to keep the download date for this application.

An SQL query can query multiple tables. The names of the different tables are indicated in the "FROM" clause. The names are separated by a comma. The order of names does not matter.
It is recommended that you always declare and always use aliases. The aliases resolve the possible ambiguities on the field names (the same field name may appear in the schema of different tables).

```
SELECT *
FROM customers c, downloads d, games g;
```

```
SELECT *
FROM customers AS c, downloads AS d, games AS g;
```

The result of the query above is a table that contains all the fields of the three tables in the "FROM" clause, i.e. 7 + 3 + 3 = 13 fields. The records in this table correspond to all combinations of records in the three tables. Each record in the "customers" table, that is, each customer, is combined with each record in the "games" table, that is, each game, and with each record in the "downloads" table ", that is, each download, to form one of the records in the resulting table, making a total of 1001 × 430 × 4214 = 181383202 records. This is called a Cartesian product.

first_name	last_name	email	dob	since	c.customerid	country	d.customerid	d.name	d.version	g.name	g.version	price
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Aerified	1.1	3.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Aerified	1.2	1.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Aerified	2.0	5
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Aerified	2.1	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Aerified	3.0	3.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	1.0	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	1.1	3.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	1.2	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	2.0	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	2.1	2.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alpha	3.0	5
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alphazap	1.1	5
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alphazap	1.2	3.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alphazap	2.0	5
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alphazap	2.1	3.99
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Alphazap	3.0	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Andalax	1.0	12
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore	Adam1983	Biodex	1	Andalax	1.1	12
…												

A query on several tables is interesting if we add a condition on the combination of the records. For example, the following query combines the registration of a customer with the one of a game with the records corresponding to the download of that game by that customer. To do this, a condition is specified on the client identifier, the name and the version of the game in the "WHERE" clause. The customer identifier in the table "customers" must be the same as the customer identifier in the table "downloads" and the name and version of the game in the table "games" must be the same as those in the table "downloads", that is, equality of the corresponding primary and foreign keys.

```
SELECT *
FROM customers AS c, downloads AS d, games AS g
WHERE c.customerid = d.customerid
      AND d.name = g.name
      AND d.version= g.version;
```

Note that field references can now use the dot-notation to remove ambiguities. "c.customerid" and "d.customerid" are the fields with the same name "customerid" in the tables "customers" and "downloads", respectively. They are clearly distinguished by prefixing with the aliases using the dot notation.

```
SELECT *
FROM customers AS c, downloads AS d, games AS g
WHERE c.customerid = d.customerid
      AND d.name = g.name
      AND d.version= g.version;
```

You can also use the names of the tables for the dot-notation.

```
SELECT *
FROM customers, downloads, games
WHERE customers.customerid = downloads.customerid
      AND downloads.name = games.name
      AND downloads.version= games.version;
```

NUS National University of Singapore

There are now 4124 records in the result of the query since each download corresponds to one user and one game.

first_name	last_name	email	dob	since	c.customerid	country	d.customerid	d.name	d.version	g.name	g.version	price
Adam	Green	agreenf4@fc2.com	8/22/1983	5/15/2016	Adam1983	Singapore	Adam1983	Biodex	1.0	Biodex	1.0	2.99
Adam	Green	agreenf4@fc2.com	8/22/1983	5/15/2016	Adam1983	Singapore	Adam1983	Domainer	2.1	Domainer	2.1	2.99
Adam	Green	agreenf4@fc2.com	8/22/1983	5/15/2016	Adam1983	Singapore	Adam1983	Subin	1.1	Subin	1.1	2.99
Adam	Stone	astonea3@businesswire.com	4/12/1990	6/26/2016	Adam1990	Singapore	Adam1990	Fixflex	3.0	Fixflex	3.0	5
Adam	Howell	ahowelli l@storify.com	9/15/1997	1/14/2016	Adam1997	Indonesia	Adam1997	Mat Lam Tam	2.1	Mat Lam Tam	2.1	1.99
. . .												

You can now use this query and add conditions in the "WHERE" clause.

