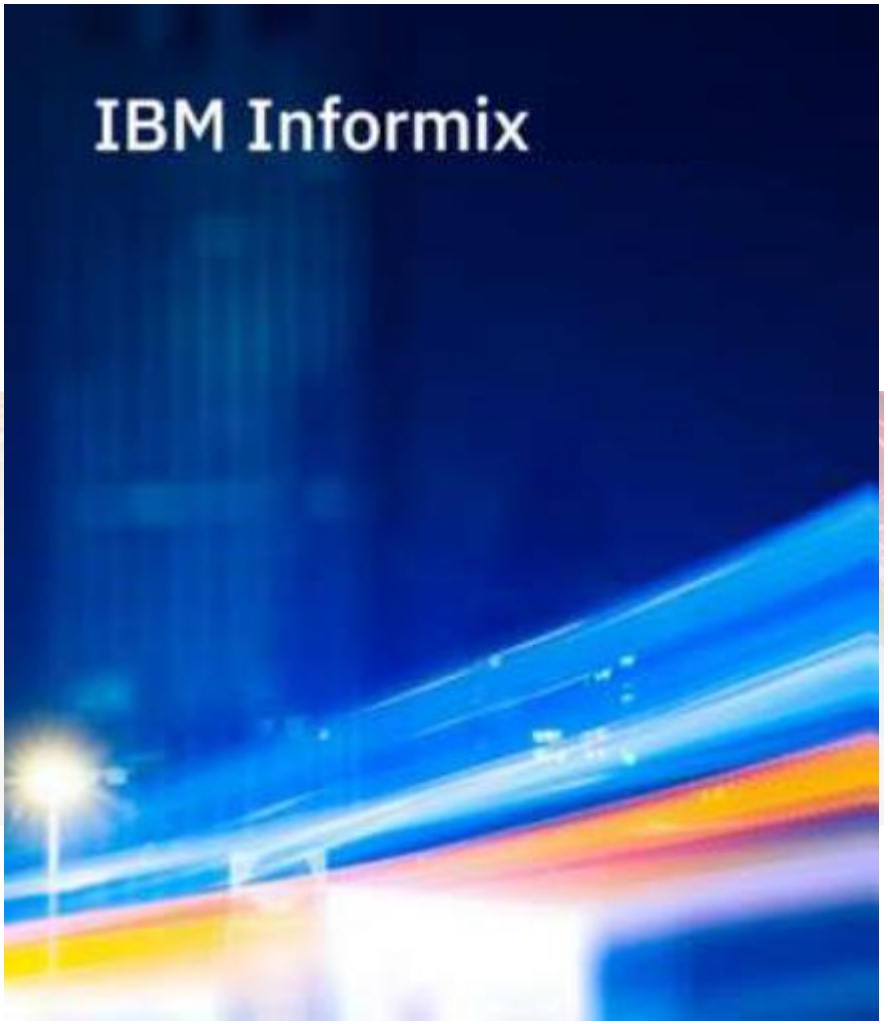
IBM Inform*ix*

Mejores prácticas para escribir queries.

Aquiles Loranca.
IBM Cloud Services.
Certified L3 IT Specialist.
aloranca@mx1.ibm.com



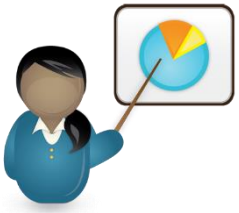
Agenda

- OLAP vs. OLTP.
- Optimizador.
- Estadísticas
- Planes de ejecución.
- Indices.
- JOINs
- UNION.
- Subqueries.
- Otras consideraciones.
- Preguntas.

Relacionamiento con el cliente (CRM)

Sistemas operacionales

Nómina



Sistemas de Información Gerencial

Sistemas de Información Geográfica

Planeación empresarial de recursos (ERP)

Sistemas de Información Financiera

Tipos de ambientes de bases de datos.

OLTP: (OnLine Transaction Processing)

- Muchos usuarios concurrentes.
- UPDATE, INSERT, DELETE, SELECT.
- Afectando unos cuantos registros por sesión.
- Soportan la operación.

