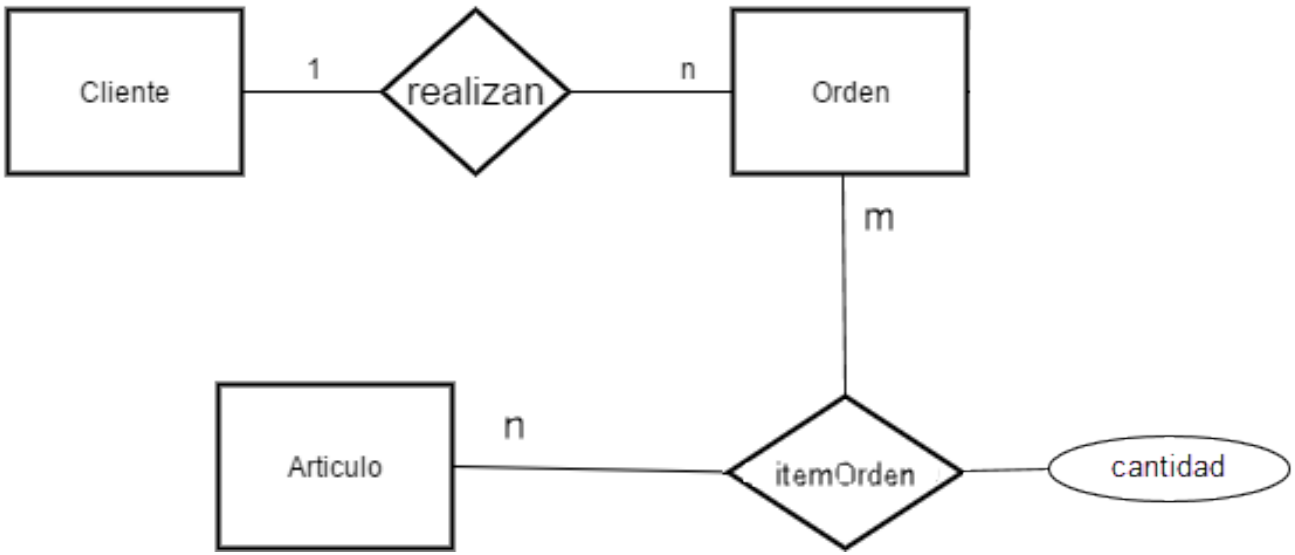
Un poco sobre SQL

Dr. Gerardo Rossel

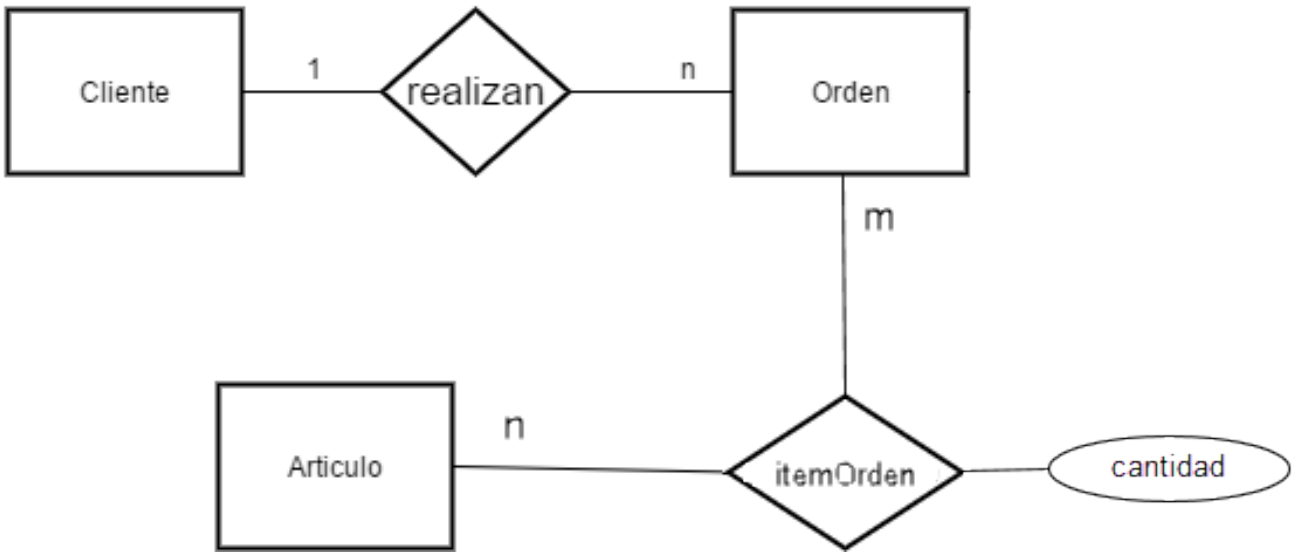
Bases de Datos

2024

Esquema



Esquema



- Cliente(<u>idCliente</u>, nombre, ciudad)
- Orden(<u>idOrden</u>, <u>idCliente</u>, Fecha)
- ItemOrden(<u>idOrden</u>, <u>idArticulo</u>, cantidad)
- Articulo(<u>idArticulo</u>, nombre, precio, categoría)

Consulta

Resolver en SQL

Obtener los clientes que viven en **Gotham** y ordenaron un **Batimovil** en los últimos 6 meses.

- Cliente(idCliente, nombre, ciudad)
- Orden(idOrden, idCliente, Fecha)
- ItemOrden(idOrden, idArticulo, cantidad)
- Articulo(idArticulo, nombre, precio, categoría)

SQL

```sql
SELECT DISTINCT
       c.RazonSocial,
       o.fecha
FROM Cliente c
     JOIN Orden o ON o.idCliente = c.idCliente
     JOIN ItemOrden io ON io.idOrden = o.idOrden
     JOIN Articulo a ON a.idArticulo = io.idArticulo
WHERE c.ciudad = 'Gotham'
      AND a.nombre = 'Batimovil'
      AND DATEDIFF(month, o.Fecha, GETDATE()) < 6;
```

```sql
SELECT DISTINCT
       c.RazonSocial,
       o.fecha
FROM Cliente c
     JOIN Orden o ON o.idCliente = c.idCliente
     JOIN ItemOrden io ON io.idOrden = o.idOrden
     JOIN Articulo a ON a.idArticulo = io.idArticulo
WHERE c.ciudad = 'Gotham'
      AND a.nombre = 'Batimovil'
      AND DATEDIFF(month, o.Fecha, GETDATE()) < 6;
```

DISTINCT

¿Que pasa con el DISTINCT?

SQL -Consultas anidadas

La misma consulta usando *Exists*

SQL -Consultas anidadas

La misma consulta usando *Exists*

```sql
SELECT c.RazonSocial
FROM Cliente c
WHERE c.ciudad = 'Gotham'
