Introduction to Database Systems

SQL Simple and Algebraïc Queries

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.

1. Go to `www.sqlite.org`
2. Go to `www.sqlite.org/download.html`
3. Download the command-line shell for accessing and modifying SQLite databases ("`A bundle of` …")
4. Extract the executable
5. You will find a short documentation at www.sqlite.org/sqlite.html


```
› .open cs2102.db
› .mode column
› .headers on
› .help
› PRAGMA foreign_keys = ON;
› .read customers.sql
› .read games.sql
› .read downloads.sql
› …
› .quit
```

open or create a database
display results in columns
display names of fields
enables foreign constraints
read the SQL file

save and quit

You may skip the "`.open cs2102.db`" for now. That step creates a persistent database but also may slow down some update operations.

Here is the complete code of the example with primary and foreign keys.

```
CREATE TABLE customers (
        first_name VARCHAR(64) NOT NULL,
        last_name VARCHAR(64) NOT NULL,
        email VARCHAR(64) UNIQUE NOT NULL,
        dob DATE NOT NULL,
        since DATE NOT NULL,
        customerid VARCHAR(16) PRIMARY KEY,
        country VARCHAR(16) NOT NULL);

CREATE TABLE games (
        name VARCHAR(32),
        version CHAR(3),
        price NUMERIC NOT NULL,
        PRIMARY KEY (name, version));

CREATE TABLE downloads(
        customerid VARCHAR(16) REFERENCES customers(customerid),
        name VARCHAR(32),
        version CHAR(3),
        FOREIGN KEY (name, version) REFERENCES games(name, version),
        PRIMARY KEY(customerid, name, version));
```

NUS National University of Singapore

Data: Three Tables

first_name	last_name	email	dob	since	userid	country
Deborah	Ruiz	druiz0@drupal.org	8/1/1984	10/17/2016	Deborah84	Singapore
Tammy	Lee	tlee1@barnesandnoble.com	9/14/1998	8/21/2016	Tammy1998	Singapore
Rebecca	Garza	rgarza2@cornell.edu	6/11/1984	9/26/2016	RebeccaG84	Malaysia
Walter	Leong	wleong3@shop-pro.jp	6/26/1983	6/12/2016	Walter83	Singapore
Kathryn	Edwards	kedwards4@twitter.com	11/27/1993	5/17/2016	Kathryn1993	Singapore
…						

userid	name	version
Aaron1986	Wrapsafe	1.2
Adam1983	Biodex	1
Adam1983	Domainer	2.1
Adam1983	Rank	2
Adam1983	Subin	1.1
…		

name	version	price
Aerified	1	12
Aerified	1.1	3.99
Aerified	3	3.99
Alpha	1	12
Alpha	1.1	3.99
…		

A simple SQL query includes a "SELECT" clause, which indicates the fields to be printed, a "FROM" clause, which indicates the table (s) to be queried and possibly a "WHERE" clause, which indicates a possible condition on the records to be printed. We have seen the following query that displays the first and last names of registered customers from Singapore.

```
SELECT first_name, last_name
FROM customers
WHERE country = 'Singapore';
```

Structured Query Language (SQL) is a simplified, specialized query language for updating and querying relational databases.

It is an international industrial standard (unfortunately implemented with slight variations according to the vendors).

The following two queries print all the customer information, that is, all fields in the "customers" table.

```
SELECT first_name, last_name, email, dob, since,
     customerid, country
FROM customers;
```

The asterisk is a shorthand for all the field names.

```
SELECT *
FROM customers;
```

first_name	last_name	email	dob	since	customerid	country
Aaron	Griffin	agriffinfo@zdnet.com	5/31/1986	3/3/2016	Aaron1986	Singapore
Adam	Green	agreenf4@fc2.com	8/22/1983	5/15/2016	Adam1983	Singapore
Adam	Stone	astonea3@businesswire.com	4/12/1990	6/26/2016	Adam1990	Singapore
Adam	Howell	ahowellil@storify.com	9/15/1997	1/14/2016	Adam1997	Indonesia
Adam	Romero	aromerofh@rambler.ru	11/4/1998	12/19/2016	Adam1998	Singapore
Adam	Wijaya	awijaya38@xinhuanet.com	2/21/2000	1/8/2016	Adam2000	Singapore
Alan	Hansen	ahansenp3@webnode.com	11/22/1998	8/1/2016	Al8	Singapore
…						

It is possible to choose, reorder and replicate fields in the "SELECT" clause.

```
SELECT email, email, last_name, first_name
FROM customers;
```

The "SELECT" clause determines the schema of the table containing the result of the query.

