IBM Software Group

Informix Performance Tuning

SQL Explain & SQL Tracing

Informix software

Information Management software

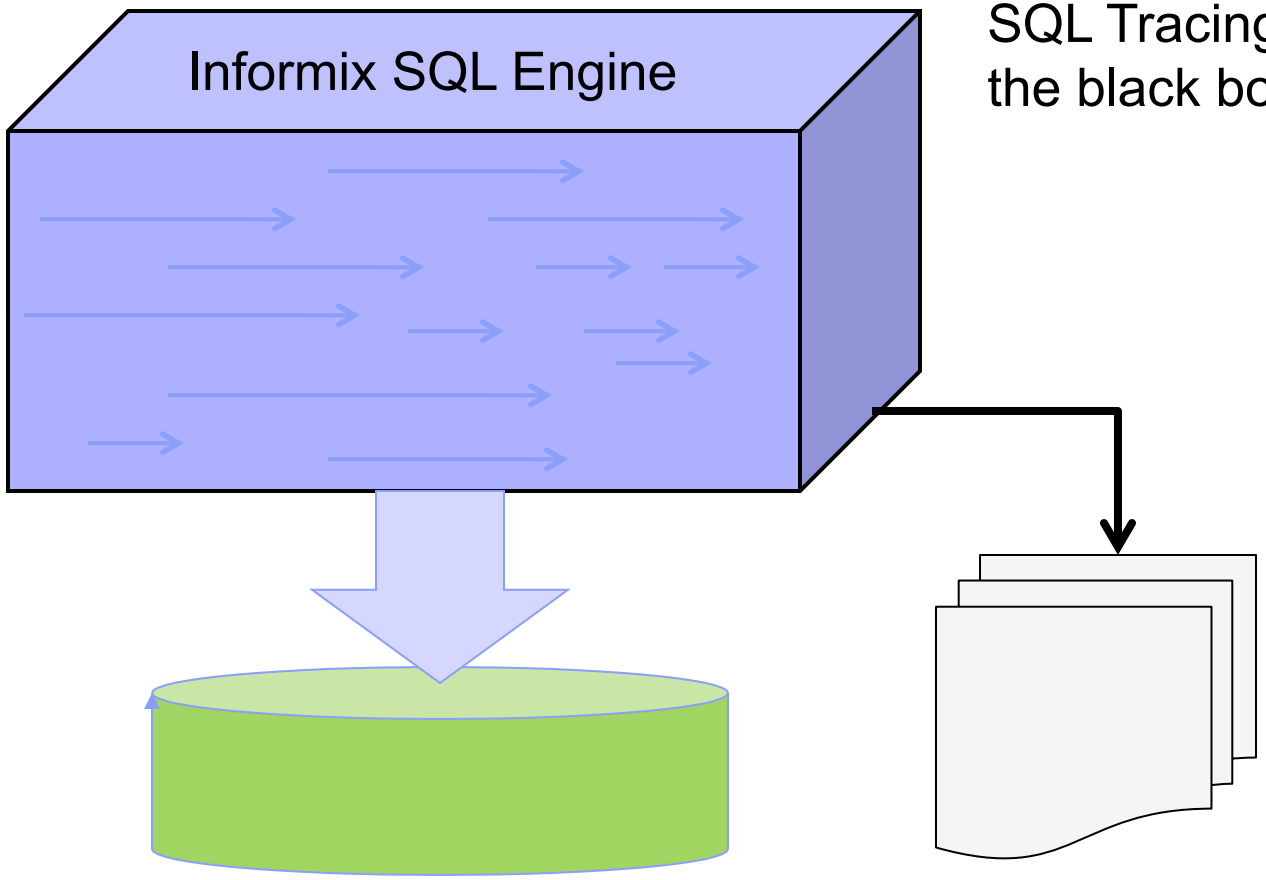
譚永貽 中國開發試驗室

Informix Performance Tuning

- Informix調優的方法列舉如下
  - Informix參數調優
  - Informix Update Statistics
  - Informix Statement cache
  - Fragmentation and Indexing
  - Optimizer SQL Directives
  - Informix 4K, 8K, 10K可調整的頁
  - Informix Parallel Data Query(PDQ)
  - Informix 11 – Repack, Shrink, and cleanup
  - Informix 11 – data compression
  - Informix 11.5 – SQL Tracing

Informix 11.5 SQL跟蹤

SQL Tracing
SQL Tracing opens the black box.



