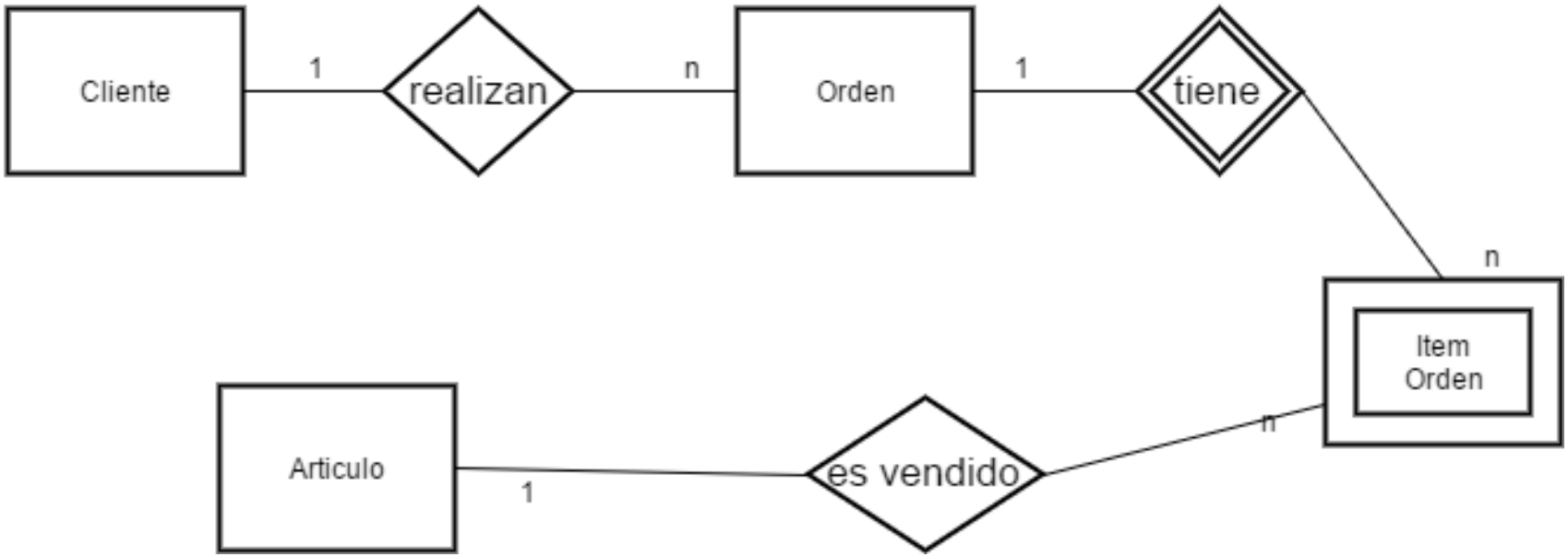
Un poco sobre SQL

Lic. Gerardo Rossel

Bases de Datos

1C 2016

Esquema


Cliente
1
realizan
n
Orden
1
tiene
n
Item
Orden
Articulo
1
es vendido
n

Esquema

- Cliente(clienteId, nombre, ciudad)
- Orden(ordenId, ClienteId, Fecha)
- ItemOrden(ordenId, articuloId, cantidad)
- Articulo(articuloId, nombre, precio, categoría)

Consulta

Resolver en SQL

Obtener los clientes que viven en Gotham y ordenaron un Batimovil después de comienzo de año‘

- Cliente(clienteId, nombre, ciudad)
- Orden(ordenId, ClienteId, Fecha)
- ItemOrden(ordenId, articuloId, cantidad)
- Articulo(articuloId, nombre, precio, categoría)

SQL

```
SELECT DISTINCT c.nombre FROM
Cliente c join Orden o  on o.clienteId = c.clienteId
join ItemOrden io on io.ordenId = o.ordenId
join Articulo a on a.articuloId = io.articuloID
WHERE c.ciudad = "Gotham"
and a.nombre = "Batimovil"
and o.Fecha >= '1/01/2016'
```

SQL

```
SELECT DISTINCT c.nombre FROM
Cliente c join Orden o  on o.clienteId = c.clienteId
join ItemOrden io on io.ordenId = o.ordenId
join Articulo a on a.articuloId = io.articuloID
WHERE c.ciudad = "Gotham"
and a.nombre = "Batimovil"
and o.Fecha >= '1/01/2016'
```

DISTINC

¿Que pasa con el DISTINC?