For example, you can display the name, version and price of games downloaded by the user whose email is awijaya38@xinhuanet.com.

```
SELECT g.name, g.version, g.price
FROM customers AS c, downloads AS d, games AS g
WHERE c.customerid = d.customerid
      AND d.name = g.name
      AND d.version= g.version
      AND c.email = 'awijaya38@xinhuanet.com';
```

The conditioned combination of several tables is called a join. It is possible to directly indicate the join in the "FROM" clause using the operator "INNER JOIN" (or "JOIN") and the key word "ON".

```
SELECT *
FROM (customers AS c
INNER JOIN downloads AS d
ON c.customerid = d.customerid)
INNER JOIN games AS g
ON d.name = g.name AND d.version = g.version;
```

is the same query as:

```
SELECT *
FROM customers, downloads, games
WHERE customers.customerid = downloads.customerid
      AND downloads.name = games.name
      AND downloads.version= games.version;
```

You can now use this query and add conditions in the "WHERE" clause.

For example, you can display the name, version and price of games downloaded by the user whose email is awijaya38@xinhuanet.com.

```
SELECT g.name, g.version, g.price
FROM (customers AS c
INNER JOIN downloads AS d
ON c.customerid = d.customerid)
INNER JOIN games AS g
ON d.name = g.name AND d.version = g.version
WHERE c.email = 'awijaya38@xinhuanet.com';
```

Exercise

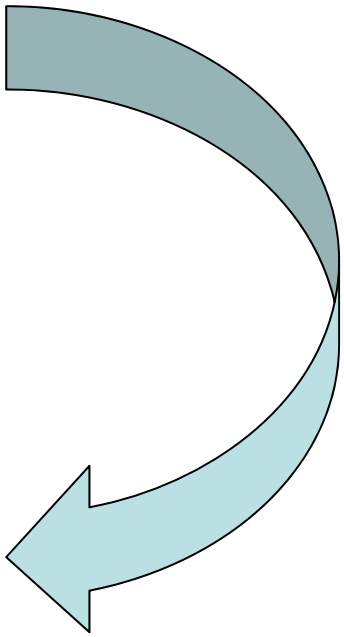
Print the first name, last name and email of the different customers who downloaded one or more versions of the game called "Domainer".

```
select a.firstname, a.lastname, email
from customers as a, downloads as b
where bnames = 'Domainer' and
a.customerid = b.customerid
```

There are customers who have not downloaded any games. The operator "LEFT OUTER JOIN" (or "LEFT JOIN") and the keyword "ON" allow to keep these customers in the result of the request by padding the corresponding records with null values "NULL".

```
SELECT *
FROM customers LEFT JOIN downloads
ON customers.customerid = downloads.customerid;
```

first_name	last_name	email	dob	since	customers.customerid	country
...						
Jacqueline	Graham	jgrahamrq@addthis.com	11/29/1995	1/4/2016	Jacqueline1995	Singapore
Samuel	Lee	sleerr@amazonaws.com	4/30/1999	8/10/2016	Samuel1999	Malaysia
Carole	Yoga	cyoga@glarge.org	8/1/1989	9/15/2016	Carole89	France



downloads.customerid	name	version
...		
Jacqueline1995	Flowdesk	1.0
Jacqueline1995	Prodder	2.0

Not all records in the `customers` table find a match in the `downloads` table

Null values

first_name	last_name	email	dob	since	cust .custo id	country	downloads.customerid	name	version
...									
Jacqueline	Graham	jgrahamrq@addthis.com	11/29/1995	1/4/2016	Jacqueline 1995	gapore	Jacqueline 1995	Flowdesk	1.0
Jacqueline	Graham	jgrahamrq@addthis.com	11/29/1995	1/4/2016	Jacqueline 1995	Sing ore	Jacqueline 1995	Prodder	2.0
Samuel	Lee	sleerr@amazonaws.com	4/30/1999	8/10/2016	Samuel1999	Malaysia			
Carole	Yoga	cyoga@glarge.org	8/1/1989	9/15/2016	Carole89	France			

The operator "LEFT OUTER JOIN" pads with null values the records of the left table that do not correspond to any field in the table on the right.