OLAP: (OnLine Analytical Processing)

- Pocos usuarios concurrentes.
- SELECTs muy pesados, cargas masivas en otros horarios.
- Los SELECTs pueden leer millones de registros.
- Soportan la toma de decisiones.

Lo que es bueno para uno,
puede ser fatal para el otro.

OLTP

Factores a favor:

- Integridad referencial.
- **Indices**.
- Normalización (3FN)
- Separe datos de índices

Factores en contra:

- **No abuse de los índices.**
- **No sobre-normalice**.
- **Depure**. Guarde sólo la información necesaria para poder operar.

OLAP

Factores a favor:

- Bases de datos **columnares en memoria**.
- Fragmentación. (Divide y vencerás)
- Paralelismo.
- Tabla de hechos. (1FN)
  - JOINs prehechos.
  - Sumarizados.

Factores en contra:

- Los indices convencionales pueden ser perjudiciales.
- Alto consumo de memoria por agregados.
- Incluya solo las columnas necesarias.

Optimizador

- Es el responsable de decidir como va a resolverse un query.
- Evalúa las diferentes opciones.
- Sus decisiones son tan **buenas**, como la **información** que tenga para tomarlas.

¡Existe más de una forma de matar pulgas!

Estadísticas.

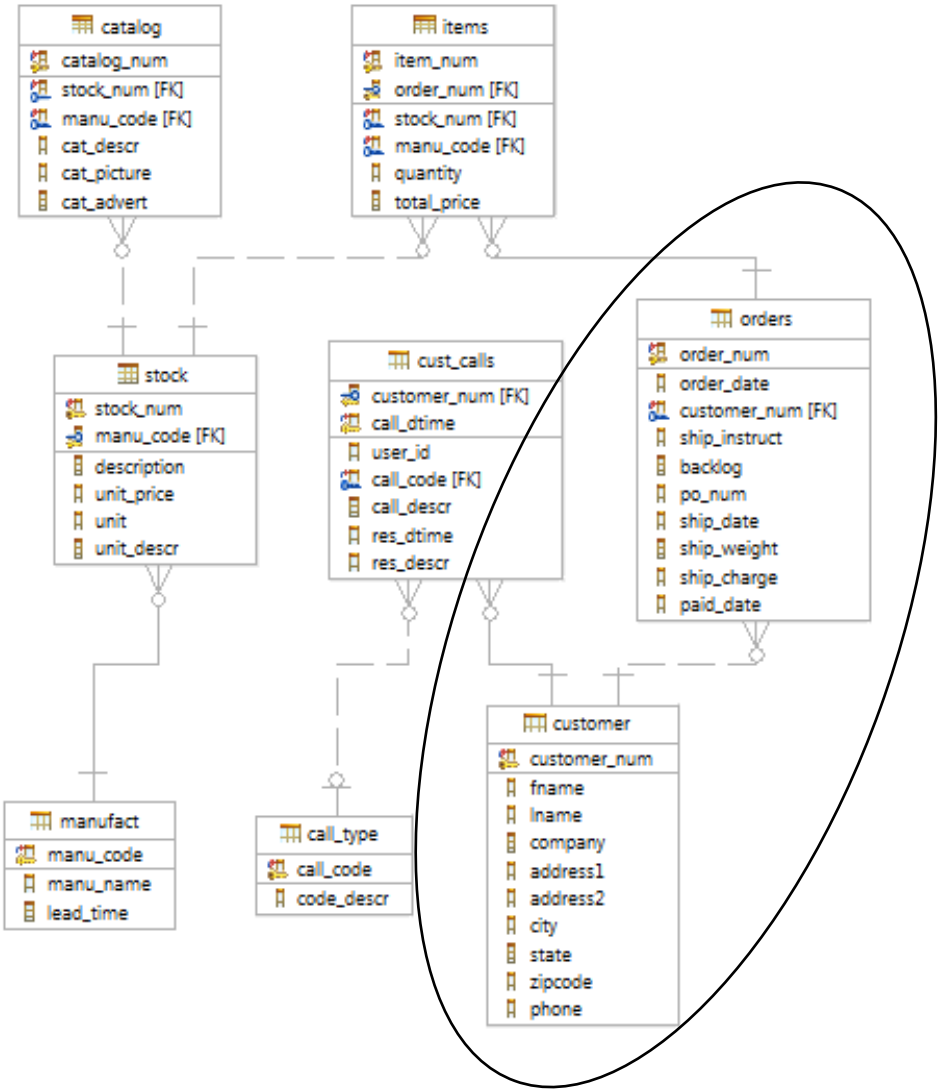
- Comportamiento de los datos en las tablas.
- Histogramas.
- Actualizadas.
- Precisión.
- Ejecutarse periódicamente, es especialmente después de cargas masivas o procesos por lotes.