SQL

```
SELECT DISTINCT c.nombre FROM Cliente c, Orden o,
    ItemOrden io, Articulo a
WHERE c.ciudad = "Gotham"
and c.clienteId = o.clienteId
and o.ordenId = io.ordenId
and io.articuloId = a.articuloID
and a.nombre = "Batimovil"
and o.Fecha >= '1/01/2016'
```

SQL

```
SELECT DISTINCT c.nombre FROM Cliente c, Orden o,
    ItemOrden io, Articulo a
WHERE c.ciudad = "Gotham"
and c.clienteId = o.clienteId
and o.ordenId = io.ordenId
and io.articuloId = a.articuloID
and a.nombre = "Batimovil"
and o.Fecha >= '1/01/2016'
```

DISTINC

¿Que pasa con el DISTINC? ¿Hay otras alternativas?

Usando Subqueries

La misma consulta usando *Exists*

Usando Subqueries

La misma consulta usando *Exists*

```
SELECT  c.nombre FROM Cliente c
WHERE c.ciudad = "Gotham"
and EXISTS( SELECT null From Orden o, ItemOrden io,
    Articulo a
WHERE a.nombre = "Batimovil"
and io.articuloId = a.articuloID
and o.ordenId = io.ordenId
and o.clienteId =c.clienteId  )
and o.Fecha >= '1/01/2016')
```

Usando Subqueries

La misma consulta usando *Exists*

```
SELECT  c.nombre FROM Cliente c
WHERE c.ciudad = "Gotham"
and EXISTS( SELECT null From Orden o, ItemOrden io,
    Articulo a
WHERE a.nombre = "Batimovil"
and io.articuloId = a.articuloID
and o.ordenId = io.ordenId
and o.clienteId =c.clienteId  )
and o.Fecha >= '1/01/2016')
```

Usando Subqueries

Correlacionada. No se puede ejecutar antes de que se conozca el Cliente

Usando Subqueries

La misma consulta usando *IN*. No Correlacionada

Usando Subqueries

La misma consulta usando *IN*. No Correlacionada

```
SELECT  c.nombre FROM Cliente c
WHERE c.ciudad = "Gotham"
and clienteId IN
( SELECT clienteId
From Orden o, ItemOrden io, Articulo a
WHERE a.nombre = "Batimovil"
and io.articuloId = a.articuloID
and o.ordenId = io.ordenId
and o.Fecha >= '1/01/2016')
```

Usando Subqueries

Mas anidamiento...

Usando Subqueries

Mas anidamiento...

```
SELECT  c.nombre FROM Cliente c
WHERE c.ciudad = "Gotham"
and clienteId IN
(SELECT clienteId From Orden o
WHERE o.Fecha >= '1/01/2016'
and ordenId IN
( SELECT io.OrdenId
FROM ItemOrden io, Articulo a
WHERE a.nombre = "Batimovil"
and io.articuloId = a.articuloID
)
)
```

Conclusiones I

Hay muchisimas maneras de resolver una consulta. La mejor manera dependerá de los datos en sí y de la organización física.

Conclusiones I

Hay muchisimas maneras de resolver una consulta. La mejor manera dependerá de los datos en sí y de la organización física.