The operator "RIGHT OUTER JOIN" pads with null values the records of the right table that do not correspond to any field in the table on the left.

The operator "FULL OUTER JOIN" pads with null values both the records in the right table do not correspond to any field in the left table and the records in the left table do not correspond to any field in the table right.

Microsoft Access 2010 supports the keywords "LEFT JOIN" and "RIGHT JOIN". Microsoft Access 2010 does not have a "FULL OUTER JOIN". Microsoft Access 2010 has problems parsing complex queries. You will need to pay attention to the syntax or compose the query interactively.

SQLite does not yet support the "RIGHT OUTER JOIN" and "FULL OUTER JOIN".

It is possible to test whether a value is null (usually in the "WHERE" clause) with the operator "IS NULL".

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c LEFT JOIN downloads AS d
ON c.customerid = d.customerid
WHERE d.name IS NULL AND d.version IS NULL;
```

The above query displays the first name, last name, and e-mail of customers who have not downloaded any games.

first_name	last_name	email
Ralph	Thomas	rthomasf@imgur.com
Jennifer	Lee	jleen@mlb.com
Robert	Welch	rwelch1a@wufoo.com
Jane	Gomez	jgomez1o@epa.gov
Kathleen	Kanh	kkanh3i@phpbb.com
Beverly	Armstrong	barmstrong4k@ovh.net
Rachel	Cole	rcole6m@baidu.com
Steven	Welch	swelch74@businessweek.com
Tina	Bennett	tbennett7x@altervista.org
Johnny	Stevens	jstevensb0@un.org
Albert	Perkins	aperkinsb8@apple.com
Johnny	Gilbert	jgilberte8@nymag.com
Amanda	Reyes	areyese9@cnbc.com
Adam	Romero	aromerofh@rambler.ru
Aaron	Griffin	agriffinfo@zdnet.com
Michael	Richardson	mrichardsongy@nbcnews.com
Kanh	Simmons	msimmonsh0@tuttocitta.it
Ashley	Edwards	aedwardslr@myspace.com
Sharon	Green	sgreenmx@dyndns.org
Alan	Hansen	ahansenp3@webnode.com
Antonio	Freeman	afreemanqn@wikia.com
Samuel	Lee	sleerr@amazonaws.com
Carole	Yoga	cyoga@glarge.org

Exercise

Print the name and version of the games that have never been downloaded.

The results of two queries can be combined with the key word "UNION". The query below displays the last name, first name and e-mail of registered customers from Singapore and registered customers from Vietnam, i.e. customers registered from Singapore or from Vietnam, in this example.

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
UNION
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Vietnam';
```

Both queries must return results that have compatible schemas (same field names and domains in the same order).

Union eliminates duplicate records. The two queries below have the same result but this may not be the case in general.

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
UNION
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Vietnam';
```

```
SELECT DISTINCT c.first_name, c.last_name,
c.email
FROM customers as c
WHERE c.country = 'Singapore' OR c.country =
'Vietnam';
```

You can intersect the results of two queries with the keyword "INTERSECT". This is not possible with Microsoft Access 2010. The query below displays the name, first name and e-mail of customers registered from Singapore who are also the last name, first name and email of customers whose name begins with the capital letter "D", that is, customers registered from Singapore whose name begins with the capital letter "D".

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
INTERSECT
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.last_name LIKE 'D%';
```

Both queries must return results that have compatible schemas (same field names and domains in the same order).

Intersection eliminates duplicate records. The two queries below have the same result but this may not be the case in general.

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
INTERSECT
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.last_name LIKE 'D%';

SELECT DISTINCT c.first_name, c.last_name,
c.email
FROM customers as c
WHERE c.country = 'Singapore' AND c.last_name
LIKE 'D%';
```

One can make the difference (non-symmetric) of two queries with the keyword "EXCEPT" (or "MINUS" in some systems like Oracle). The difference eliminates duplicate records. This is not possible with Microsoft Access 2010. The query below displays the name, first name and e-mail of clients registered from Singapore that are not those whose name does not begin with the capital letter "D".

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
EXCEPT
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.last_name LIKE 'D%';
```

Both queries must return results that have compatible schemas (same field names and domains in the same order).