Un “mismo” query, puede comportarse diferente con argumentos distintos.

Planes de ejecución.

- ¿Cómo resolvemos el query?
- ¿Qué tabla leemos primero?
- ¿Cómo la leemos?
- ¿Cómo hacemos un JOIN?
- ¿Ignoro fragmentos?
- ¿Paralelizo lecturas?
- ¿Hago uso de algún acelerador? (IWA)

Planes de ejecución.




```
SELECT a.fname, a.lname, b.order_num
  FROM customer a, orders b
 WHERE a.customer_num=b.customer_num
   AND a.lname=? ;
```

- ¿Cuál leo primero? (2)
- ¿Uso índices, o secuencial? (2) * (2)
- ¿Tipo de JOIN? (Nested, Hash, Merge) (3)

24 Posibles opciones.

Puede no ser lo mismo hacer el query

por:

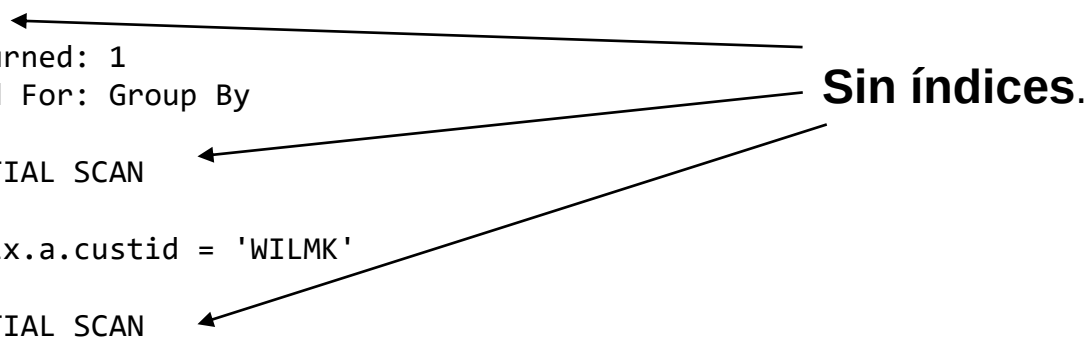
lname="Sanchez"

que por:

lname="Loranca"

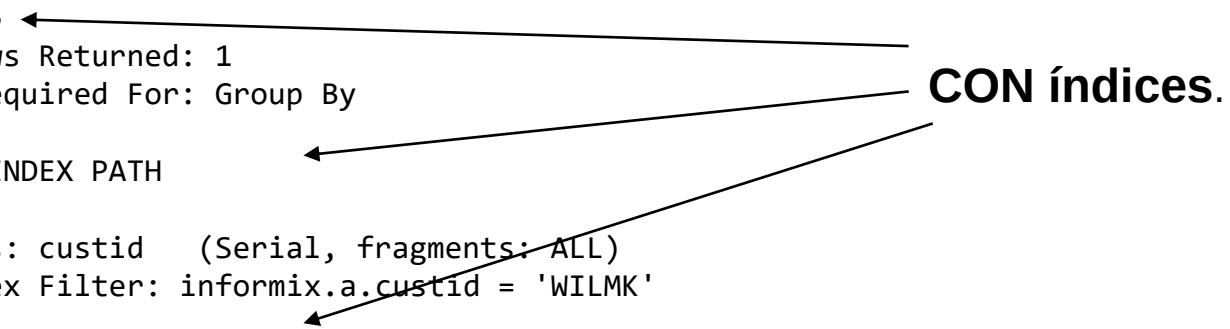
SET EXPLAIN

```
QUERY:
------
select companyname, address, city, state, country, phone, fax, email, count(*)
from customer a, order b
where a.custid = b.custid
and a.custid = 'WILMK'
group by 1,2,3,4,5,6,7,8

Estimated Cost: 1135675
Estimated # of Rows Returned: 1
Temporary Files Required For: Group By

  1) omcadmin.a: SEQUENTIAL SCAN

        Filters: informix.a.custid = 'WILMK'

  2) omcadmin.b: SEQUENTIAL SCAN

        Filters:
        Table Scan Filters: informix.b.custid = 'WILMK'


DYNAMIC HASH JOIN
    Dynamic Hash Filters: informix.a.custid = informix.b.custid
```



