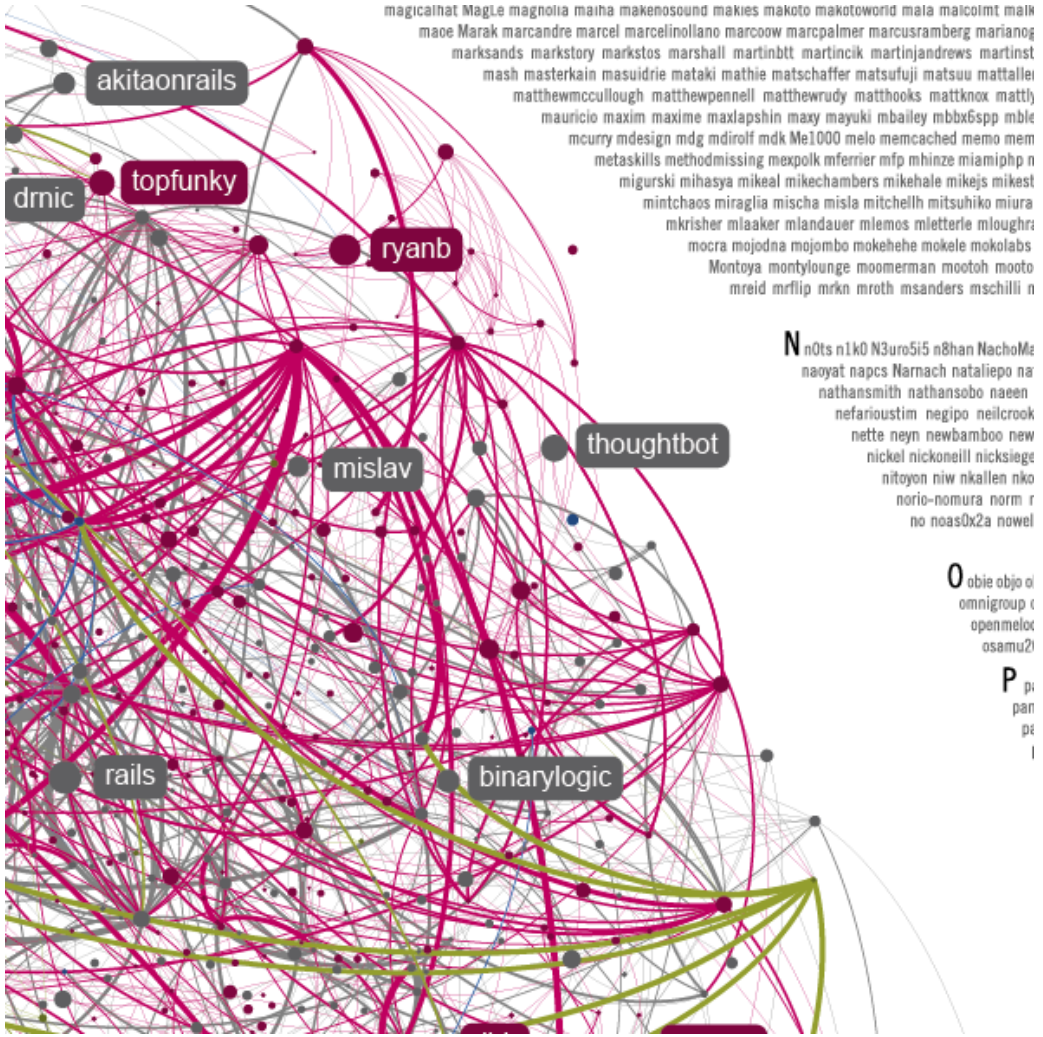
DB2设计与性能优化

第9周



提纲

- 监控找出问题SQL
- 获取SQL的访问计划
- 解读和分析访问计划
- 调优SQL语句的招式
- 小结与练习

系统性能下降：你的选择？

从监控开始：找出问题SQL

- 揪出最消耗资源的那些SQL！
- 快照监控?
- 管理视图？
- 事件监控器？
- db2pd?
- 用你擅长的工具…

高级DBA

快照：动态SQL

```
                Dynamic SQL Snapshot Result
Number of executions                = 81341
Number of compilations              = 0
Worst preparation time (ms)         = 12
Best preparation time (ms)          = 12
Internal rows deleted               = 0
Internal rows inserted              = 0
Rows read                           = 81336
Internal rows updated               = 0
Rows written                        = 81336
Statement sorts                     = 0
Statement sort overflows            = 0
Total sort time                     = 0
Buffer pool data logical reads      = 162672
Buffer pool data physical reads     = 0
Buffer pool temporary data logical reads    = 0
Buffer pool temporary data physical reads   = 0
Buffer pool index logical reads     = 162732
Buffer pool index physical reads    = 0
Buffer pool temporary index logical reads   = 0
Buffer pool temporary index physical reads  = 0
Buffer pool xda logical reads       = 0
Buffer pool xda physical reads      = 0
Buffer pool temporary xda logical reads     = 0
Buffer pool temporary xda physical reads    = 0
Total execution time (sec.ms)       = 18.542262
Total user cpu time (sec.ms)        = 7.900000
Total system cpu time (sec.ms)      = 0.470000
Total statistic fabrication time (milliseconds) = 0
Total synchronous runstats time  (milliseconds) = 0
Statement text                      = Update DISTRICT …
```

语句编译:
- 含义: 执行前需要重新编译的次数
- 计算: (number of compilations) / (number of executions)
- 良好: 0，或者接近于0；调优措施：PCKCACHESZ太小?

读取行数:
- 含义:
  - 读的行数，不是选择的行数
  - 在index-only access时可能为0
  - OLTP：如果远大于执行次数，可能说明索引有问题

语句的缓冲池命中率:
- 含义:
  - 含义跟BP或者DB层次的指标一样
  - 定位该SQL是否导致缓冲池问题的最佳指标

总时间:
- 含义: -该语句在数据执行的总时间

■对大多数语句来说应该关注的指标
- 执行次数，读写行数，执行时间, 排序等
- 关注和调优这些指标最大的语句，达到事半功倍的效果

快照：排序

```
                Database Snapshot
:
Total Private Sort heap allocated          = 2058
Total Shared Sort heap allocated           = 0
Shared Sort heap high water mark           = 0
Total sorts                                = 24
Total sort time (ms)                       = 866719
Sort overflows                             = 15
Active sorts                               = 3
:
```

```
    Database Manager Snapshot
:
Private Sort heap allocated                = 2058
Private Sort heap high water mark          = 4101
Post threshold sorts                       = 0
Piped sorts requested                      = 9
Piped sorts accepted                       = 9
:
```

排序溢出:
- 含义: 排序溢出写到磁盘的比率
- 计算: (sort overflows) / (total sorts)
- 良好: 对OLTP负载，0或接近于0

平均排序时间:
- 含义: 排序的平均运行时间
- 计算: (total sort time) / (total sorts)
- 良好: 远远小于语句预期的响应时间

排序内存超出:
- 含义: 排序堆高水位大于SHEAPTHRES值
  - 引发降低私有排序堆分配或可能溢出到磁盘

- 排序溢出的代价很高
  - 需要在调优时平衡SORTHEAP和bufferpool的分配
  - 如果排序总时间不大，那么不必太在意排序溢出
- 排序的监控数据也在“动态SQL快照”中体现
  - 该SQL语句的排序是否可以通过调优消除

找出问题SQL：管理视图

■获取当前执行成本最高的SQL

```
SELECT APPLICATION_HANDLE, ELAPSED_TIME_SEC, QUERY_COST_ESTIMATE ,
ROWS_RETURNED, ROWS_READ  FROM SYSIBMADM.MON_CURRENT_SQL
```

■获取运行最长的SQL

```
SELECT SUBSTR(APPL_NAME,1,15) AS APPL_NAME, ELAPSED_TIME_MIN AS "Elapsed Min.",
APPL_STATUS AS "Status ", SUBSTR(AUTHID,1,10) AS AUTH_ID ,
SUBSTR(INBOUND_COMM_ADDRESS,1,15) AS "IP Address“, SUBSTR(STMT_TEXT,1,30)  AS
"SQL Statement"  FROM SYSIBMADM.LONG_RUNNING_SQL ORDER BY 2 DESC
```

■获取执行次数最多的SQL

```
SELECT NUM_EXECUTIONS AS "执行次数", AVERAGE_EXECUTION_TIME_S AS "平均时间秒",
STMT_SORTS AS "排序次数", SORTS_PER_EXECUTION AS "每语句排序", SUBSTR(STMT_TEXT,
1, 30) AS "执行语句"  FROM SYSIBMADM.TOP_DYNAMIC_SQL  WHERE NUM_EXECUTIONS > 0
ORDER BY 2 DESC FETCH FIRST 5 ROWS ONLY
```