Expr1000	email	last_name	first_name
agriffinfo@zdnet.com	agriffinfo@zdnet.com	Griffin	Aaron
agreenf4@fc2.com	agreenf4@fc2.com	Green	Adam
astonea3@businesswire.com	astonea3@businesswire.com	Stone	Adam
ahowellil@storify.com	ahowellil@storify.com	Howell	Adam
aromerofh@rambler.ru	aromerofh@rambler.ru	Romero	Adam
awijaya38@xinhuanet.com	awijaya38@xinhuanet.com	Wijaya	Adam
ahansenp3@webnode.com	ahansenp3@webnode.com	Hansen	Alan
…			

In SQL, the result of a query is a table. You can give a name to the query by using the SQL command "CREATE VIEW". The result of the query is now just like any other table for the purpose of querying.

```
CREATE VIEW customers_basic AS
SELECT last_name, first_name, email
FROM customers;
```

This table can then be reused in other queries. If the "customers" table is updated, the "customers_basic" view also changes.

```
SELECT email
FROM  customers_basic
WHERE last_name='Yoga';
```

It is possible to rename the fields in the "SELECT" clause using the keyword "AS".

```
SELECT last_name AS family_name, first_name
FROM customers;
```

It is possible to make calculations on the fields in the "SELECT" clause.

```
SELECT name || ' ' || version AS game,
      price * 1.18 AS pricetax
FROM games;
```

Caution, the syntax of operations and functions can be specific to the DBMS used. For example, Oracle and MySQL use the "CONCAT ()" function while PostgreSQL and SQLite use "||" instead of "&" used by Microsoft Access 2010 and "+" by SQL Server.

game	pricetax
Aerified 1	14.16
Aerified 1.1	4.7082
Aerified 1.2	2.3482
Aerified 2	5.9
Aerified 2.1	14.16
Aerified 3	4.7082
Alpha 1	14.16
...	

SQL can use the calculated fields, i.e. the fields for which the values are calculated from existing fields, in the "SELECT" and "WHERE" clauses (and also "HAVING" that we will see later) . SQL includes operations and arithmetic functions (for example: addition +, subtraction -, multiplication * and division /, with parentheses if necessary), operations and functions for other domains (types) such as dates and strings characters.

```
SELECT name || ' ' || version
FROM games
WHERE price * 1.18 > 10;
```

Caution, the syntax of operations and functions can be specific to the DBMS used.

Expr1000
Aerified 1
Aerified 2.1
Alpha 1
Alpha 1.2
Alpha 2
…

The keyword "DISTINCT" in the "SELECT" clause (it appears only once after the keyword "SELECT", it eliminates repeated records) eliminates duplicates in the result.

```
SELECT first_name, last_name
FROM customers;
```

```
SELECT DISTINCT first_name, last_name
FROM customers;
```

The result of the first query can contain several occurrences of the same pair of last name and first name, not that of the second.

The query without "`DISTINCT`" displays 1001 records

The query with "`DISTINCT`" displays 983 records

first_name	last_name
Aaron	Griffin
Adam	Green
Adam	Stone
Adam	Howell
Adam	Romero
Adam	Wijaya
…	

The "WHERE" clause is optional. It is used to filter records that satisfy a single or compound condition.