Measurement Tool – dbaccess and time

```
% time dbaccess store_demo sel_join.sql
Database selected.
(count(*)) 5958 1 row(s) retrieved.
Database closed.

real 0m0.09s
user 0m0.01s
sys 0m0.06s
```

SET EXPLAIN – Measurement tool

執行一個**SQL**我們需要消耗多少資源?
我們怎樣估算和衡量**SQL**的性能?

```
SET EXPLAIN ONAVOID_EXECUTE;
    SELECT C.CUSTOMER_NUM, C.LNAME, C.FNAME, C.PHONE,
    O.ORDER_DATE
    FROM CUSTOMER C, ORDERS O
    WHERE C.CUSTOMER_NUM = O.CUSTOMER_NUM
    AND C.LNAME = 'Watson'
SET EXPLAIN OFF;
```

SET EXPLAIN - sqexplain.out (I)

```
QUERY: (OPTIMIZATION TIMESTAMP: 12-11-2009 20:28:42)
------
SELECT C.CUSTOMER_NUM, C.LNAME, C.FNAME, C.PHONE, O.ORDER_DATE
FROM CUSTOMER C, ORDERS O
WHERE C.CUSTOMER_NUM = O.CUSTOMER_NUM
AND C.LNAME = 'Watson'
Estimated Cost: 1014294
Estimated # of Rows Returned: 4
  1) informix.c: SEQUENTIAL SCAN  (Serial, fragments: ALL)
        Filters: informix.c.lname = 'Watson'
  2) informix.o: INDEX PATH
    (1) Index Name: informix. 112_23
        Index Keys: customer_num   (Serial, fragments: ALL)
        Lower Index Filter: informix.c.customer_num = informix.o.customer_num
NESTED LOOP JOIN
```

SET EXPLAIN - sqexplain.out (II)

```
Internal name     Table name
---------------------------
t1                c
t2                o

type     table  rows_prod  est_rows  rows_scan  time       est_cost
-------------------------------------------------------------------
scan     t1     1          1         10000041   01:44.91   1014293

type     table  rows_prod  est_rows  rows_scan  time       est_cost
-------------------------------------------------------------------
scan     t2     14         50000000  14         00:00.28   1

type     rows_prod  est_rows  time       est_cost
-----------------------------------------------
nljoin   14         5         01:45.19   1014294
```

SET EXPLAIN

在SET EXPLAIN語句執行：

- 顯示查詢執行計畫（或查詢計畫）
- 顯示查詢執行計畫所產生的成本為基礎的優化
- 估計行數返回
- 估計查詢的相對成本
- 預設情況下生成的輸出檔案名，是“sqexplain.out”

Set Explain Sample

- SET EXPLAIN ON;
- select * from customer
- where zipcode = '94063';
- SET EXPLAIN OFF;

1. Sequential Scan without index
2. Index Scan after creating index

```
CREATE INDEX zip_idx ON customer(zipcode);
```

Sequential Scan without index

```
QUERY: (OPTIMIZATION TIMESTAMP: 12-10-2009 14:02:23)
------
select * from customer
where zipcode = '94063'

Estimated Cost: 1014289
Estimated # of Rows Returned: 100

  1) informix.customer:SEQUENTIAL SCAN(Serial, fragments: ALL)
        Filters: informix.customer.zipcode = '94063'

Query statistics:
…….
  Internal name     Table name
  --------------------------
  t1                customer

  type     table  rows_prod  est_rows   rows_scan  time        est_cost
  ------------------------------------------------------------------
  scan     t1     99          10010000041   00:14.071014290
```