■获取排序次数最多的SQL

```
SELECT STMT_SORTS, SORTS_PER_EXECUTION, SUBSTR(STMT_TEXT, 1, 80) AS
STMT_TEXT FROM SYSIBMADM.TOP_DYNAMIC_SQL ORDER BY STMT_SORTS DESC FETCH
FIRST 5 ROWS ONLY
```

找出问题SQL：语句事件监控

■执行耗时最长的SQL语句

```
SELECT STMT_TEXT,(STOP_TIME-START_TIME) as “执行时间秒” FROM
STMT_SQL_TRACE WHERE STMT_OPERATION NOT IN (7,8,9,19) ORDER BY
DECIMAL(“执行时间秒”) DESC FETCH FIRST 10 ROWS ONLY
```

■执行次数最多的SQL

```
SELECT DISTINCT (STMT_TEXT), COUNT(1) AS “次数”  FROM STMT_SQL_TRACE
WHERE STMT_OPERATION NOT IN (7, 8, 9, 19)  GROUP BY STMT_TEXT  ORDER BY COUNT(1)
DESC  FETCH FIRST 10 ROWS ONLY
```

■最耗CPU时间的SQL语句

```
SELECT STMT_TEXT, USER_CPU_TIME AS "用户CPU秒" FROM
STMT_SQL_TRACE  WHERE STMT_OPERATION NOT IN (7, 8, 9, 19)
ORDER BY "用户CPU秒" DESC FETCH FIRST 10 ROWS ONLY
```

■总排序时间最长的SQL语句

```
SELECT STMT_TEXT,TOTAL_SORT_TIME “排序时间秒” FROM
STMT_SQL_TRACE WHERE STMT_OPERATION NOT IN (7,8,9,19) ORDER
BY DECIMAL(TOTAL_SORT_TIME) DESC FETCH FIRST 10 ROWS ONLY
```

提纲

- 监控找出问题SQL
- 获取SQL的访问计划
- 解读和分析访问计划
- 调优SQL语句的招式
- 小结与练习

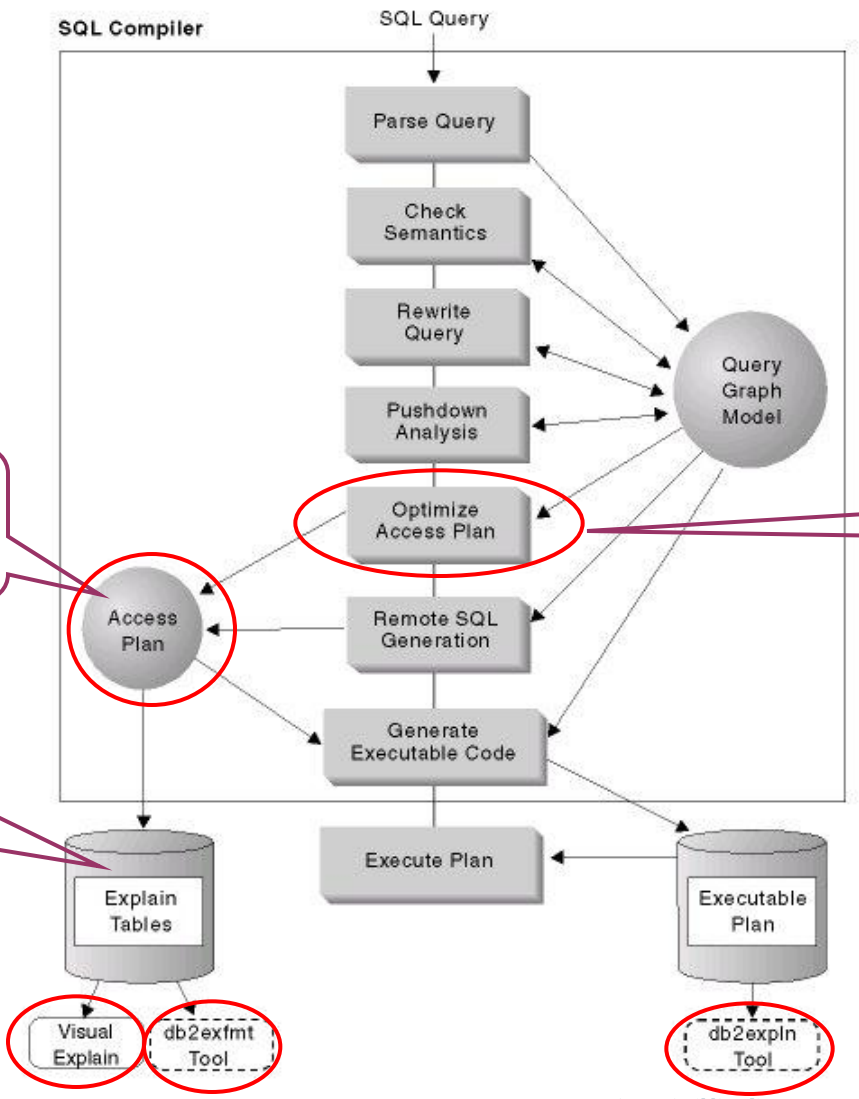
SQL语句查询计划说明工具

- 什么是SQL语句的访问计划(Access Plan)
  - SQL语句编译和优化后生成的内部操作执行顺序
  - DB2引擎依照此计划访问表数据并执行相关操作
- 如何获取查询计划？ 三个工具：
  - Visual Explain：图形化说明工具
  - db2expln：快速便捷工具
  - db2exfmt：终极说明工具