SET EXPLAIN

```
QUERY:
------
select companyname, address, city, state, country, phone, fax, email, count(*)
from customer a, order b
where a.custid = b.custid
and a.custid = 'WILMK'
group by 1,2,3,4,5,6,7,8

Estimated Cost: 15
Estimated # of Rows Returned: 1
Temporary Files Required For: Group By

  1) omcadmin.a: INDEX PATH

    (1) Index Keys: custid   (Serial, fragments: ALL)
        Lower Index Filter: informix.a.custid = 'WILMK'

  2) omcadmin.b: INDEX PATH

    (1) Index Keys: custid   (Key-Only)  (Serial, fragments: ALL)
        Lower Index Filter: informix.a.custid = informix.b.custid
NESTED LOOP JOIN
```



¡De 10 minutos de ejecución a 30 segundos!

SET EXPLAIN

- Los “costos” de los queries son adimensionales.
- Son función del uso estimado de disco, cpu y número de registros y otros factores. (Se le dá mayor peso al uso de IO.)
- Solo sirven para comparar las opciones de ESE query, no debe ser usado para comparar queries diferentes.
- No siempre hay que satanizar las búsquedas secuenciales.

A veces un vuelo tarda mucho…

¡Pero es el tiempo que tarda!

Indices.

- Excelentes para encontrar “la aguja en el pajar” en ambientes OLTP…

Pero alenta:

- UPDATEs,
- INSERTs (Y por tanto LOADs)
- DELETEs.

Pueden no ser la mejor opción para ambientes OLAP.

Mejores prácticas para índices.

Evite índices redundantes.

Un índice sobre A,B,C:

- Puede usarse en lugar de: A y AB
- Pero no de: B, C, BC, AC, BA.
- Multi-Index Scan (11.70+)

Pero sigue siendo mejor el uso de índices compuestos específicos.

Mejores prácticas para índices. (Cont)

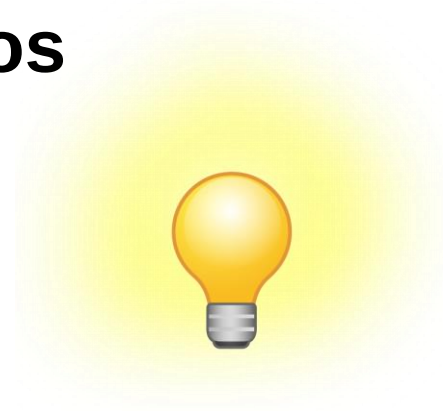
En el caso de índices compuestos (OLTP):

- Revise la cardinalidad (unicidad) de las columnas.
- Mientras más selectivo sea el índice en el primer campo, mejor.

¡No es lo mismo el índice AB, que BA!

JOINs

- Respete la integridad referencial.
- En ambientes OLTP asegúrese de tener **índices** sobre las columnas de **JOIN**.
- No use **OUTER JOINs** a menos de que sea **indispensable**.
- Limite la búsqueda a los datos **mínimos indispensables**.



UNIONs

- Siempre que pueda, prefiera el uso de **UNION ALL** en lugar de **UNION**.
- No abuse de los UNION, si el número de UNIONs es elevado considere el uso de tablas temporales.
- Limite la búsqueda a los datos **mínimos indispensables**.

SUBQUERIES.

- Varios subqueries, se pueden convertir en JOINs, siempre que pueda prefiera el uso de JOINs a los subqueries.
- Evite el exceso de subqueries, si necesita hacer varios subqueries, considere el uso de tablas temporales… ¡Sin abusar!
- Solo use subqueries cuando sea estrictamente necesario.
- Una vez más: Limite la búsqueda a los datos **mínimos indispensables**.