Conclusiones II

Tomar en cuenta que como escribimos la consulta afecta su velocidad principalmente en la parte No Relacional

El problemático NULL

NULL

Mucho cuidado con los valores **NULL**

- *precio* + *NULL* devuelve *NULL*
- *precio* < *NULL* devuelve *UNKNOWN*

El problemático NULL

NULL

Mucho cuidado con los valores **NULL**

- *precio* + *NULL* devuelve *NULL*
- *precio* < *NULL* devuelve *UNKNOWN*

CONTEXTUALIZAR NULL

¿Que significa que un precio es NULL?

El problemático NULL

WHERE y HAVING

El SQL elimina filas para las cuales el WHERE/HAVING **no evaluan** TRUE. No es lo mismo que sacar las que evaluan FALSE.

¿Que pasa con CHECK?

El problemático NULL

WHERE y HAVING

El SQL elimina filas para las cuales el WHERE/HAVING **no evaluan** TRUE. No es lo mismo que sacar las que evaluan FALSE.

¿Que pasa con CHECK?

A diferencia de WHERE/HAVING el CHECK debe NO evaluar a falso para conformar la restricción.

Lógica de tres valores

AND	true	false	unknown
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

OR	true	false	unknown
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

NOT	
true	false
false	true
unknown	unknown

NULL con IN

Cómo se evalúa: $x\ IN\ (y_1, ..., y_n)$,

NULL con IN

Cómo se evalúa: $x\ IN\ (y_1, ..., y_n)$,

- Si al menos una de las comparaciones $x = y_i$ evalúa a *true* la condición evalúa a *true*
- Si todas las comparaciones $x = y_i$ evalúan a *false* o la lista esta vacía entonces la condición evalúa a *false*
- Si ninguno de estos casos se cumple, entonces la condición devuelve *unknown*.

NULL con IN

Cómo se evalúa: $x\ IN\ (y_1, ..., y_n)$,

- Si al menos una de las comparaciones $x = y_i$ evalúa a *true* la condición evalúa a *true*
- Si todas las comparaciones $x = y_i$ evalúan a *false* o la lista esta vacía entonces la condición evalúa a *false*
- Si ninguno de estos casos se cumple, entonces la condición devuelve *unknown*.

¿Que pasa con $x\ NOT\ IN\ (y_1, ..., y_n)$?

NULL con IN

Cómo se evalúa: $x\ IN\ (y_1, ..., y_n)$,

- Si al menos una de las comparaciones $x = y_i$ evalúa a *true* la condición evalúa a *true*
- Si todas las comparaciones $x = y_i$ evalúan a *false* o la lista esta vacía entonces la condición evalúa a *false*
- Si ninguno de estos casos se cumple, entonces la condición devuelve *unknown*.

¿Que pasa con $x\ NOT\ IN\ (y_1, ..., y_n)$?

- Es equivalente a $NOT\ x\ IN\ (y_1, ..., y_n)$

El problemático NULL

- Obtener los empleados que no tienen gente a cargo

```
(1,'Juan (el dueño)', null);
(2,'Pedro Perez', 1);
(3,'Maria Lopez', 2);
(4,'Pepin Gonzalez', 2);
```

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

```
SELECT E1.nombre FROM empleados E1
WHERE E1.legajo NOT IN
  (SELECT E2.legGer FROM empleados E2);

  (1,'Juan (el dueño)', null);
  (2,'Pedro Perez', 1);
  (3,'Maria Lopez', 2);
  (4,'Pepin Gonzalez', 2);
```

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

```
SELECT E1.nombre FROM empleados E1
WHERE E1.legajo NOT IN
  (SELECT E2.legGer FROM empleados E2);

  (1,'Juan (el dueño)', null);
  (2,'Pedro Perez', 1);
  (3,'Maria Lopez', 2);
  (4,'Pepin Gonzalez', 2);

SELECT E1.nombre FROM empleados E1
WHERE NOT EXISTS (SELECT E2.* FROM empleados E2
            WHERE E2.legmgr = E1.legajo);
```

COALESCE

COALESCE

$COALESCE(< expr1 >, < expre2 >, < expre3 > ....)$

- COALESCE retorna la primer expresión no NULL de una lista de expresiones.
- Al menos una expresión **no** debe ser el literal NULL
- Si todas las ocurrencias evalúan a NULL la función retorna NULL.