我想看看SQL语句在DB2后台到底是如何执行的？

骨灰级DBA

SQL编译

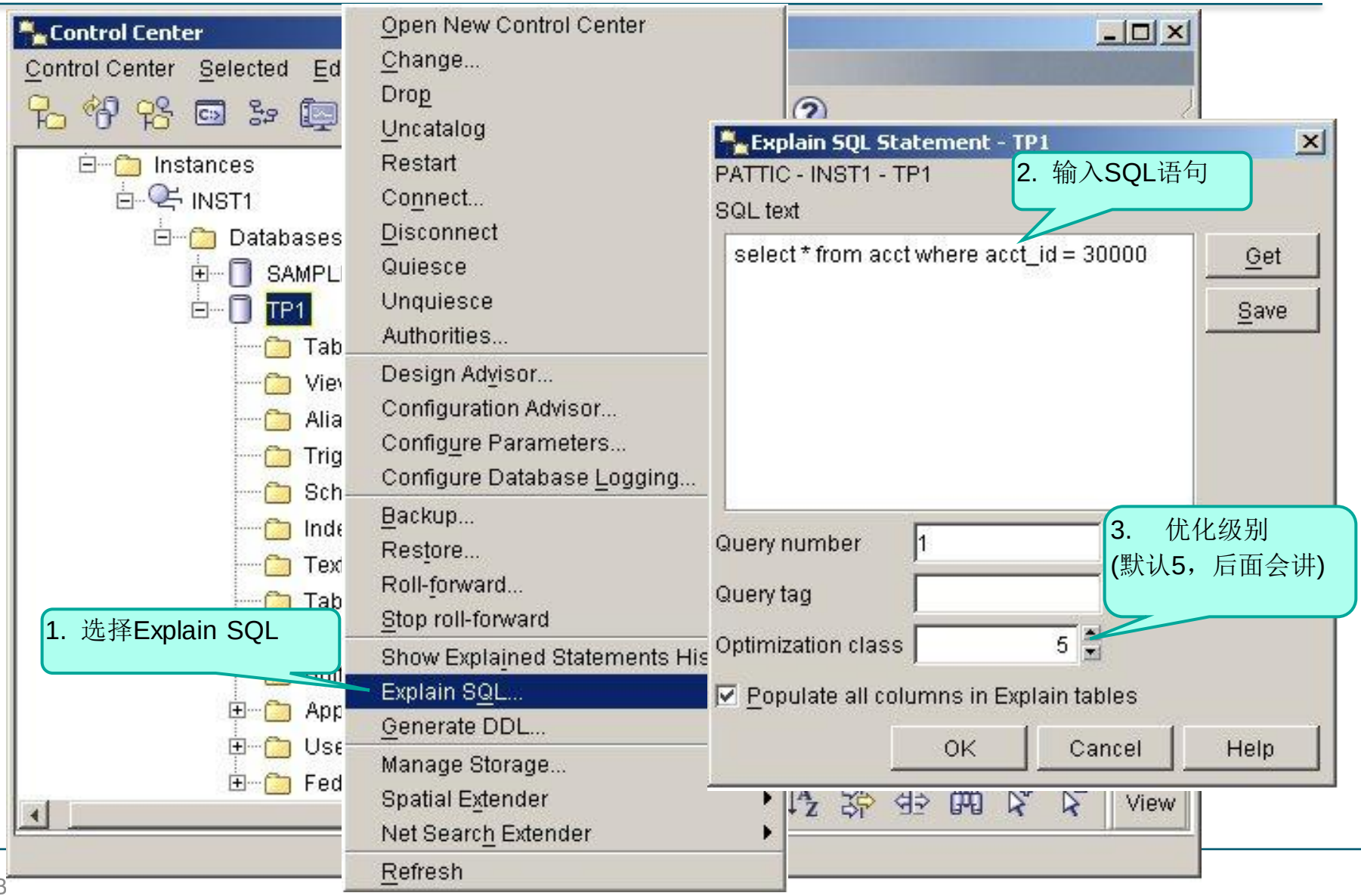


访问计划：
一切奥秘就在其中！

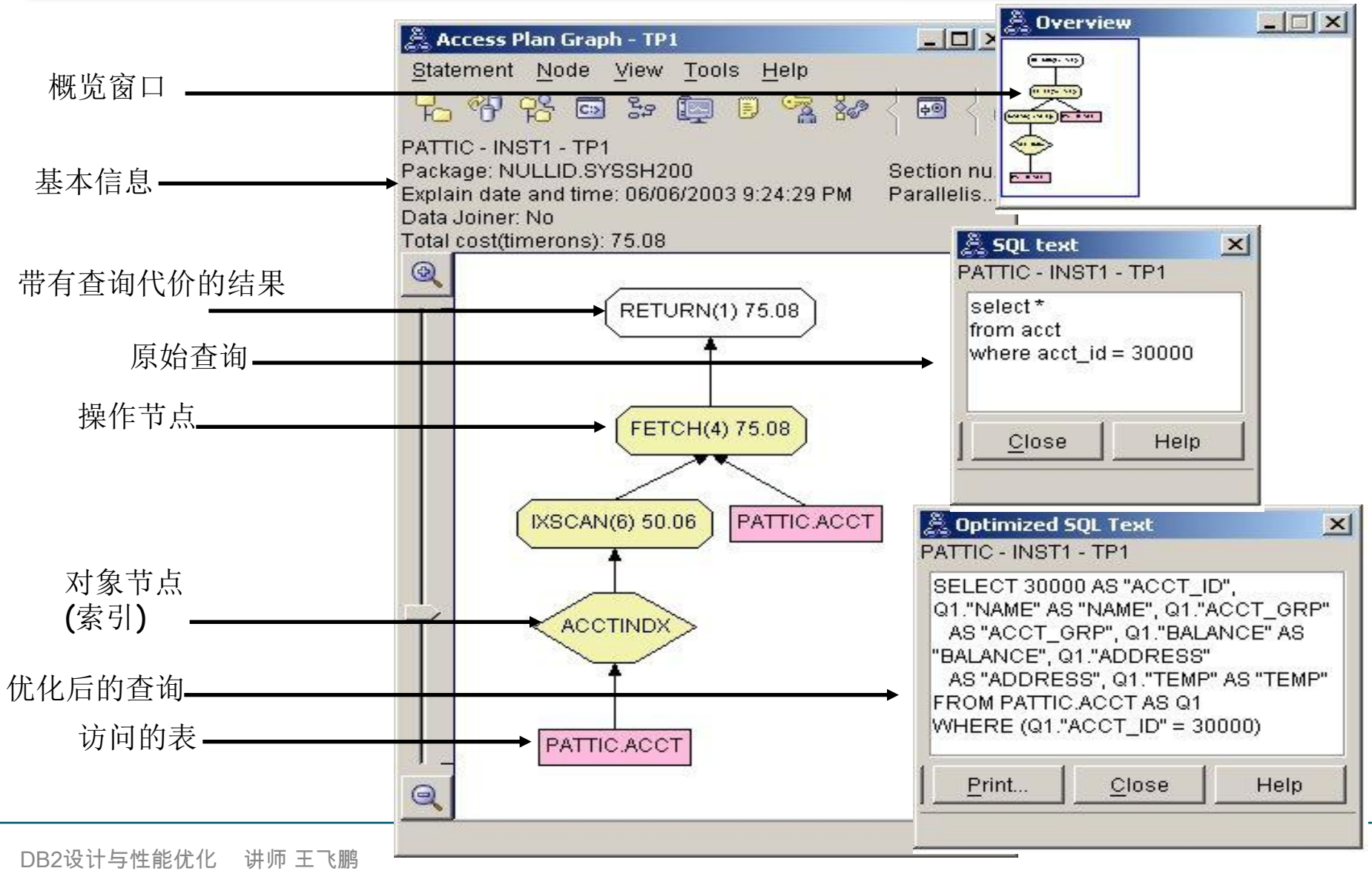
优化器：生成和选择最优的访问计划

Explain表：
存储访问计划信息

Visual Explain – DB2控制中心（或命令编辑器）



访问计划图形描述


概览窗口
基本信息
带有查询代价的结果
原始查询
操作节点
对象节点
(索引)
优化后的查询
访问的表
Access Plan Graph - TP1
Statement Node View Tools Help
PATTIC - INST1 - TP1
Package: NULLID.SYSSH200
Explain date and time: 06/06/2003 9:24:29 PM
Data Joiner: No
Total cost(timerons): 75.08
RETURN(1) 75.08
FETCH(4) 75.08
IXSCAN(6) 50.06
PATTIC.ACCT
ACCTINDX
PATTIC.ACCT
Overview
SQL text
PATTIC - INST1 - TP1
select *
from acct
where acct_id = 30000
Close
Help
Optimized SQL Text
PATTIC - INST1 - TP1
SELECT 30000 AS "ACCT_ID",
Q1."NAME" AS "NAME", Q1."ACCT_GRP"
AS "ACCT_GRP", Q1."BALANCE" AS
"BALANCE", Q1."ADDRESS"
AS "ADDRESS", Q1."TEMP" AS "TEMP"
FROM PATTIC.ACCT AS Q1
WHERE (Q1."ACCT_ID" = 30000)
Print...
Close
Help

db2expln说明工具

- 文本化输出
- 简单易用，所有平台通用
- 能用于动态SQL和静态SQL
- 命令参数
  - -d *database-name*
  - -statement *query-statement*
  - *-stmtfile query-statement-file*
  - -output(o) *output-file*
  - -terminal(t)
  - -graph(g)

```
Optimizer Plan:

          Rows
        Operator
          (ID)
          Cost

          42
          n/a
        RETURN
         ( 1)
        15.8372
          |
          42
          n/a
        NLJOIN
         ( 2)
        15.8372
       /       \-\
     14           *
     n/a          |
   TBSCAN         42
    ( 3)         n/a
   7.59249   Table: DB2ADMIN
     |          EMPLOYEE
     14
    n/a
Table: DB2AMDIN
DEPARTMENT
```

db2expln -d sample -statement "SELECT e.firstnme, d.deptno, d.mgrno FROM employee e, department d where e.workdept = d.deptname " -g -output expln.out

db2exfmt

- 文本化输出
- 访问计划信息最详尽完整
- 访问计划信息保存在EXPLAIN表中
- 1. 创建EXPLAIN表,在数据库中只需建一次
  - db2 –tvf ~/sqllib/misc/EXPLAIN.DDL
- 2. 将SQL的说明信息填充到EXPLAIN表，两种方式：
  - explain plan for **select … from …;**
  - set current explain mode explain;
    select … from … ;
    set current explain mode no;
- 3. 格式化说明信息，生成访问计划
  - db2exfmt -d <db> -1 -o <filename>