      AND EXISTS
(
    SELECT NULL
    FROM Orden o,
         ItemOrden i,
         Articulo a
    WHERE a.nombre = 'Batimovil'
          AND i.idArticulo = a.idArticulo
          AND o.idOrden = i.idOrden
          AND o.idCliente = c.idCliente
          AND DATEDIFF(month, o.Fecha, GETDATE()) < 6
);
```

SQL -Consultas anidadas

La misma consulta usando *Exists*

```
SELECT c.RazonSocial
FROM Cliente c
WHERE c.ciudad = 'Gotham'
      AND EXISTS
(
    SELECT NULL
    FROM Orden o,
         ItemOrden i,
         Articulo a
    WHERE a.nombre = 'Batimovil'
          AND i.idArticulo = a.idArticulo
          AND o.idOrden = i.idOrden
          AND o.idCliente = c.idCliente
          AND DATEDIFF(month, o.Fecha, GETDATE()) < 6
);
```

Usando Subqueries

Correlacionada.Parece que no se puede ejecutar la subquery antes de que se conozca el Cliente. Pero los optimizadores pueden generar el mismo árbol

SQL - Consultas anidadas

La misma consulta usando *IN*. **No Correlacionada**

SQL - Consultas anidadas

La misma consulta usando *IN*. **No Correlacionada**

```
SELECT c.RazonSocial
FROM Cliente AS c
WHERE c.ciudad = 'Gotham' AND
      c.idCliente IN ( SELECT c.idCliente
                       FROM Orden AS o , ItemOrden AS io , Articulo AS a
                       WHERE a.nombre = 'Batimovil' AND
                             io.idArticulo = a.idArticulo AND
                             o.idOrden = io.idOrden AND
                             DATEDIFF(month , o.Fecha , GETDATE()) < 6
                     );
```

SQL - Consultas anidadas

Más anidamiento...