Index search with index

```
QUERY: (OPTIMIZATION TIMESTAMP: 12-10-2009 15:09:21)
------
select * from customer
where zipcode = '94063'

Estimated Cost: 28
Estimated # of Rows Returned: 25

  1) informix.customer:INDEX PATH

    (1) Index Name: informix.zip_idx
        Index Keys: zipcode   (Serial, fragments: ALL)
        Lower Index Filter: informix.customer.zipcode = '94063'

Query statistics:
Internal name     Table name
  ---------------------------
  t1                customer

  type     table  rows_prod  est_rows  rows_scan  time        est_cost
  ------------------------------------------------------------------
  scan     t1     99         2599        00:00.0028
```

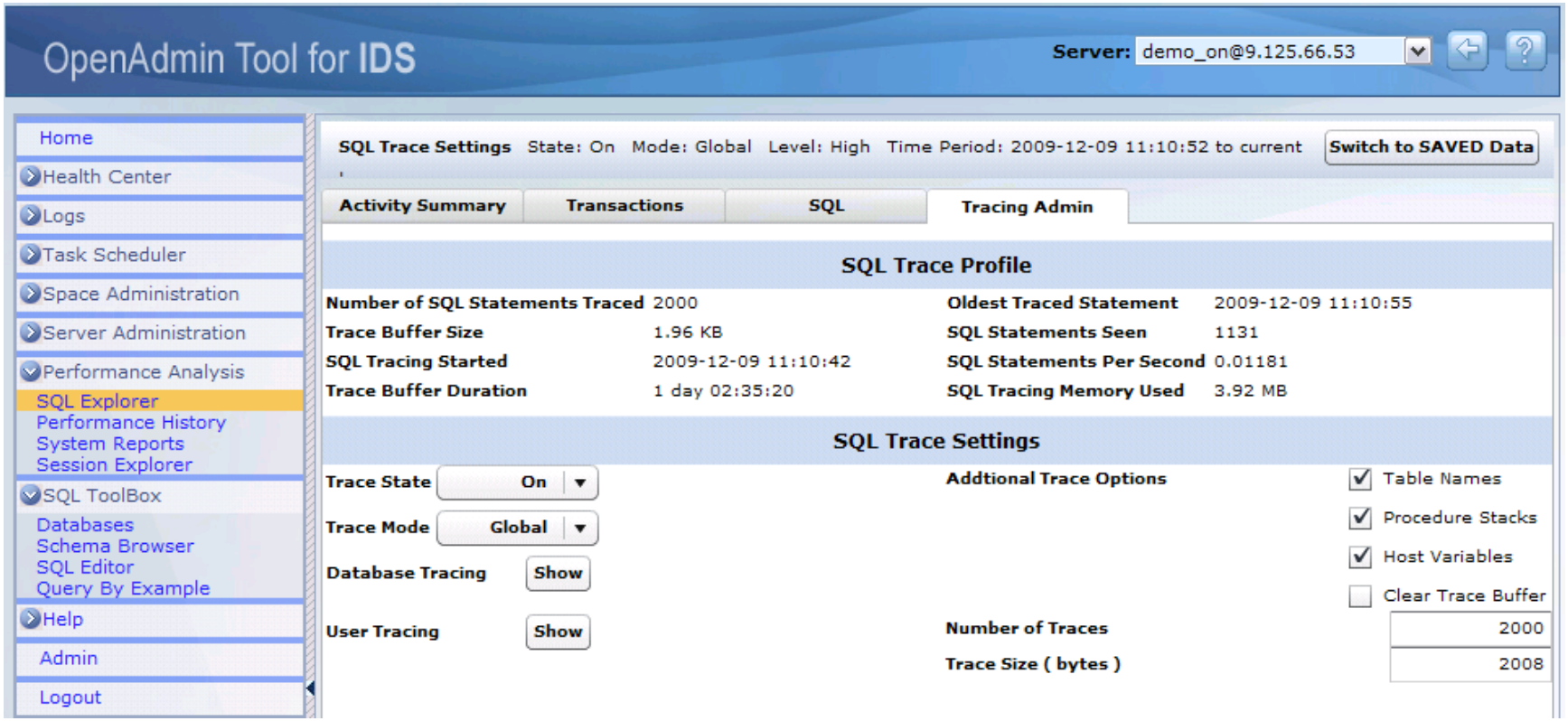
“預計執行時間”最長的SQL語句

- 找到“預計執行時間”最長的SQL語句
  - 執行SET EXPLAIN ON後，過一段時間後執行如下SQL語句
  - select first 25 sqx_estcost,sqx_estrows,sqx_sqlstatement
    from sysmaster:syssqexplain
    where sqx_sqlstatement not like '%first%'
    order by sqx_estcost desc

Performance Tuning Topics

- SET EXPLAIN output
- SQL Tracing

OpenAdmin Tool :跟蹤SQL語句



跟蹤SQL語句

- SQL跟蹤
- 通過配置SQL語句跟蹤參數監控最近執行的SQL語句的性能
- 提供系統裡執行的每一條SQL語句的統計資訊
- 統計資訊存儲在可配置的環形緩衝區裡
- 缺省情況下，這個特性是關閉的
- 可以有選擇的將某些用戶設置為使用這個特性
- 可更方便的分析SQL語句從而更好的進行性能調優

性能調優:跟蹤SQL (I)

- Onconfig參數: SQLTRACE
  - level = [off, low, med, high]