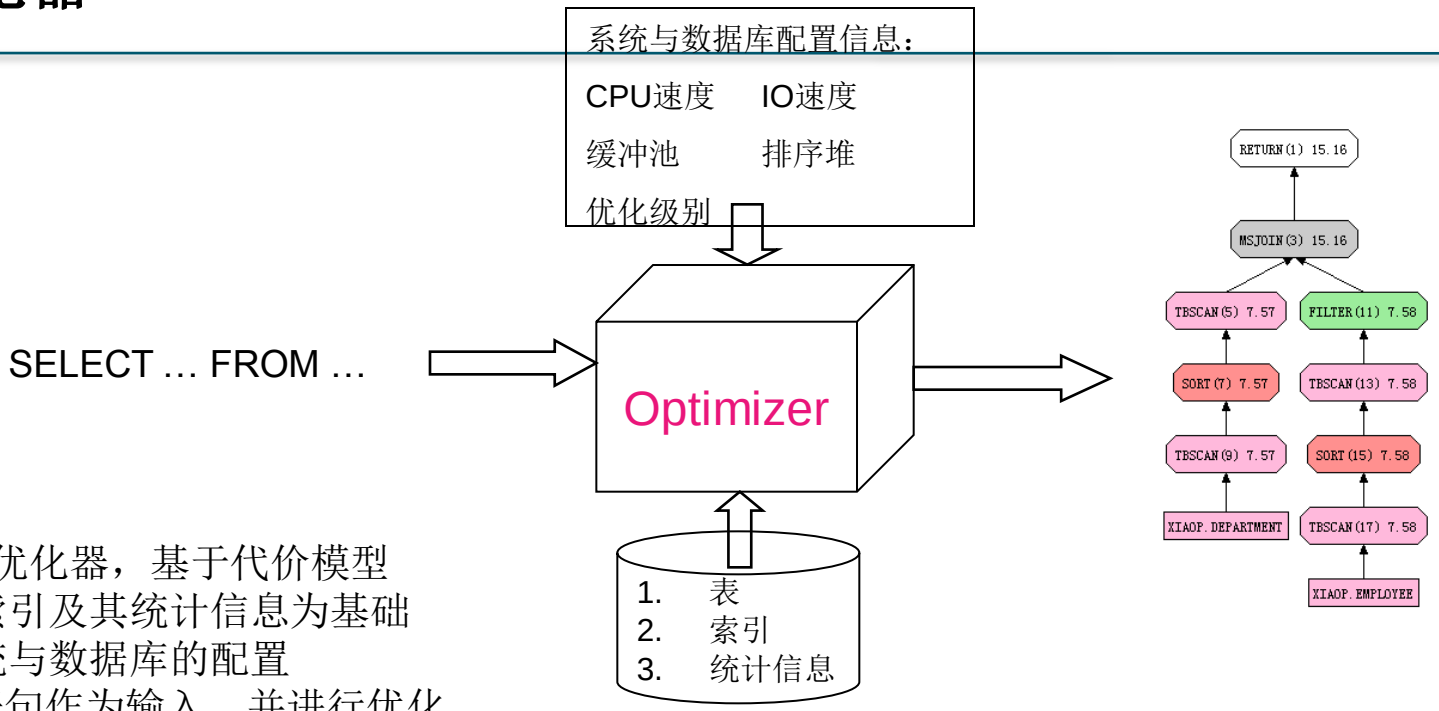
```
Access Plan:
-----------
        Total Cost:
        15.8372
        Query Degree:                        1

                    Rows
                   RETURN
                   (   1)
                    Cost
                     I/O
                     |
                     42
                   NLJOIN
                   (   2)
                   15.8372
                     2
                /--------+--------\
              14                   3
            TBSCAN               TBSCAN
            (   3)               (   4)
            7.59249              7.62888
              1                    1
              |                    |
              14                   42
        TABLE:  DB2ADMIN     TABLE:  DB2ADMIN
          DEPARTMENT           EMPLOYEE
              Q1                   Q2
```

```
db2 connect to sample;
db2 explain plan for SELECT  e.firstnme, d.deptno, d.mgrno FROM employee e, department d where
e.workdept = d.deptname;
db2 connect reset;
db2exfmt –d sample -1 –o exfmt.out;
```

提纲

- 监控找出问题SQL
- 获取SQL的访问计划
- 解读和分析访问计划
- 调优SQL语句的招式
- 小结与练习

DB2优化器



- 业界最好的优化器，基于代价模型
  - 以表和索引及其统计信息为基础
  - 根据系统与数据库的配置
  - 以SQL语句作为输入，并进行优化
  - 生成DB2引擎能执行的最优访问计划
- SQL调优的几个方向
  - 表和索引的设计
  - 即时的统计信息
  - 数据库调优配置
  - 精致的SQL语句

这也是我们数据调优的几大方向！
通过分析访问计划逆向找出性能问题所在！

优化器的模型

- 基于代价的模型
  - 估算每个操作的基数(Cardinality,输出结果的行数)
  - 估算CPU, I/O, 内存和通信的代价
  - 考虑 I/O并行(预取), CPU并行(SMP & DPF)
- 计划评估和搜索策略
  - 自底向上的方式生成计划
  - 在不同的优化级别下使用动态规划或者贪心算法
  - 选择最优的计划
- 实现方式选择
  - 表扫描(table scan) vs. 索引扫描(index scan)
  - 连接：嵌套循环(nested-loop) vs. 排序合并(sorted-merge) vs. 哈希连接(hash join)
- 操作顺序的选择
  - 连接(Joins)
  - 谓词过滤(predicate)
  - 聚集(group by, sum)

访问计划：主要操作符一览

- TBSCAN：表扫描，扫描基本表或者系统临时表读取每一行数据
- IXSCAN：索引扫描，索引扫描减少读取数据的行数，得到RID(row id)
- FETCH：提取，通过索引扫描得到的RID从基本表中读取数据行
- FILTER ：过滤器，复杂谓词计算
- SORT：排序，按某些列的值对数据行进行排序，或者对RID列表排序
- RIDSCAN：扫描RID列表，从排序的RID列表中读取RID，用于列表预取
- IXAND: 索引AND操作，
- NLJN：嵌套循环连接
- HSJN：哈希连接
- MGJN：归并连接
- GROUPBY: 分组与聚集
- TQ：table queue，表队列，用于DPF或SMP环境下在不同的节点间传数据，有BTQ，DTQ，LTQ等。
- XML相关的操作：XSCAN，XISCAN

db2exfmt访问计划：全局上下文信息

```
DB2_VERSION:        09.07.3
SOURCE_NAME:        SQLC2H21
SOURCE_SCHEMA:      NULLID
SOURCE_VERSION:
EXPLAIN_TIME:       2011-12-20-18.21.59.031000
EXPLAIN_REQUESTER:  DB2ADMIN

Database Context:
----------------
     Parallelism:           None
     CPU Speed:             4.251098e-007
     Comm Speed:            100
     Buffer Pool size:      4250
     Sort Heap size:        256
     Database Heap size:    600
     Lock List size:        4096
     Maximum Lock List:     22
     Average Applications:  1
     Locks Available:       28835
```

```
Package Context:
---------------
     SQL Type:              Dynamic
     Optimization Level:    5
     Blocking:              Block All Cursors
     Isolation Level:       Cursor Stability



------------ STATEMENT 1  SECTION 201 ------------
     QUERYNO:         1
     QUERYTAG:        CLP
     Statement Type:    Select
     Updatable:         No
     Deletable:         No
     Query Degree:      1
```

访问计划：优化SQL和计划树

```
Original Statement:
------------------
SELECT e.firstnme, d.deptno, d.mgrno
FROM employee e, department d
where e.workdept = d.deptname


Optimized Statement:
-------------------
SELECT Q2."FIRSTNME" AS "FIRSTNME",
       Q1."DEPTNO" AS "DEPTNO",
       Q1."MGRNO" AS "MGRNO"
FROM "DB2ADMIN".DEPARTMENT AS Q1,
     "DB2ADMIN".EMPLOYEE AS Q2
WHERE (Q2."WORKDEPT"= Q1."DEPTNAME")
```

```
Access Plan:
-----------
     Total Cost:                    15.8372
     Query Degree:                  1

                    Rows
                   RETURN
                   (   1)
                    Cost
                     I/O
                     |
                     42
                   NLJOIN
                   (   2)
                   15.8372
                     2
                /--------+--------\
              14                    3
            TBSCAN                TBSCAN
            (   3)                (   4)
            7.59249               7.62888
              1                     1
              |                     |
              14                    42
      TABLE:  DB2ADMIN      TABLE: DB2ADMIN
           DEPARTMENT            EMPLOYEE
              Q1                    Q
```

