Database

SQL

Algebraic Queries

Case Study

Game Store

Requirement

Game Store Requirement

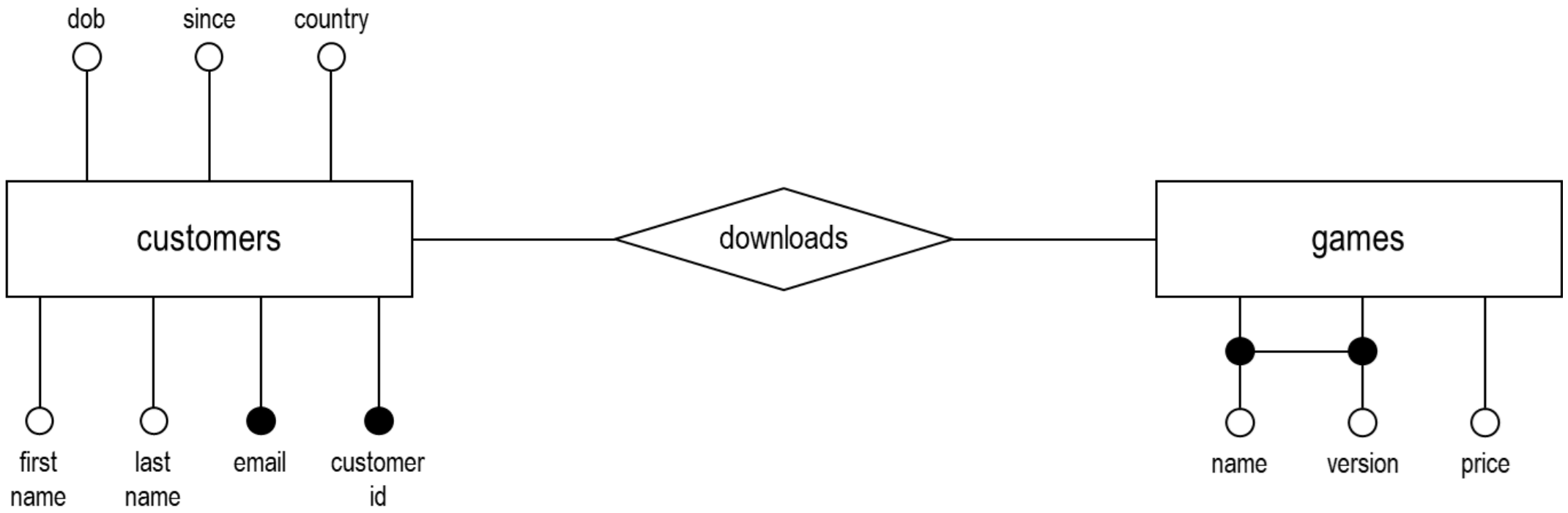
Our company, **Apasaja Pte Ltd**, has been commissioned to develop an application to manage the data of an online app store. We want to store several items of information about our customers such as their first name, last name, date of birth, e-mail, date and country of registration to our online sales service and the customer identifier that they have chosen.

We also want to manage the list of our products, games, their name, their version, and their price. The price is fixed for each version of each game. Finally, our customers buy and download games. We record which version of which game each customer has downloaded. It is not essential to keep the download date for this application.

Case Study

Design

Entity-Relationship Diagram



Joins

Inner Join

Basic

Expressiveness

While **inner join** is a popular construct, there is **no added expressiveness** or performance in `INNER JOIN`. The two queries below are **equivalent**.

Inner Join

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

Cross Join

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

Joins

Inner Join

JOIN

Synonym

`JOIN` is **synonymous** with `INNER JOIN`. We do **NOT** recommend either as `CROSS JOIN` *(or comma)* is typically easier to read and will be optimized by DBMS.

Join

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

Cross Join

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

Joins

Natural Join

What is Natural?

Automatic Equality

If we managed to give the same name to columns with the same meaning across the tables, we can use natural join. `NATURAL JOIN` joins rows that have the same values for columns with the same name. It also prints only one of the two columns.

Natural Join

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

Question

Can you write the equivalent of the query on the left using `CROSS JOIN`?

Joins

Outer Join

Basic

What is Outer?

The **outer join** keeps the columns of the rows in the left *(left outer join)*, the right *(right outer join)*, or in both *(full outer join)* tables that **do not match** anything in the other table according to the join condition. The remaining values are **padded** with `NULL` values.

Warning

It is better to **avoid outer joins** whenever possible as they introduce `NULL` values. They can sometimes be justified for efficiency reasons. However, this course does not care about efficiency as long as the query can finish within reasonable time.

Note

There are also the **natural** variant of outer joins. For instance, `NATURAL LEFT OUTER JOIN` is a natural version of **left join**.

The meaning is the **combination** of natural join *(i.e., automatic equality)* and left join *(i.e., keeps unmatched column, padded with **NULL**)*.

Outer Join

Example

Left Outer Join

In the example below, the customers --*from the left table*-- who never downloaded a game are combined with **NULL** values to replace missing values for the columns of the **downloads** table. Columns from the right table are padded with **NULL** values.

```
SELECT c.customerid, c.email, d.customerid, d.name, d.version
FROM customers c LEFT OUTER JOIN downloads d ON c.customerid = d.customerid;
```

c.customerid	c.email	d.customerid	d.name	d.version
...				
"Willie90"	"wlongjj@moonfruit.com"	"Willie90"	"Ronstring"	"1.1"
"Willie90"	"wlongjj@moonfruit.com"	"Willie90"	"Veribet"	"2.1"
"Al8"	"ahansenp3@webnode.com"	null	null	null
"Johnny1997"	"jstevensb0@un.org"	null	null	null

Joins

Outer Join

Example

Right Outer Join

In the example below, the games --*from the right table*-- that have never been downloade are combined with `NULL` values to replace missing values for the columns of the `downloads` table. Columns from the left table are padded with `NULL` values.

```
SELECT *
FROM downloads d RIGHT OUTER JOIN games g ON g.name = d.name AND g.version = d.version;
```

Full Outer Join

A **full outer join** pads missing values with `NULL` for both the tables on the left and on the right.