  - ntraces = [跟蹤的SQL語句的數量]
  - size = [每個跟蹤緩衝區的大小，以KB為單位]
  - mode = [global, user]

```
level=low, ntraces=1000,size=2,mode= global
```

- 關閉和打開SQL跟蹤

```
EXECUTE FUNCTION task(' SET SQL TRACING OFF');
EXECUTE FUNCTION task(' SET SQL TRACING ON');
```

性能調優:跟蹤SQL (II)

- IDS 11.5C3對指定資料庫進行跟蹤
  可指定對某一個或某幾個資料庫進行跟蹤。缺省情況下是跟蹤所有的資料庫。要跟蹤的資料庫的數目最多可以是16。

```
EXECUTE FUNCTION task(' SET SQL TRACING DATABASE ADD');
```

- 添加、清除所有、顯示所有、刪除要跟蹤SQL語句的資料庫

1. "SET SQL TRACING DATABASE ADD" , *"database_name ")*
2. "SET SQL TRACING DATABASE CLEAR"
3. "SET SQL TRACING DATABASE LIST"
4. "SET SQL TRACING DATABASE REMOVE" , *"database_name"*

性能調優:跟蹤SQL (III)

- 用於SQL跟蹤的Sysmaster表
  - syssqltrace:存儲與SQL語句的統計資料有關的資訊
  - syssqltrace_info:存儲與SQL跟蹤的配置有關的資訊
  - syssqltrace_iter:存儲與反覆運算器(iterator)和explain輸出有關的資訊

Example : Seek the of queries that ran > 2 seconds

```
database sysmaster;
select count(*) from syssqltrace  WHERE sql_totaltime > 2;
```

Example : SQL trace information for a particular statement

```
database sysmaster;
select * from syssqltrace a, syssqltrace_iter b
where a.sql_id =b.sql_id and a.sql_id=329;
```

跟蹤的方法(續)

- 用於SQL跟蹤的Sysmaster表
  - syssqltrace:存儲與SQL語句的統計資料有關的資訊
  - syssqltrace_info:存儲與SQL跟蹤的配置有關的資訊
  - syssqltrace_iter:存儲與反覆運算器(iterator)和explain輸出有關的資訊