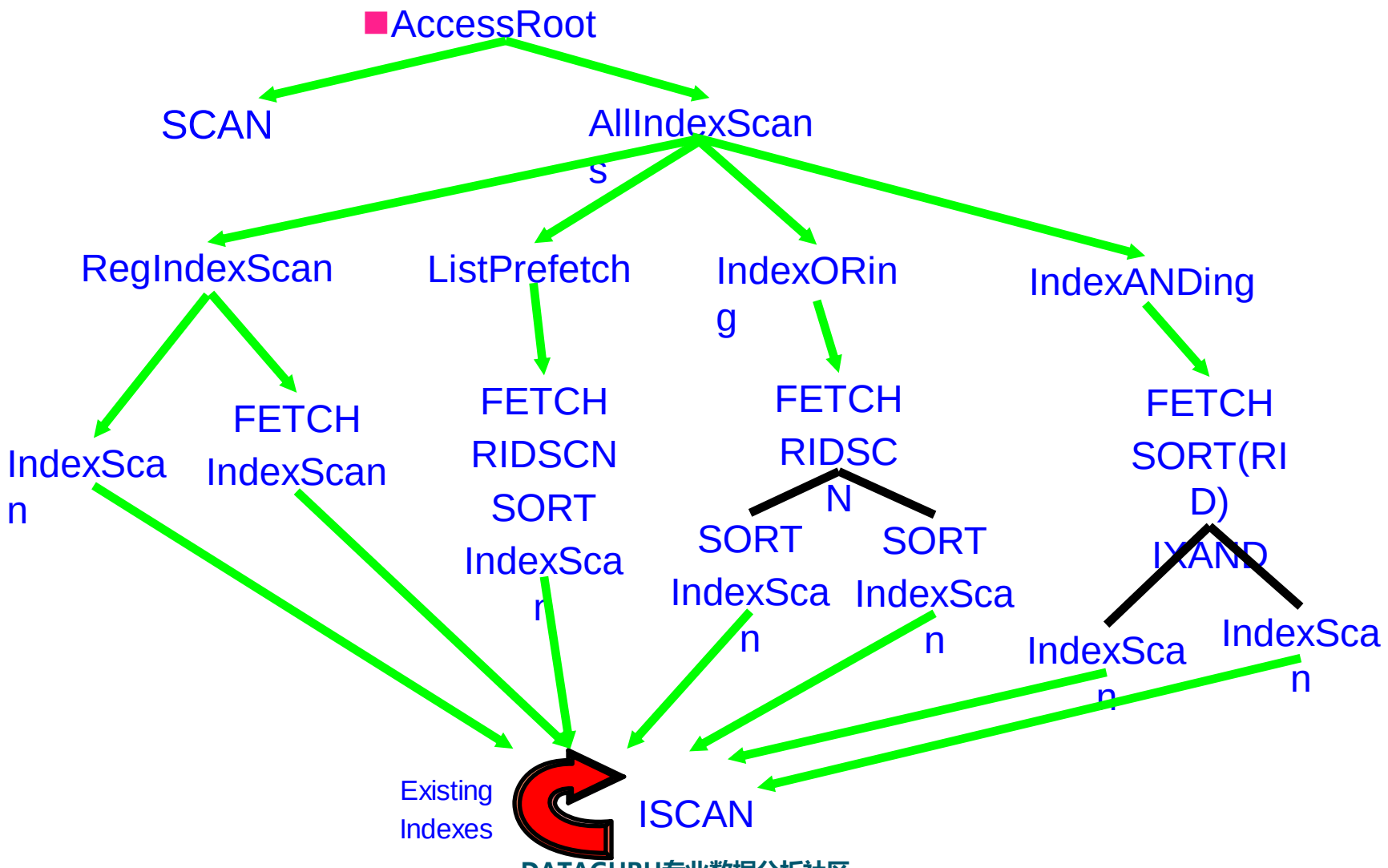
Rows
Operator Name
(Op number)
Cost
I/O

访问计划：操作符详细信息

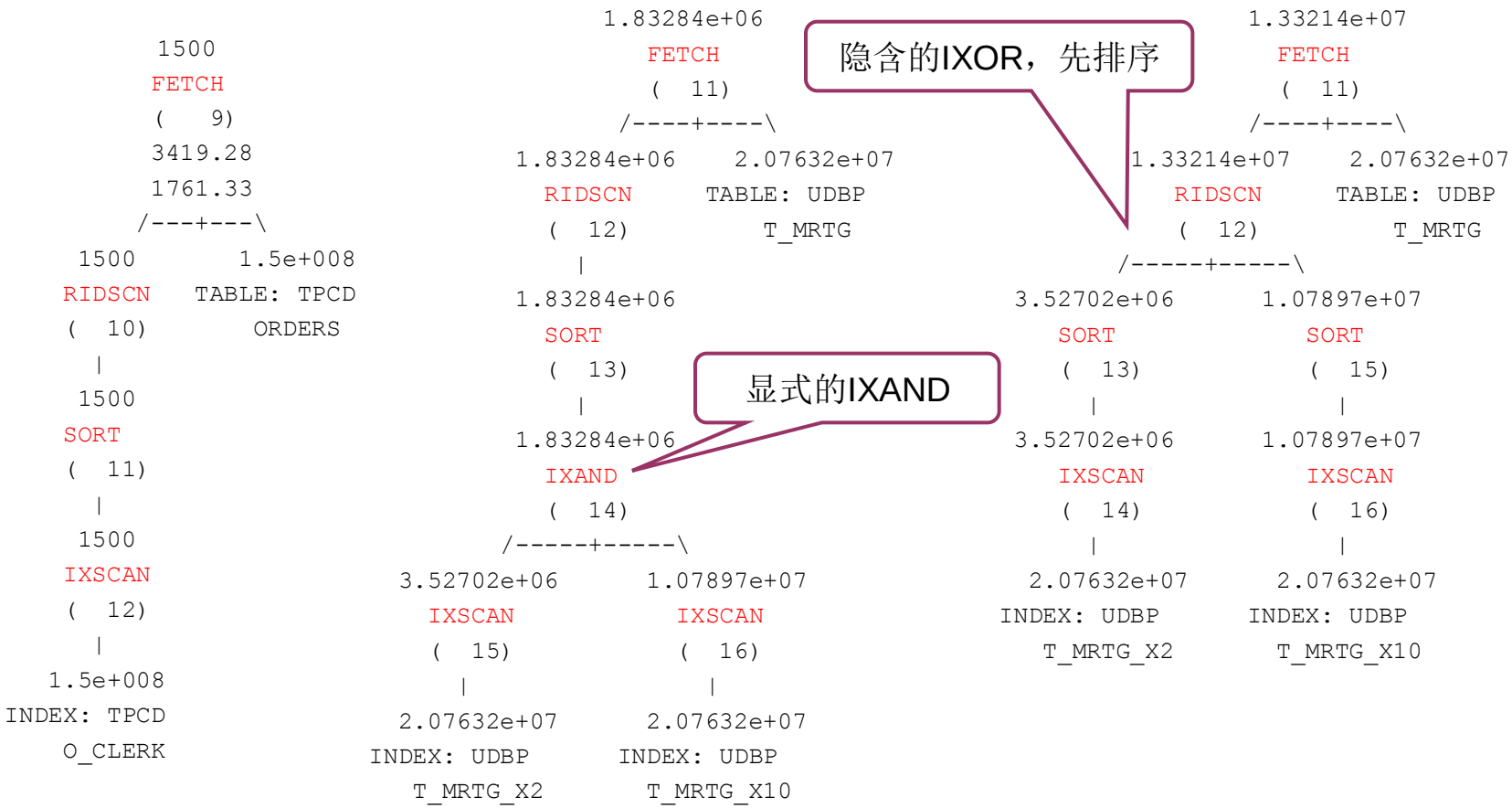
```
2) NLJOIN: (Nested Loop Join)
        Cumulative Total Cost:
        15.8372
        Cumulative CPU Cost:
        1.68698e+006
        Cumulative I/O Cost:                2
        Cumulative Re-Total Cost:
        0.674137
        Cumulative Re-CPU Cost:
        1.58579e+006
        Cumulative Re-I/O Cost:             0
        Cumulative First Row Cost:
        15.1792
        Estimated Bufferpool Buffers:       2

        Arguments:
        ---------
        EARLYOUT: (Early Out flag)
                  NONE
        FETCHMAX: (Override for FETCH
    MAXPAGES)
                  IGNORE
        ISCANMAX: (Override for ISCAN
    MAXPAGES)
                  IGNORE

        Predicates:
        ----------
        2) Predicate used in Join,
                  Comparison Operator:
        Equal (=)
                  Subquery Input Required:
        No
                  Filter Factor:
                  0.0714286
```

```
Input Streams:
-------------
        2) From Operator #3
                  Estimated number of rows:  14
                  Number of columns:
        3
                  Subquery predicate ID:     Not
    Applicable

                  Column Names:
                  ------------

        +Q1."MGRNO"+Q1."DEPTNO"+Q1."DEPTNAME"

        4) From Operator #4
                  Estimated number of rows:  3
                  Number of columns:
        1
                  Subquery predicate ID:     Not
    Applicable

                  Column Names:
                  ------------
                  +Q2."FIRSTNME"


Output Streams:
--------------
        5) To Operator #1
                  Estimated number of rows:  42
                  Number of columns:
        3
                  Subquery predicate ID:     Not
    Applicable

                  Column Names:
                  ------------
```

数据访问方式


AccessRoot
SCAN
AllIndexScans
RegIndexScan
ListPrefetch
IndexORing
IndexANDing
IndexScan
FETCH
IndexScan
FETCH
RIDSCN
SORT
IndexScan
FETCH
RIDSCN
SORT
IndexScan
SORT
IndexScan
FETCH
SORT(RID)
IXAND
IndexScan
IndexScan
Existing
Indexes
ISCAN