Joins

Outer Join

Anti Join

Only Missing Value

Find customers who never downloaded a game.

```
SELECT c.customerid
FROM customers c
  LEFT OUTER JOIN downloads d
    ON c.customerid = d.customerid
WHERE d.customerid IS NULL;
```

Further Restriction

Further restriction should be on `WHERE` clause and not `ON` clause.

```
SELECT c.customerid
FROM customers c
  LEFT OUTER JOIN downloads d
    ON c.customerid = d.customerid
WHERE d.customerid IS NULL
  AND c.country = 'Singapore';
-- try moving the AND above
```

Joins

Outer Join

Closing

Warning

Outer joins are not easy to write as conditions in the `ON` clause are not equivalent to conditions in the `WHERE` clause *(as it was the case with `INNER JOIN`)*. Conditions in the `ON` clause determines which rows are **dangling**.

Synonyms

- `LEFT JOIN` is synonym for `LEFT OUTER JOIN`
- `RIGHT JOIN` is synonym for `RIGHT OUTER JOIN`
- `FULL JOIN` is synonym for `FULL OUTER JOIN`

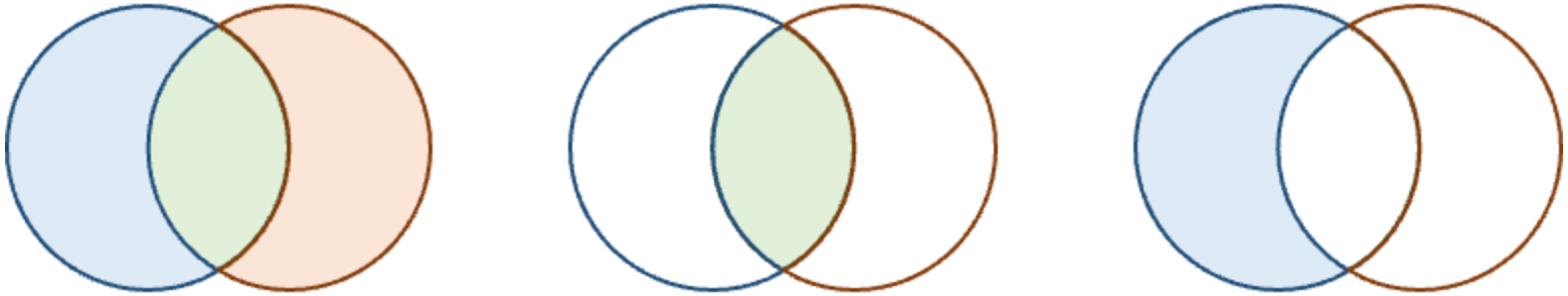
Set Operations

Set

Basic

Operators

The set operators `UNION`, `INTERSECT`, and `EXCEPT` return the **union**, **intersection**, and **non-symmetric difference** of the results of two queries respectively.



Deduplication

Union, intersection, and non-symmetric difference **eliminate duplicates** unless annotated with the keyword ALL.

Set Operations

Set

Compatible

Union-Compatible

Two queries must be **union-compatible** to be used with `UNION`, `INTERSECT`, or `EXCEPT`. They must return the same number of columns with the same domain in the same order.

Compatible

student.name (`VARCHAR(32)`)

department.department (`VARCHAR(32)`)

Incompatible

student.year (`DATE`)

department.department (`VARCHAR(32)`)

Note

Just because they are **union-compatible** does not mean it is **meaningful** to perform set operations on the two tables.

Set Operations

Examples

Union

Question

Find the `customerid` of all the **customers** who **downloaded** version 1.0 or 2.0 of the game Aerified.

```sql
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '1.0'
UNION
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '2.0';
```

Set Operations

Examples

Union

Question

Find the `name` and `versions` of all the games after GST is applied. GST of 9% is applied if it is more than 30 cents.

```sql
SELECT g.name || ' ' || g.version AS game, ROUND(g.price * 1.09) AS price
FROM games g
WHERE g.price * 0.09 >= 0.3
UNION
SELECT g.name || ' ' || g.version AS game, g.price
FROM games g
WHERE g.price * 0.09 < 0.3;
```

Set Operations

Examples

Intersection

Question

Find the `customerid` of all the **customers** who **downloaded** both **version 1.0 and 2.0** of the game **Aerified**.

```sql
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '1.0'
INTERSECT
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '2.0';
```

Set Operations

Examples

Difference

Question

Find the `customerid` of the **customers** who **downloaded version 1.0 but not version 2.0** of the game Aerified.

```sql
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '1.0'
EXCEPT
SELECT d.customerid
FROM downloads d
WHERE d.name = 'Aerified' AND d.version = '2.0';
```

Conclusion

Reading

What Does This Query Find?

```
SELECT c.email, SUM(g.price)
FROM customers c, downloads d, games g
WHERE c.customerid = d.customerid AND g.name = d.name
  AND g.version = d.version AND c.country = 'Indonesia' AND g.name=  'Fixflex'
GROUP BY c.email
UNION
SELECT c.email, 0
FROM customers c LEFT JOIN
  (downloads d INNER JOIN games g
    ON g.name = d.name AND g.version = d.version AND g.name = 'Fixflex')
  ON c.customerid = d.customerid
WHERE c.country = 'Indonesia' AND d.name IS NULL;
```

```
postgres=# exit
Press any key to continue . . .
```