El problemático NULL

```
DECLARE
@x AS INT = NULL,
@y AS INT = 1,
@z AS INT = 2;

SELECT COALESCE(@x, @y, @z);
```

El problemático NULL

```
DECLARE
@x AS INT = NULL,
@y AS INT = 1,
@z AS INT = 2;

SELECT COALESCE(@x, @y, @z);
```

Devuelve 1

El problemático NULL

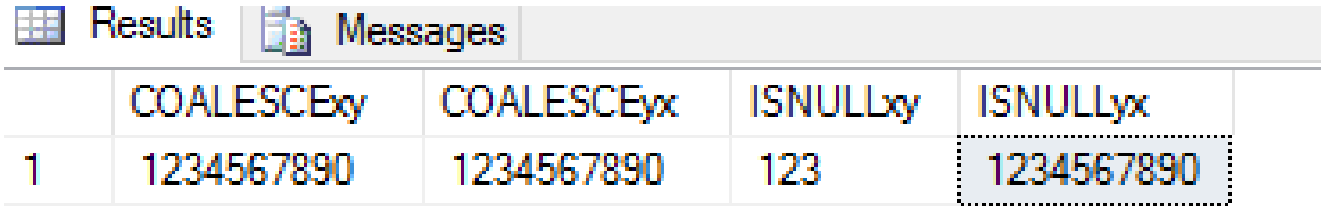
```
DECLARE
@x AS VARCHAR(3) = NULL,
@y AS VARCHAR(10) = '1234567890';

SELECT
  COALESCE(@x, @y) AS COALESCExy, COALESCE(@y, @x)
  AS COALESCEyx, ISNULL(@x, @y) AS ISNULLxy,
  ISNULL(@y, @x) AS ISNULLyx;
```

El problemático NULL

```
DECLARE
@x AS VARCHAR(3) = NULL,
@y AS VARCHAR(10) = '1234567890';

SELECT
  COALESCE(@x, @y) AS COALESCExy, COALESCE(@y, @x)
  AS COALESCEyx, ISNULL(@x, @y) AS ISNULLxy,
  ISNULL(@y, @x) AS ISNULLyx;
```



	COALESCExy	COALESCEyx	ISNULLxy	ISNULLyx
1	1234567890	1234567890	123	1234567890

CASE

Lista de Valores

```
CASE <target expression>
WHEN <candidate expression> THEN <result expression>
WHEN <candidate expression> THEN <result expression>
...
WHEN <candidate expression> THEN <result expression>
[ELSE <result expression>]
END
```

CASE

Lista de Valores

```
CASE <target expression>
WHEN <candidate expression> THEN <result expression>
WHEN <candidate expression> THEN <result expression>
...
WHEN <candidate expression> THEN <result expression>
[ELSE <result expression>]
END
```

Lista Condicional

```
CASE
WHEN <match conditional> THEN <result expression>
WHEN <match conditional> THEN <result expression>
...
WHEN <match conditional> THEN <result expression>
[ELSE <result expression>]
END
```

CASE

Ejemplo

```
SELECT cust_last_name, limite =
  (CASE credit_limit WHEN 100 THEN 'Low'
  WHEN 5000 THEN 'High'
  ELSE 'Medium' END)
FROM customers;
```

CASE

Ejemplo

```
SELECT cust_last_name, limite =
  (CASE credit_limit WHEN 100 THEN 'Low'
  WHEN 5000 THEN 'High'
  ELSE 'Medium' END)
FROM customers;

SELECT nombre, apellido =
    (CASE WHEN sueldo > 35000 THEN 'Afectado'
      ELSE 'No Afectado' END)
FROM empleados;
```