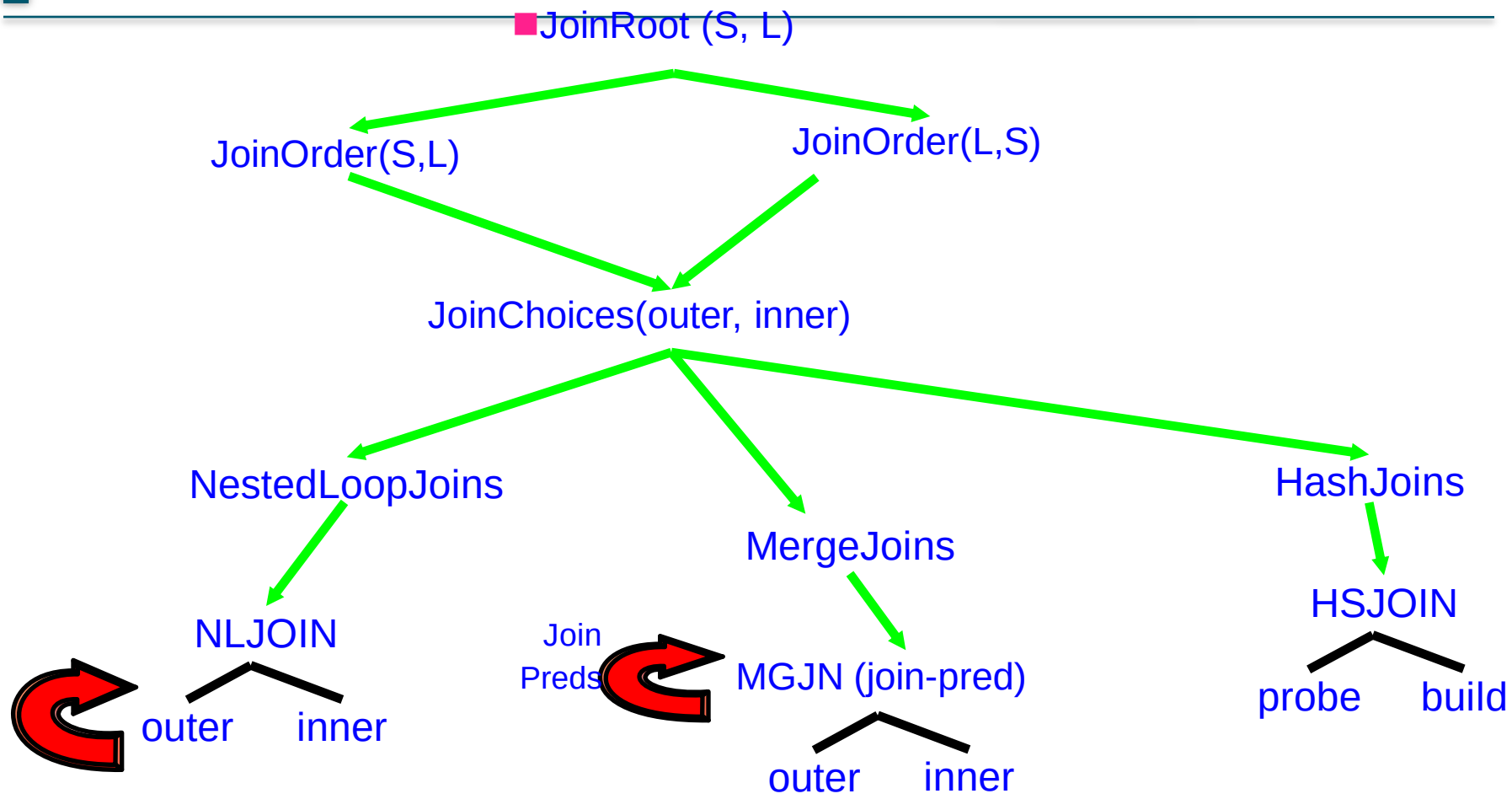
索引扫描：List Prefetch, Index ANDing和Index ORing



表访问方式问题分析

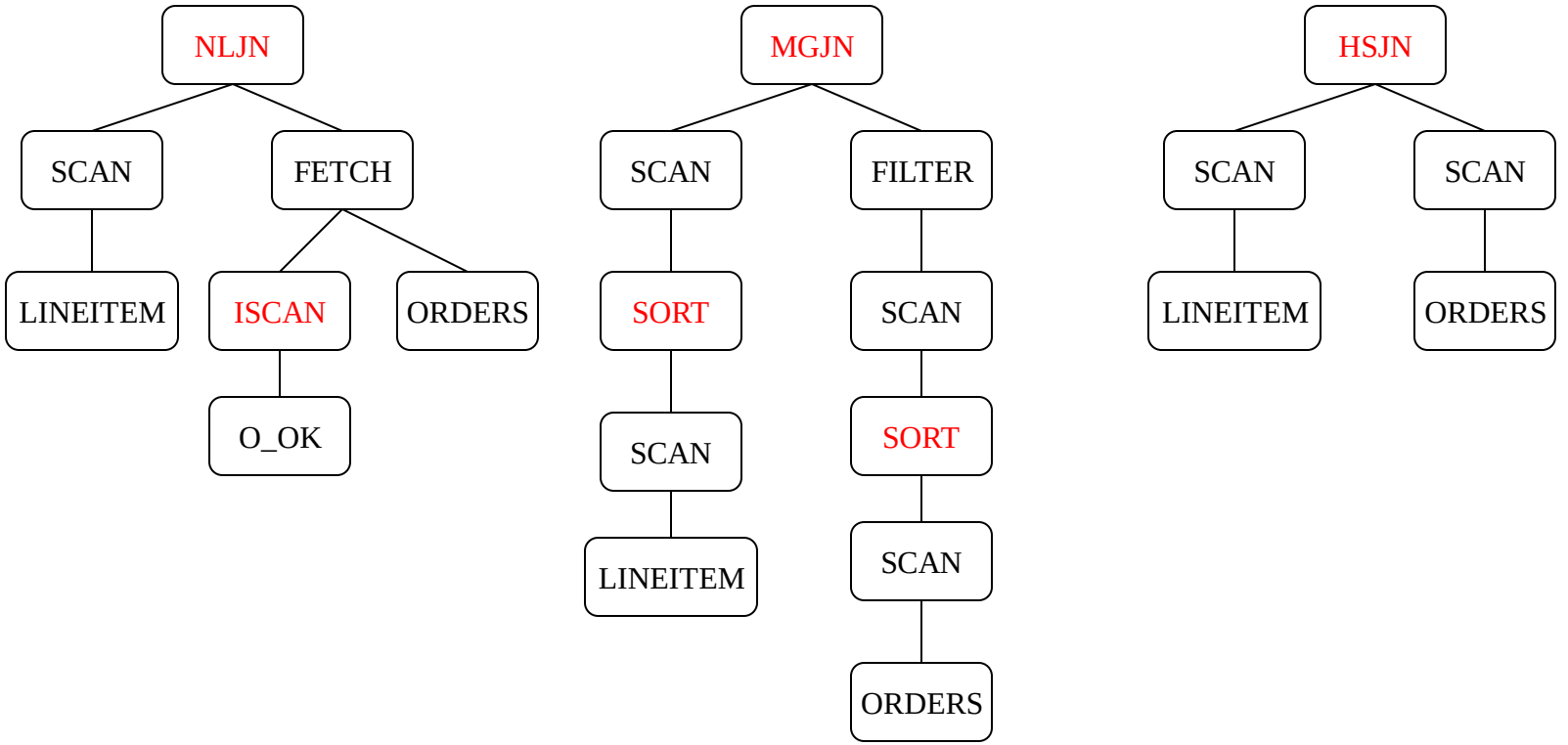
- 表扫描(TABSCAN)并不一定就是坏的
  - 当一个大表没有局部谓词又需要做连接时，TABSCAN可作为连接的外表
  - 当使用索引扫描的选择度太低，以致于还得再访问表中的大部分行时，DB2优化器可能会根据代价模型直接选择表扫描
- 索引丢失导致的表扫描
  - 如果查询计划中表扫描上的谓词选择度很高，对于大表只输出相对较少的行，表扫描操作的代价(cost)特别大，那么可能索引丢失，需要建立合适的索引来消除表扫描。
  - 另一个特征是快照监控时，发现读写效率(rows selected/read)特别低，也可能表明索引丢失。
- 索引没有用上
  - 检查索引列上的谓词选择度是否太低从而导致使用索引的代价太高，这是正常的
  - 检查SQL语句上谓词是否书写不当，比如类型不匹配，特别在类型自动转换导致的类型不匹配
  - 检查SQL语句中where条件是否在该列上使用函数