Example : Seek the of queries that ran > 2 seconds

```
database sysmaster;
select count(*) from syssqltrace  WHERE sql_totaltime > 2;
```

Example : SQL trace information for a particular statement

```
database sysmaster;
select * from syssqltrace a, syssqltrace_iter b
where a.sql_id =b.sql_id and a.sql_id=329;
```

“實際執行時間”最長的SQL語句

- 找到“實際執行時間”最長的SQL語句，並進行調優

  - 打開SQL tracing，過一段時間後執行如下SQL語句
  - select first 20 sql_id from sysmaster:syssqltrace
    where sql_runtime > 0
    order by sql_runtime desc;

  - 使用上面的SQL語句得到一些sql_id後，再通過下面的SQL語句得到這些sql_id對應的SQL語句的資訊（將下面SQL語句中的my_sql_id替換成這些sql_id)
  - SELECT sql_sid, sql_uid,
    dbinfo('UTC_TO_DATETIME',sql_finishtime) as finishtime,
    TRUNC(sql_runtime,7) || ' Sec' AS sql_runtime,
    sql_actualrows, sql_statement, sql_database
    FROMsysmaster:syssqltrace WHERE sql_id =my_sql_id

跟蹤的方法(續)

- Onconfig參數: SQLTRACE

```
level=low, ntraces=1000,size=2,mode= global
```

資料庫實例級別的設置，由於Tracing會消耗大量資源，不建議使用這種方式。一般使用原則是：需要時打開，不需要時要確認關閉:

```
execute function sysadmin:task('set sql tracing info')
        (expression)  SQL TracingOFF.
        1 row(s) retrieved.
```

跟蹤的方法(續)

- level可以是LOW，MED，HIGH，
  - LOW包括：statement statistics, statement text, and statement iterators.
  - MED包括：all of the information included in low-level tracing, plus table names, the database name, and stored procedure stacks
  - HIGH包括：all of the information included in lower level tracing, plus parameter values
- ntraces表示追蹤的最多SQL條數
- size是指追蹤每條SQL使用的記憶體大小，以K為單位
- Mode可以是global或者user
  - global表示對所有用戶打開SQL追蹤功能，此模式下所有SQL都被追蹤
  - user表示以user模式打開SQL追蹤，此模式下只有打開SQL追蹤的session下的SQL才被追蹤

跟蹤的方法(續)

- 以global模式打開Tracing：

```
EXECUTE FUNCTION task("set sql tracing on", 1000, 1,"high","global");
EXECUTE FUNCTION task("set sql tracing on", "global");
```

此模式下，所有使用者的所有SQL都被追蹤

- 以user模式打開Tracing：

```
EXECUTE FUNCTION task("set sql tracing on", 1000, 1,"high",'user');
EXECUTE FUNCTION task("set sql tracing user add", "informix");
```

此模式下，只有添加到追蹤列表中用戶的SQL才被追蹤

需要注意的地方(續)

- SQLTracing會影響系統性能，必要時才打開
- syssqltrace表中的sql_runtime/sql_totaltime的單位為秒
- 對於使用某种AMDprocessors的機器，可能會出現特別大的值，原因在於Informix使用了" fast light weight clock"的機制計算時間，這種機制效率高，但是這種cpu架構不保證該時鐘的同步
- 對於sql_runtime/sql_totaltime列的值，如果是多執行緒的查詢，會比實際花費時間要長，原因是，該值為把多個執行緒的執行時間都做了加和，性能調優時，可以對比調整前後的值，從應用角度看到的時間，只在單執行緒查詢時才是一致的。

實例1

- database sysmaster;
- execute function sysadmin:task('set sql tracing on', 100,2,'high','global');
- database demodb;
- select * from customer c, orders o, cust_calls cc where c.customer_num = o.customer_num and cc.customer_num=c.customer_num and c.lname not like '%a' ;
- database sysmaster;
- select * from sysmaster:syssqltrace where sql_id=3;