A single condition compares the value of a field with a constant or the values of two fields with each other with the comparison operators =, <, <=, >=, <>, LIKE, BETWEEN, AND and IN.

```
SELECT name, version
FROM games
WHERE price >= 10;
```

```
SELECT name, price
FROM games
WHERE price BETWEEN 4 AND 20;
```

```
SELECT name, version
FROM games
WHERE version IN ('1.0','1.1');
```

name	version
Aerified	1
Aerified	2.1
Alpha	1
Alpha	1.2
. . .	

name	price
Aerified	12
Aerified	5
Aerified	12
Alpha	12
…	

name	version
Aerified	1.0
Aerified	1.1
Alpha	1.0
Alpha	1.1
Alphazap	1.1
…	

"LIKE" compares strings according to a pattern (% replaces any character string).

```
SELECT first_name, last_name, customerid
FROM customers
WHERE customerid LIKE 'M%88';
```

Some systems use other symbols. Some systems support full regular expression.

first_name	last_name
Margaret	Watkins
Maria	Myers
Marie	Armstrong
Matthew	Vasquez
Mildred	Robinson

The "WHERE" clause is optional. It is used to filter records that satisfy a single or compound condition.

A compound condition combines simple conditions into a Boolean expression with the Boolean operators AND, OR, NOT and parentheses, if necessary.

```
SELECT name
FROM games
WHERE price BETWEEN 4 AND 20
AND version IN ('1.0','1.1');
```

```
SELECT name
FROM games
WHERE price BETWEEN 4 AND 20
AND NOT (version <> '1.0' AND version <> '1.1');
```

name
Aerified
Alpha
Alphazap
Andalax
Andalax
Asoka
…

NULL Values Logic

"`SELECT FROM WHERE condition`" returns results when the condition is true.

P	Q	P AND Q	P OR Q	NOT P
True	True	True	True	False
False	True	False	True	True
Unknown	True	Unknown	True	Unknown
True	False	False	True	False
False	False	False	False	True
Unknown	False	False	Unknown	Unknown
True	Unknown	Unknown	True	False
False	Unknown	False	Unknown	True
Unknown	Unknown	Unknown	Unknown	Unknown

Exercise

```
version IN ('1.0', '1.1')
```

The above condition is the same as the one below.

```
(version = '1.0' OR version = '1.1')
```

Check that it is the above condition is the same as the one below.

```
NOT (version <> '1.0' AND version <> '1.1')
```

Exercise

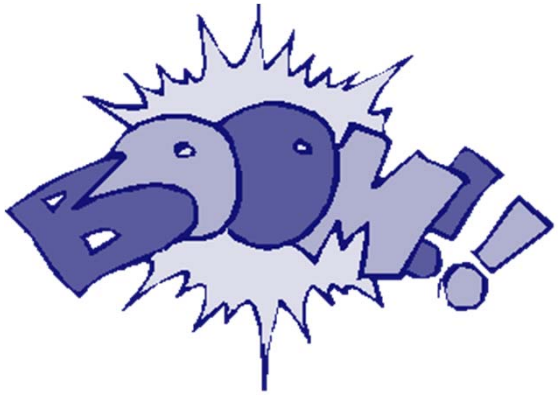
Write a query that displays the names of the games starting with the capital letter "B" or the capital letter "C" and whose version 3.0 costs just over $ 4 after taxes (18%).

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 (or cross product).