SQL - Consultas anidadas

Más anidamiento...

```
SELECT c.RazonSocial
FROM Cliente c
WHERE c.ciudad = 'Gotham'
      AND idCliente IN
(
    SELECT idCliente
    FROM Orden o
    WHERE DATEDIFF(month, o.Fecha, GETDATE()) < 6
          AND idOrden IN
    (
        SELECT i.idOrden
        FROM ItemOrden i,
             Articulo a
        WHERE a.nombre = 'Batimovil'
              AND i.idArticulo = a.idArticulo
    )
);
```

Conclusiones I

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

Conclusiones I

Hay muchísimas 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 evalúan** TRUE. No es lo mismo que sacar las que evalúan FALSE.

¿Que pasa con CHECK?

El problemático NULL

WHERE y HAVING

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

	EmployeeID	FirstName	LastName	Title	DeptID	ManagerID
1	1	Ken	Sánchez	Chief Executive Officer	16	NULL
2	16	David	Bradley	Marketing Manager	4	273
3	23	Mary	Gibson	Marketing Specialist	4	16
4	273	Brian	Welcker	Vice President of Sales	3	1
5	274	Stephen	Jiang	North American Sales Manager	3	273
6	275	Michael	Blythe	Sales Representative	3	274
7	276	Linda	Mitchell	Sales Representative	3	274
8	285	Syed	Abbas	Pacific Sales Manager	3	273
9	286	Lynn	Tsoflias	Sales Representative	3	285

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

	EmployeeID	FirstName	LastName	Title	DeptID	ManagerID
1	1	Ken	Sánchez	Chief Executive Officer	16	NULL
2	16	David	Bradley	Marketing Manager	4	273
3	23	Mary	Gibson	Marketing Specialist	4	16
4	273	Brian	Welcker	Vice President of Sales	3	1
5	274	Stephen	Jiang	North American Sales Manager	3	273
6	275	Michael	Blythe	Sales Representative	3	274
7	276	Linda	Mitchell	Sales Representative	3	274
8	285	Syed	Abbas	Pacific Sales Manager	3	273
9	286	Lynn	Tsoflias	Sales Representative	3	285

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

	EmployeeID	FirstName	LastName	Title	DeptID	ManagerID
1	1	Ken	Sánchez	Chief Executive Officer	16	NULL
2	16	David	Bradley	Marketing Manager	4	273
3	23	Mary	Gibson	Marketing Specialist	4	16
4	273	Brian	Welcker	Vice President of Sales	3	1
5	274	Stephen	Jiang	North American Sales Manager	3	273
6	275	Michael	Blythe	Sales Representative	3	274
7	276	Linda	Mitchell	Sales Representative	3	274
8	285	Syed	Abbas	Pacific Sales Manager	3	273
9	286	Lynn	Tsoflias	Sales Representative	3	285

```
SELECT E1.FirstName, E1.LastName FROM MyEmployees E1
WHERE E1.EmployeeID NOT IN
        (SELECT E2.ManagerID FROM MyEmployees E2)
```