Result :

sql_id 11

sql_address 1275031644

sql_sid 41

sql_uid 503

sql_stmttype 2

sql_stmtname SELECT

sql_totaltime

實例1(續)

sql_runtime 0.007486995934
sql_pgreads 5
sql_bfreads 109
sql_rdcache 95.41284403670
sql_bfidxreads 0
sql_pgwrites 0
sql_bfwrites 0
sql_wrcache 0.00
sql_lockreq 81
sql_lockwaits 0
sql_lockwttime 0.00
sql_logspace 0
sql_sorttotal 0
sql_sortdisk 0
sql_sortmem 0
sql_executions 1
sql_totaltime 0.019677553892
sql_avgtime 0.019677553892
sql_maxtime 0.007486995934
sql_numiowaits 3
…… ……

sql_rowspersec 1202.084264377
sql_estcost 11
sql_estrows 11
sql_actualrows 9
sql_sqlerror 0
sql_isamerror 0
sql_isollevel 2
sql_sqlmemory 23104
sql_numiterators 5
sql_database <None>
sql_numtables 0
sql_tablelist None
sql_statement select * from customer c, orders o, cust_calls cc where c.customer_num = o.customer_num and cc.customer_num=c.customer_num
and c.lname not like '%a'
sql_stmtlen 148
sql_stmthash 755685204
sql_pdq 0
sql_num_hvars 0
sql_dbspartnum 1048969

實例1(續)