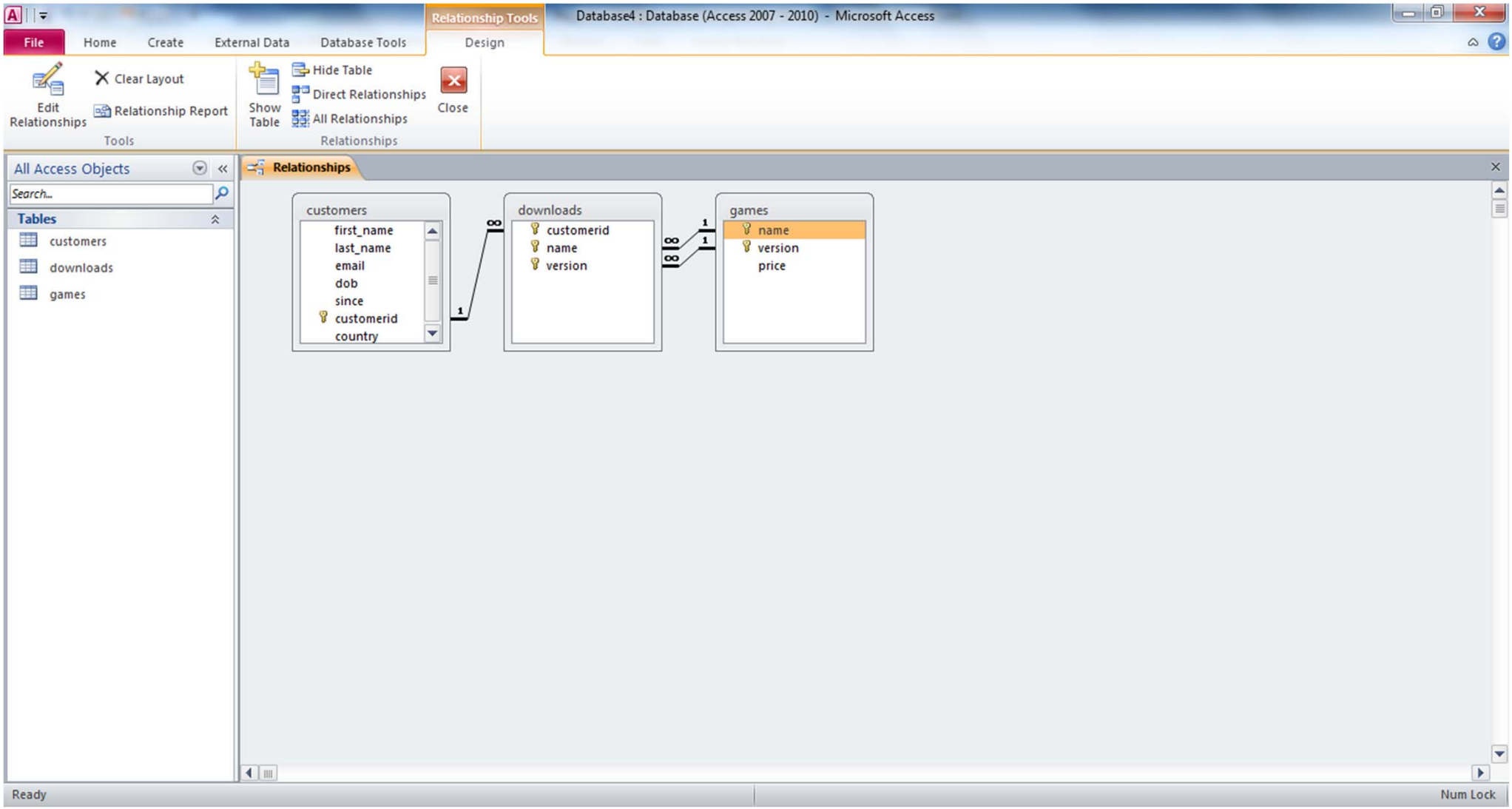
It can also be explicitly written as a `CROSS JOIN`

```
SELECT *
FROM customers c CROSS JOIN downloads d CROSS
JOIN games g;
```

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 that 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;
```


Relationship Tools
Database4 : Database (Access 2007 - 2010) - Microsoft Access
File
Home
Create
External Data
Database Tools
Design
Edit Relationships
Clear Layout
Relationship Report
Tools
Show Table
Hide Table
Direct Relationships
All Relationships
Relationships
Close
All Access Objects
Search...
Tables
customers
downloads
games
Relationships
customers
first_name
last_name
email
dob
since
customerid
country
downloads
customerid
name
version
games
name
version
price
1
∞
Ready
Num Lock

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 the prefixing.