处理连接的策略

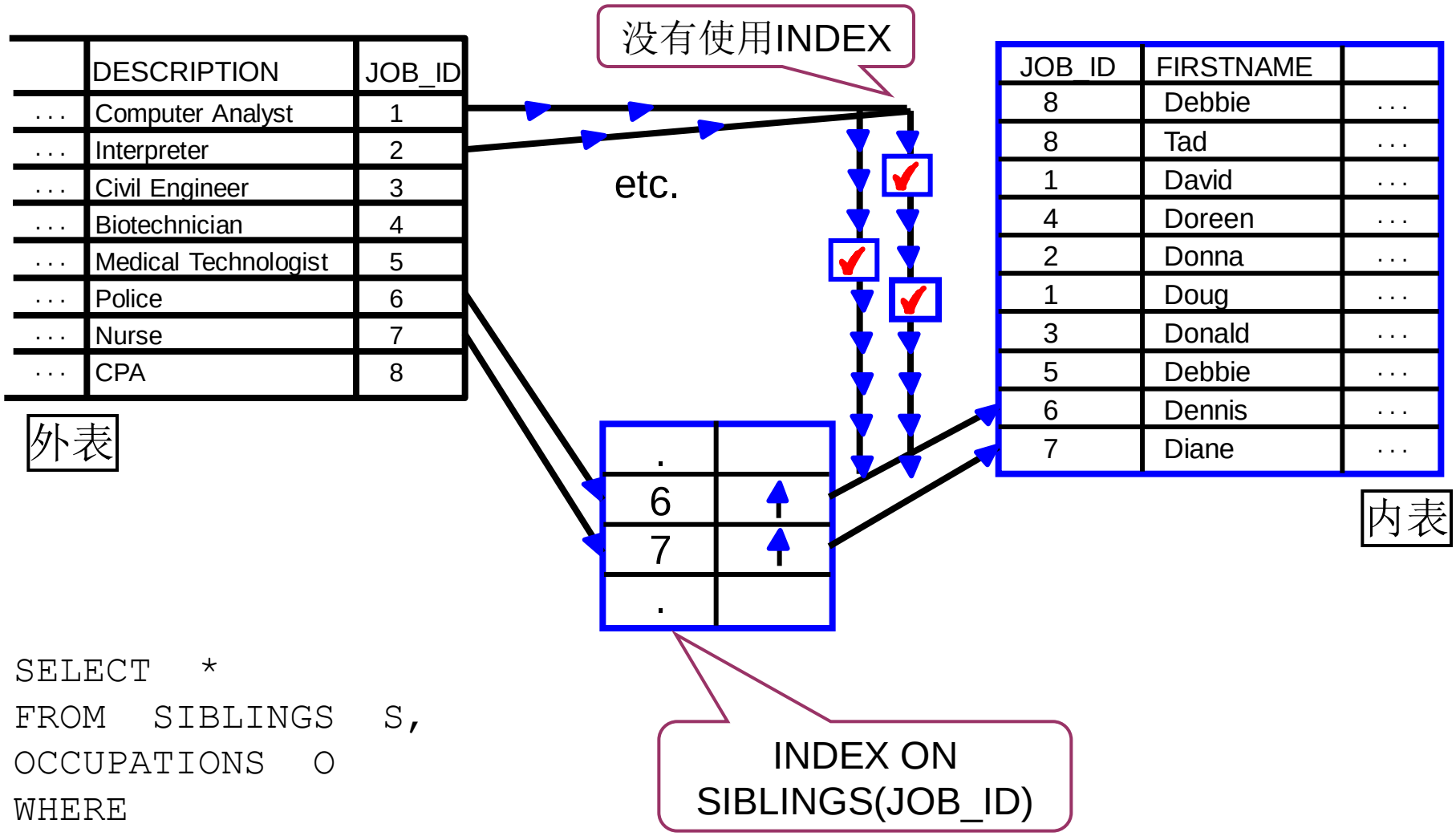

JoinRoot (S, L)
JoinOrder(S,L)
JoinOrder(L,S)
JoinChoices(outer, inner)
NestedLoopJoins
MergeJoins
HashJoins
NLJOIN
outer
inner
Join
Preds
MGJN (join-pred)
outer
inner
HSJOIN
probe
build

连接的不同方式

- SELECT … FROM LINEITEM L, ORDERS O WHERE L.ORDERKEY = O.ORDERKEY



嵌套循环连接



	DESCRIPTION	JOB_ID
...	Computer Analyst	1
...	Interpreter	2
...	Civil Engineer	3
...	Biotechnician	4
...	Medical Technologist	5
...	Police	6
...	Nurse	7
...	CPA	8

JOB_ID	FIRSTNAME	
8	Debbie	...
8	Tad	...
1	David	...
4	Doreen	...
2	Donna	...
1	Doug	...
3	Donald	...
5	Debbie	...
6	Dennis	...
7	Diane	...

.
6
7
.

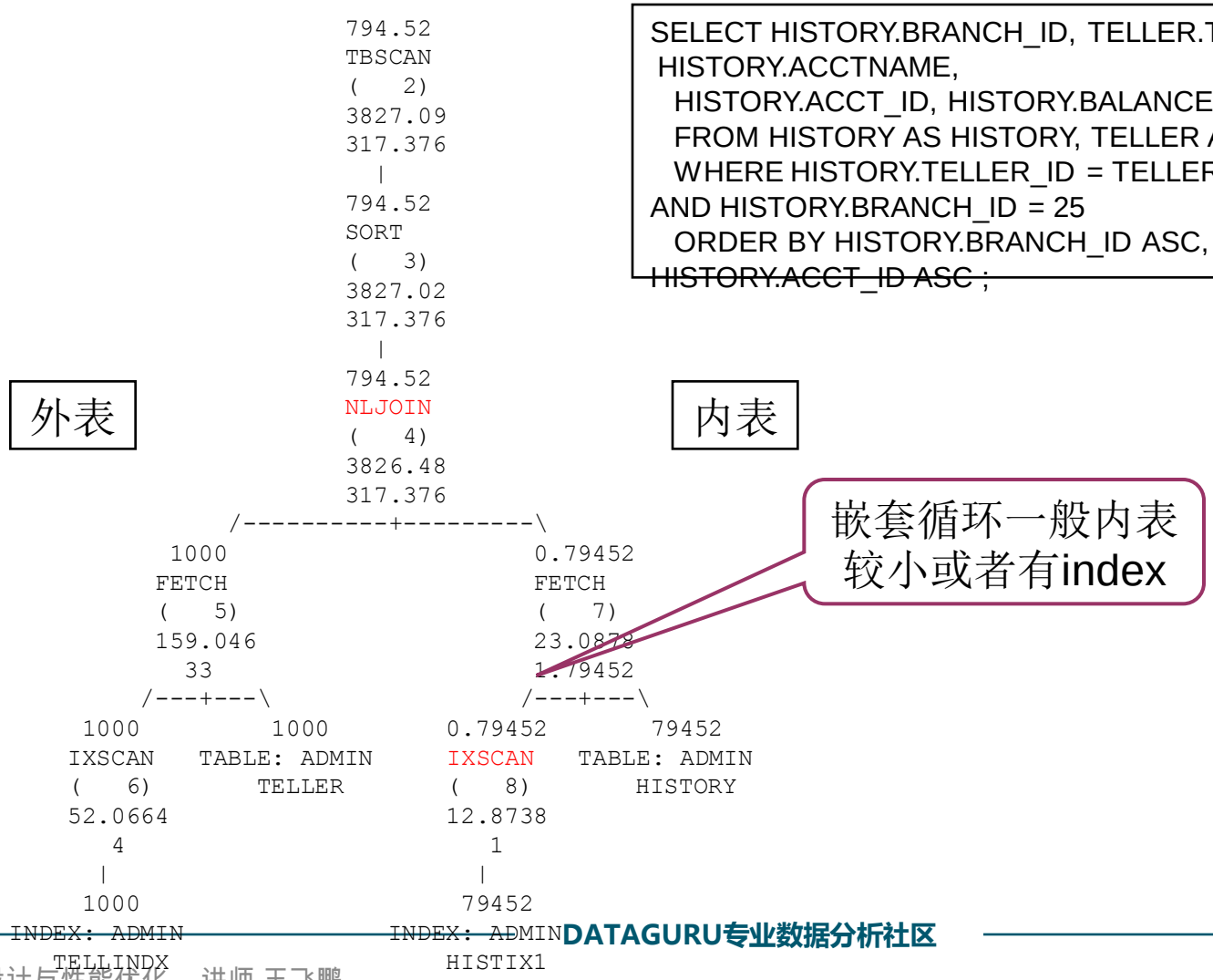
```
SELECT  *
FROM  SIBLINGS  S,
OCCUPATIONS  O
WHERE
S.JOB_ID = O.JOB_ID
```

嵌套循环的访问计划

```
SELECT HISTORY.BRANCH_ID, TELLER.TELLER_NAME,
 HISTORY.ACCTNAME,
  HISTORY.ACCT_ID, HISTORY.BALANCE
  FROM HISTORY AS HISTORY, TELLER AS TELLER
  WHERE HISTORY.TELLER_ID = TELLER.TELLER_ID
AND HISTORY.BRANCH_ID = 25
  ORDER BY HISTORY.BRANCH_ID ASC,
HISTORY.ACCT_ID ASC ;
```