Otras consideraciones.

- Típicamente un query puede mejorarse si se restringe el universo de búsqueda a través de la cláusula WHERE.
- Evite el uso de substrings en la cláusula WHERE (ie: `lname[1,2]="Lo"`), eso provoca que no se use índices.
- Evite el uso de funciones en la cláusula WHERE (ie: UPPER(`lname`)="`LORANCA`"). Si es indispensable, considere el uso de INDICES FUNCIONALES.
- Trate de agrupar los JOINS para que involucren la menor cantidad de tablas.
- Siempre use los mismos tipos de datos en comparaciones de la cláusula WHERE.

Otras consideraciones. (Cont)

- Algunos queries con opción “OR” en la cláusula WHERE, pueden ser reescritos como dos o más queries parciales unidos con el operador UNION o UNION ALL.
- Evite abusar de cláusulas del tipo “`WHERE X IN [a,b,c,d,…]`”, sin son muchas, procure intentar un criterio de selección diferente, o poblar una tabla temporal y hacer JOIN sobre ella.
- No abuse de los ORDER BY, el ordenamiento consume memoria y tiempo.
- Si **no** va a utilizar algún **agregado**, **no** use **GROUP BY** para eliminar duplicados, considere en su lugar el uso del modificador DISTINTICT de la cláusula SELECT.

SQL TRACE.

- Puede capturar y reportar el comportamiento de los últimos n queries.
- **No está diseñado para fines de auditoría.**
- Cuando llegar al query “n+1”, este sobreescribe al “1”.
- Puede ser configurado por usuario y/o base de datos.
- Permite registrar tiempos de ejecución, y si hubo esperas por I/O o candados.
- Puede llegar a consumir mucho espacio de **disco** (sysadmin).
- Puede agregar **overhead** a la instancia.
- Se recomienda que sólo sea activado y configurado por el DBA.

SQL TRACE.

SQL Profile					
Session ID	**User ID**	**Statement Type**	**PDQ**	**Statement Completion Time**	**Response Time**
85	200	CREATE TABLE	0	2008-06-02 03:37:41	0.0000000 Sec
Database	bharath				
Statement	"create table sample(name varchar(20), id int) "				

Statement Statistics							
Page Reads	**Buffer Reads**	**Reads Cache**	**Data Buffer Reads**	**Index Buffer Reads**	**Page Writes**	**Buffer Writes**	**Writes Cache**
0	41	100.00 %	41	0	0	24	100.00 %
Lock Requests	**# Lock Waits**	**Lock Wait Time (S)**	**Log Space**	**Disk Sorts**	**Memory Sorts**	**Number of Tables**	**Number of Iterators**
31	0	0	1.79 KB	0	0	2	0
Total Executions	**Total Executions Time (S)**	**Average Execution Time (S)**	**Maximum Execution Time (S)**	**Number of IO Wait**	**IO Wait Time (S)**	**Average IO Wait (S))**	**Average Rows/Second**
1	0.00000	0.00000	0.00000	0	0.00000	0.00000	1248834037.16387
Estimated Cost	**Estimated Rows**	**Actual Rows**	**SQL Error**	**ISAM Error**	**Isolation Level**	**SQL Memory**	
0	0	0	0	0	2	3.81 KB	

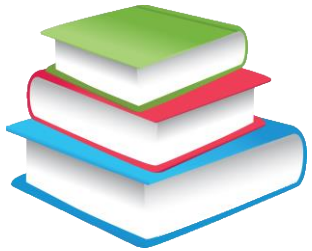
Para saber más.

Tuning Informix SQL (Part 1)
https://www.ibm.com/developerworks/data/library/techarticle/dm-0409fan/index.html

Tuning Informix SQL (Part 2)
https://www.ibm.com/developerworks/data/library/techarticle/dm-0410fan/index.html

Create and use functional indexes in Informix Dynamic Server
https://www.ibm.com/developerworks/data/library/techarticle/dm-0712wilcox/index.html

Performance analysis of Informix Dynamic Server made easy through the OpenAdmin Tool
http://www.ibm.com/developerworks/data/library/techarticle/dm-0807sudha/



¿Preguntas?

¡Gracias!