```
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;
```

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;
```

```
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;
```

Find the first and last name of the customers who downloaded an app.

```
SELECT c.fisrt_name, c.last_name
FROM customers c INNER JOIN downloads d
ON u. customerid = d. customerid;
```

`INNER JOIN (JOIN)` combines rows of the two table on the condition given after `ON`.

`INNER JOIN` does not eliminate duplicates.

`NATURAL JOIN` joins the tables on the condition that the fields of columns with the same name are equal and returns a single column for each pair of columns that have the same name (since the corresponding fields have the same value).

```
SELECT *
FROM customers c
NATURAL JOIN downloads d NATURAL JOIN games g;
```

`INNER JOIN` keeps all columns. The condition after ON can be any condition (not only equality or two attributes)

You can now use these join queries 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';
```

NUS National University of Singapore

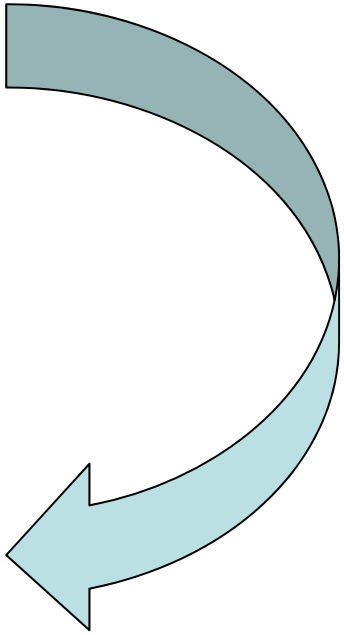
Exercise

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

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

Idea: we pad with null values

first_name	last_name	email	dob	since	customers.customerid	co	downloads.customerid	name	version
...									
Jacqueline	Graham	jgrahamrq@addthis.com	11/29/1995	1/4/2016	Jacqueline 1995	Singapore	queline 1	Flowdesk	1.0
Jacqueline	Graham	jgrahamrq@addthis.com	11/29/1995	1/4/2016	Jacqueline 1995	Singapore	Jacque ne 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.

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

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.

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

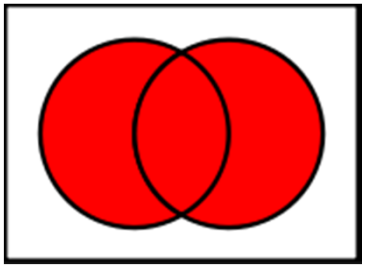
NUS National University of Singapore

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).

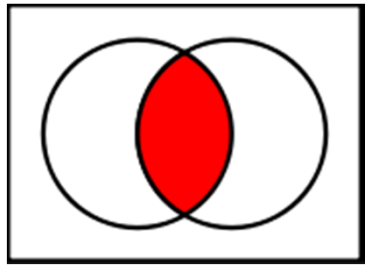
Union eliminates duplicate records. The two queries below have the same result but this may not be the case in general.

```
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).

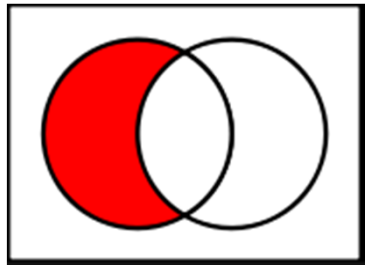
Intersection eliminates duplicate records. The two queries below have the same result but this may not be the case in general.

```
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.

NUS National University of Singapore

Summary

Syntax

1. SELECT
2. FROM
   CROSS JOIN, NATURAL JOIN, INNER JOIN (JOIN), LEFT | RIGHT | FULL OUTER JOIN
3. WHERE
4. UNION, INTERSECT, EXCEPT

Semantics

1. FROM
   CROSS JOIN, NATURAL JOIN, INNER JOIN (JOIN), LEFT | RIGHT | FULL OUTER JOIN
2. WHERE
3. SELECT
4. UNION, INTERSECT, EXCEPT

Unless necessary or otherwise indicated, we prefer simple queries to algebraïc queries.

Credits

The content of this lecture is based on chapter 2 of the book "Introduction to database Systems" By S. Bressan and B. Catania, McGraw Hill publisher

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