El problemático NULL - Exists vs In

- Obtener los empleados que no tienen gente a cargo

	EmployeeID	FirstName	LastName	Title	DeptID	ManagerID
1	1	Ken	Sánchez	Chief Executive Officer	16	NULL
2	16	David	Bradley	Marketing Manager	4	273
3	23	Mary	Gibson	Marketing Specialist	4	16
4	273	Brian	Welcker	Vice President of Sales	3	1
5	274	Stephen	Jiang	North American Sales Manager	3	273
6	275	Michael	Blythe	Sales Representative	3	274
7	276	Linda	Mitchell	Sales Representative	3	274
8	285	Syed	Abbas	Pacific Sales Manager	3	273
9	286	Lynn	Tsoflias	Sales Representative	3	285

```
SELECT E1.FirstName, E1.LastName FROM MyEmployees E1
WHERE E1.EmployeeID NOT IN
        (SELECT E2.ManagerID FROM MyEmployees E2)

SELECT E1.FirstName, E1.LastName FROM MyEmployees E1
WHERE NOT EXISTS (SELECT E2.* FROM MyEmployees E2
            WHERE E2.ManagerID = E1.EmployeeID);
```

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;
```

Results | Messages

	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;
```

CASE

Supongamos una vista *vTableInfo* que contiene nombre de las tablas de la Base de Datos y la cantidad de filas.

Consulta

Cuantas tablas contienen:

- menos que 100 rows,
- cuantas contienen entre 100 y 10000 rows,
- cuantas entre 10000 y 1000000 rows, y
- cuantas más de 1000000 rows

CASE

```
select case
         when [TotalRowCount] < 100
              then 'Menos que 100 rows'
          when [TotalRowCount] >= 100 and [TotalRowCount] < 10000
               then '100 a 10000'
          when [TotalRowCount] >= 10000 and [TotalRowCount] < 1000000
               then '10000 a 1000000'
          else
               'mas de 1000000 rows'
        end as range,
        count(*) as table_count
from vTableInfo
where [TotalRowCount] is not null
group by case
           when [TotalRowCount] < 100
                then 'Menos que 100 rows'
           when [TotalRowCount] >= 100 and [TotalRowCount] < 10000
                then '100 a 10000'
           when [TotalRowCount] >= 10000 and [TotalRowCount] < 1000000
                then '10000 a 1000000'
           else
                'mas de 1000000 rows'
         end
```

MAS PROBLEMAS

Direcciones

Tenemos una tabla de Clientes y una tabla de Direcciones de envío. Si no hay dirección de envío se debe enviar a la dirección que figura en cliente.¿Cómo hacemos?

MAS PROBLEMAS

Direcciones

Tenemos una tabla de Clientes y una tabla de Direcciones de envío. Si no hay dirección de envío se debe enviar a la dirección que figura en cliente.¿Cómo hacemos?

- Que el programador busque primero en la tabla de direcciones de envío y si no encuentra entonces en la otra.

MAS PROBLEMAS

Direcciones

Tenemos una tabla de Clientes y una tabla de Direcciones de envío. Si no hay dirección de envío se debe enviar a la dirección que figura en cliente.¿Cómo hacemos?

- Que el programador busque primero en la tabla de direcciones de envío y si no encuentra entonces en la otra.
- Full OUTER JOIN con Coalesce

MAS PROBLEMAS

Direcciones

Tenemos una tabla de Clientes y una tabla de Direcciones de envío. Si no hay dirección de envío se debe enviar a la dirección que figura en cliente.¿Cómo hacemos?

- Que el programador busque primero en la tabla de direcciones de envío y si no encuentra entonces en la otra.
- Full OUTER JOIN con Coalesce

MAS PROBLEMAS

Direcciones

Tenemos una tabla de Clientes y una tabla de Direcciones de envío. Si no hay dirección de envío se debe enviar a la dirección que figura en cliente.¿Cómo hacemos?

- Que el programador busque primero en la tabla de direcciones de envío y si no encuentra entonces en la otra.
- Full OUTER JOIN con Coalesce

```
SELECT  coalesce(d.CalleLinea1, c.CalleLinea1),
  coalesce(d.CalleLinea2, c.CalleLinea2),
  coalesce(d.Nro, c.Nro), coalesce(d.Ciudad, c.Ciudad),
  coalesce(d.CodigoPostal, c.CodigoPostal)
FROM Cliente c FULL OUTER JOIN DireccionEnvio d ON c.
    idCliente = d.idCliente  WHERE c.idCliente = 2;
```