Difference eliminates duplicate records. The two queries below have the same result, but this will not be the case in general.

```
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.country = 'Singapore'
EXCEPT
SELECT c.first_name, c.last_name, c.email
FROM customers as c
WHERE c.last_name LIKE 'D%';
```

```
SELECT DISTINCT c.first_name, c.last_name,
c.email
FROM customers as c
WHERE c.country = 'Singapore' AND NOT
(c.last_name LIKE 'D%');
```

NUS National University of Singapore

Exercise

What happens to the three queries (with "UNION", "INTERSECT" and "EXCEPT" above if the email is not displayed?

NUS National University of Singapore

Exercise

Print the name and version of the games that have never been downloaded.

It is possible to use queries in queries. This is called a nested query or a subquery. For example, a subquery can be nested in the "FROM" clause of a query. For example, the query below displays the names of clients whose first name is Jonathan from those registered from Singapore, i.e. those registered from Singapore whose first name is Jonathan.

```
SELECT cs.last_name
FROM (SELECT *
      FROM customers as c
      WHERE country = 'Singapore') AS cs
WHERE cs.first_name = 'Jonathan';
```

A query can be nested in the "WHERE" clause using the "IN" operator. For example, the query below displays the email and country of registration of customers registered in a country where a customer named Tammy has registered.

```
SELECT c1.email, c1.country
FROM customers c1
WHERE c1.country IN (
     SELECT c2.country
     FROM customers c2
     WHERE c2.first_name = 'Tammy');
```

Exercise: in which countries did customers named Tammy have registered?

You can nest a query in the "WHERE" clause by using the "NOT IN" operator. For example, the query below displays the email and country of registration of customers registered in a country where no customer named Tammy has registered.

```
SELECT c1.email, c1.country
FROM customers c1
WHERE c1.country NOT IN (
     SELECT c2.country
     FROM customers c2
     WHERE c2.first_name = 'Tammy');
```

NUS National University of Singapore

Exercise

Print the first and last names of customers who have never downloaded a game.

You can nest a query in the "WHERE" clause by using any appropriate operator with an "ALL" or "ANY" quantifier. For example, the query below displays the email and country of registration of customers registered in a country or a customer named Tammy has registered. "= ALL" is the same operator as "IN".

```
SELECT c1.email, c1.country
FROM customers c1
WHERE c1.country = ANY (
      SELECT c2.country
      FROM customers c2
      WHERE c2.first_name = 'Tammy');
```

Although some systems allow you to omit the quantifier you should never do so.

For example, the query below displays the email and country of registration of customers registered in a country where no customer named Tammy has registered. "<> ALL" is the same operator as "NOT IN".

```
SELECT c1.email, c1.country
FROM customers c1
WHERE c1.country <> ALL (
      SELECT c2.country
      FROM customers c2
      WHERE c2.first_name = 'Tammy');
```

For other purposes, we can also use "> ALL", ">= ANY" etc..

NUS National University of Singapore

Exercise

Print the first and last names of customers who have never downloaded a game.

It is also possible to correlate the outer and the nested queries: one can use fields of a outer query in the subqueries (never the other way round). This is often the case with the "EXISTS" and "NOT EXISTS" subquery builders, and it can happen with the "IN" and "NOT IN" operators.

```
SELECT g.name, g.version, g.price
FROM games g
WHERE NOT EXISTS (
      SELECT *
      FROM downloads
      WHERE g.name = d.name
      AND g.version = d.version)
```

Exercise… What does the above query print?

NUS National University of Singapore

Credits

Images and clips used in this presentation are licensed from Microsoft Office Online Clipart and Media

For questions about the content of this course and about copyrights, please contact Stéphane Bressan

steph@nus.edu.sg