命令：onstat –g his

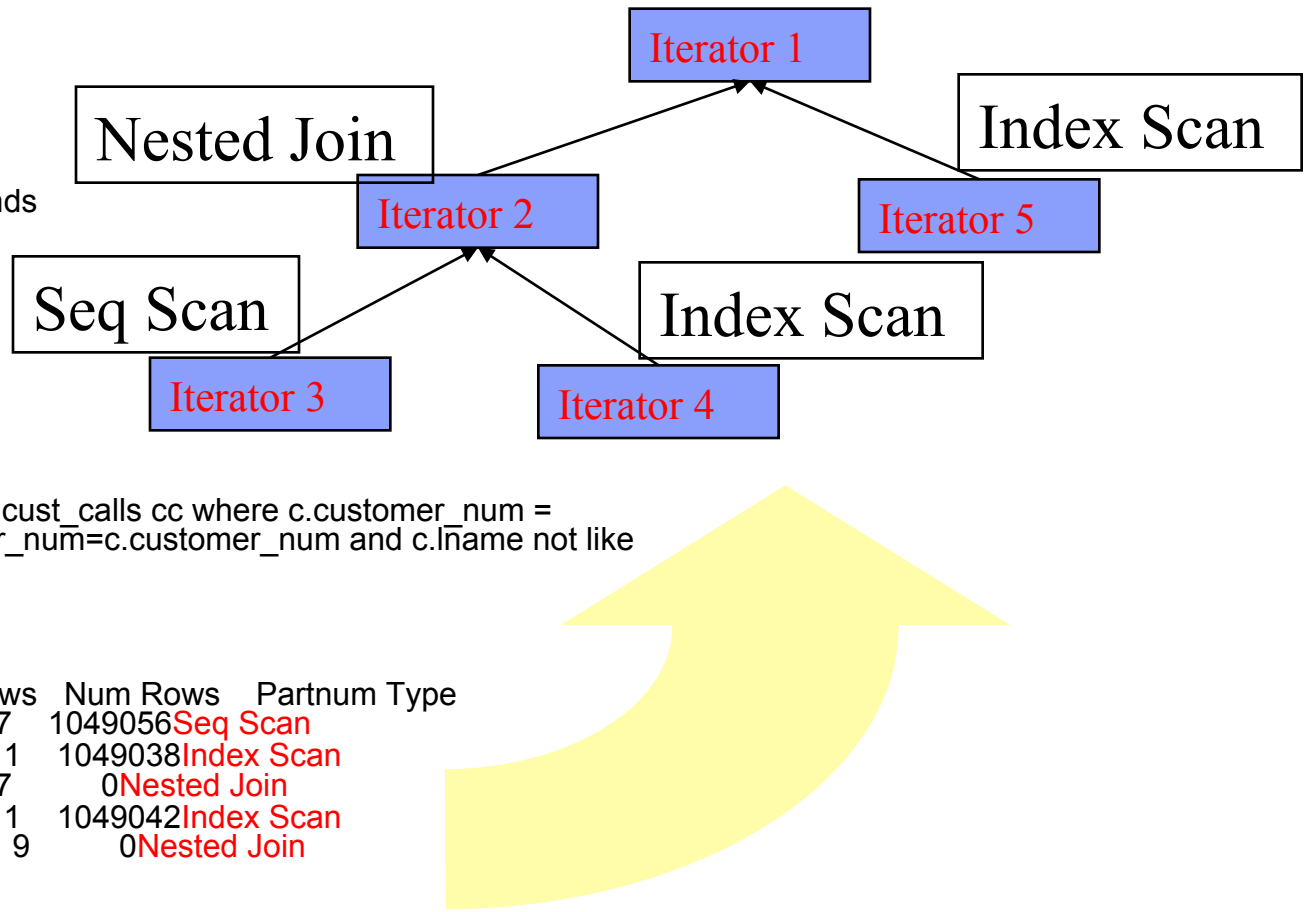
```
Statement history:

Trace Level              Low
Trace Mode               Global
Number of traces         1000
Current Stmt ID          11
Trace Buffer size        1000
Duration of buffer       2455 Seconds
Trace Flags              0x00001611
Control Block            0x4bff7018


Statement # 11:     @ 0x4bff705c

 Database:       0x100189
Statement text:
 select * from customer c, orders o, cust_calls cc where c.customer_num =
   o.customer_num and cc.customer_num=c.customer_num and c.lname not like
   '%a'

 Iterator/Explain
 ================
   ID   Left  Right   Est Cost   Est Rows   Num Rows    Partnum Type
    3      0      0          4          7          7    1049056Seq Scan
    4      0      0          1         22          1    1049038Index Scan
    2      3      4          8          5          7          0Nested Join
    5      0      0          1         23          1    1049042Index Scan
    1      2      5         11         11          9          0Nested Join
```



從這裡開始

實例1(續)

Statement information:

Sess_id	User_id	Stmt Type	Finish Time	RunTime	TX Stamp	PDQ
41	503	SELECT	18:25:03	0.0075	3b0fe4	0

Statement Statistics:

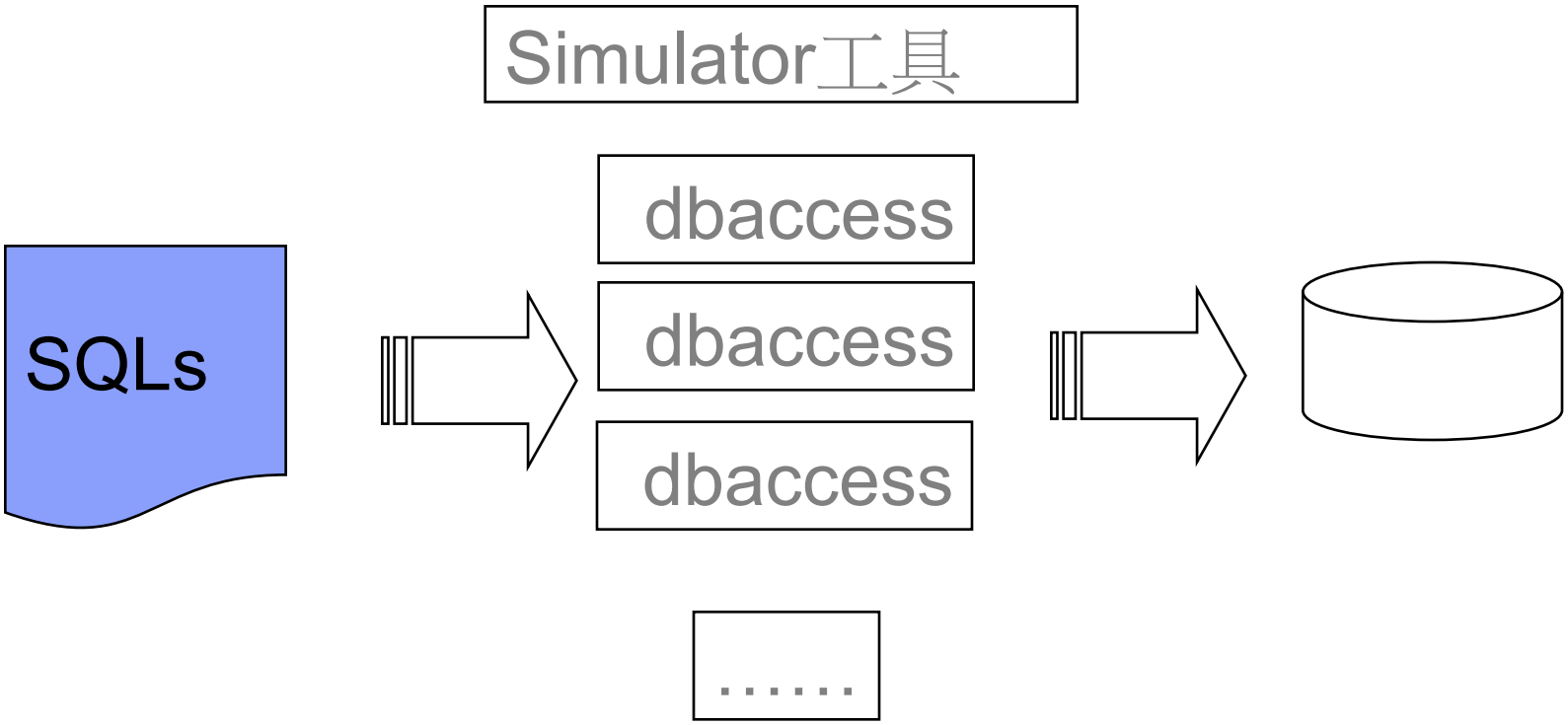
Page Read	Buffer Read	Read % Cache	Buffer IDX Read	Page Write	Buffer Write	Write % Cache
5	109	95.41	0	0	0	0.00

Lock Requests	Lock Waits	LK Wait Time (S)	Log Space	Num Sorts	Disk Sorts	Memory Sorts
81	0	0.0000	0.000 B	0	0	0

Total Executions	Total Time (S)	Avg Time (S)	Max Time (S)	Avg IO Wait	I/O Wait Time (S)	Avg Rows Per Sec
1	0.0197	0.0197	0.0075	0.0018	0.0055	1202.0832

Estimated Cost	Estimated Rows	Actual Rows	SQL Error	ISAM Error	Isolation Level	SQL Memory
11	11	9	0	0	CR	23104

實例2



實例2(續)

- 打開SQL Tracing:
  - execute function sysadmin:task('set sql tracing on', 100,2,'high','global');
- 運行Simulator，執行SQL
  - ./runsqls bankpri demodb 1 10 sqlfiles
- 分析執行過的SQL：
  - 耗時最多的
    - select first 10 * from syssqltrace order by sql_runtime desc
  - IO等待最多的
    - select first 10 * from syssqltrace order by sql_totaliowaits desc
  - LOCK最多的
    - select first 10 * from syssqltrace order by sql_lockreq desc