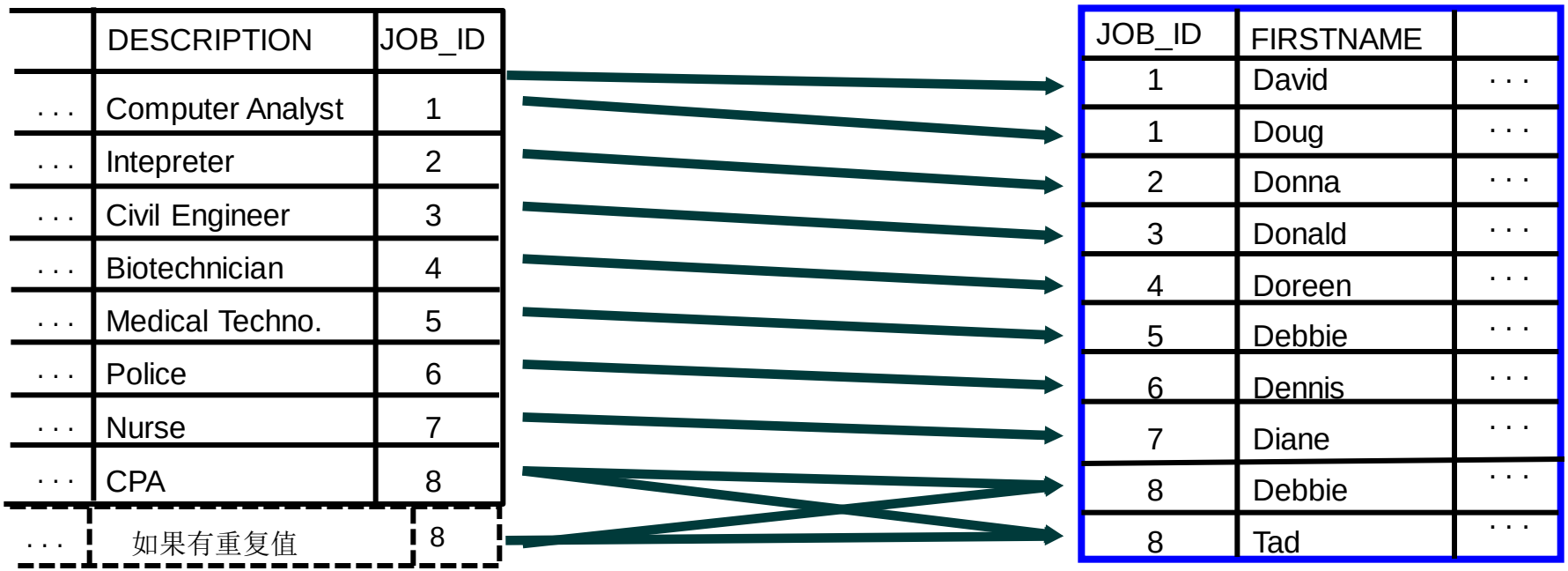
外表

内表



嵌套循环一般内表较小或者有index

归并连接



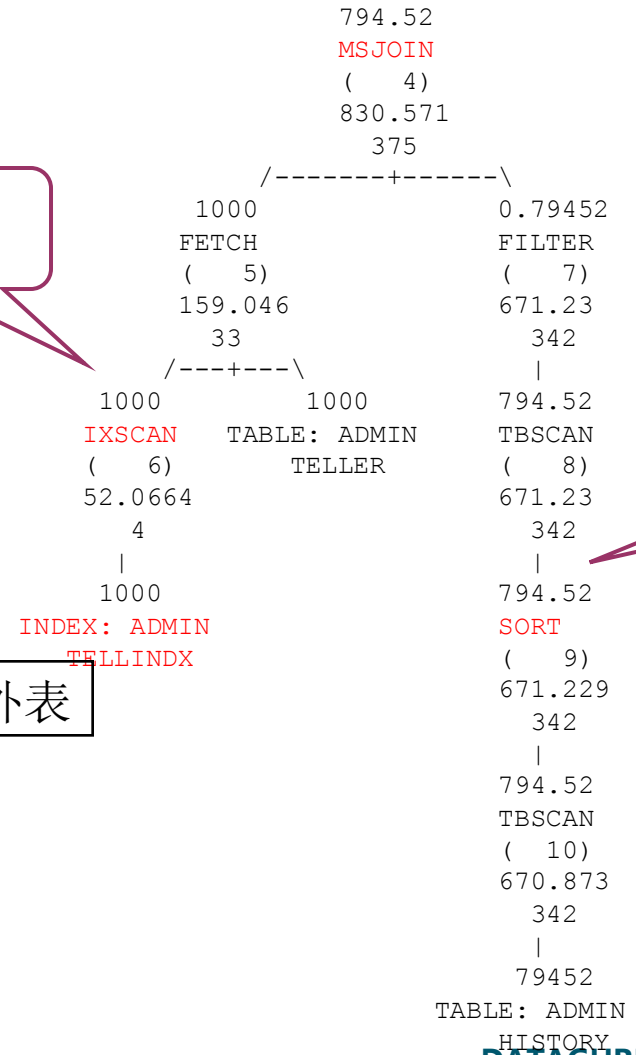
	DESCRIPTION	JOB_ID
...	Computer Analyst	1
...	Intepreter	2
...	Civil Engineer	3
...	Biotechnician	4
...	Medical Techno.	5
...	Police	6
...	Nurse	7
...	CPA	8
...	如果有重复值	8

JOB_ID	FIRSTNAME	
1	David	...
1	Doug	...
2	Donna	...
3	Donald	...
4	Doreen	...
5	Debbie	...
6	Dennis	...
7	Diane	...
8	Debbie	...
8	Tad	...

```
SELECT  *
FROM  SIBLINGS  S,
OCCUPATIONS  O
WHERE
S.JOB_ID = O.JOB_ID
```

归并连接需要内表和外表在连接的列都是有序的(索引可以避免显式排序操作)

归并连接的查询计划



索引保证连接列的序

外表

显式的排序操作保证连接列的序

内表

```
SELECT HISTORY.BRANCH_ID, TELLER.TELLER_NAME,
 HISTORY.ACCTNAME,
  HISTORY.ACCT_ID, HISTORY.BALANCE
  FROM HISTORY AS HISTORY, TELLER AS TELLER
  WHERE HISTORY.TELLER_ID = TELLER.TELLER_ID
AND HISTORY.BRANCH_ID = 25
  ORDER BY HISTORY.BRANCH_ID ASC,
HISTORY.ACCT_ID ASC ;
```

哈希连接

做连接前先对内表做预处理，建立Hash表

	DESCRIPTION	JOB_ID
...	Civil Engineer	3
...	Interpreter	2
...	Navigator	8
...	Biotechnician	4
...	Medical Technologist	5
...	Police	6
...	Nurse	7
...	Computer Analyst	1

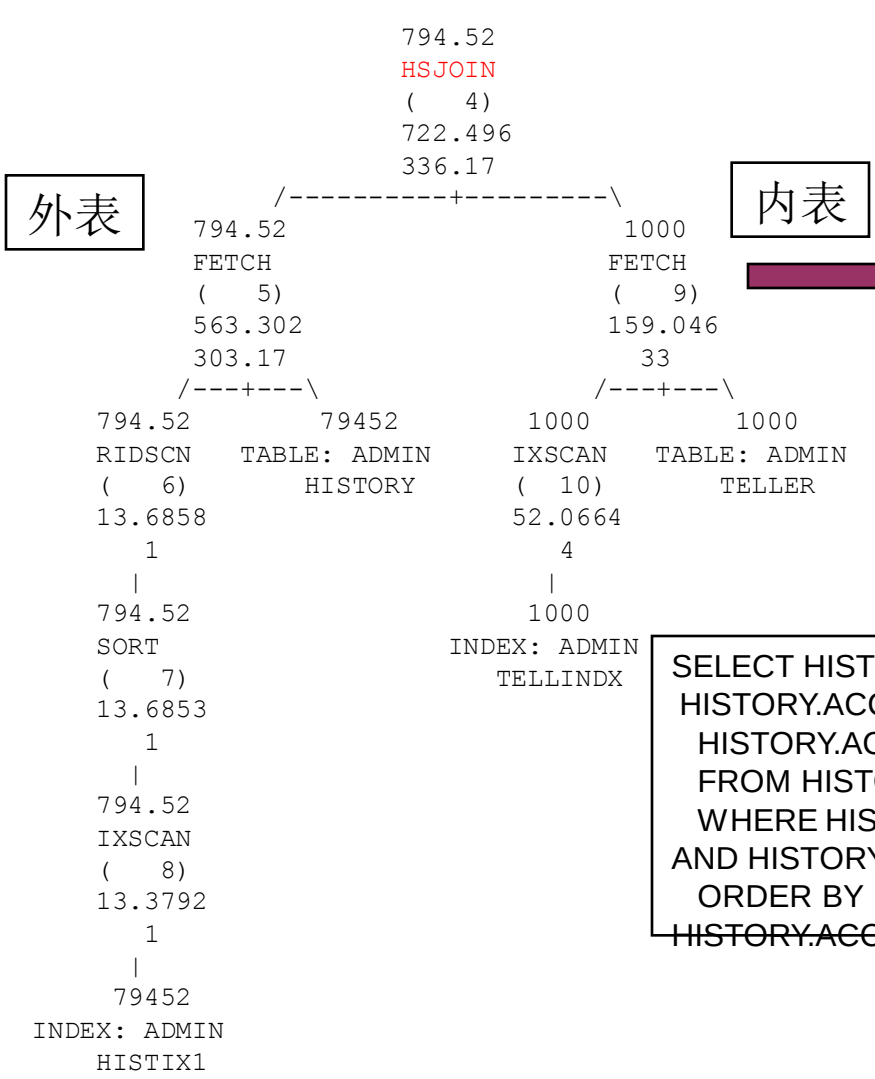
etc.

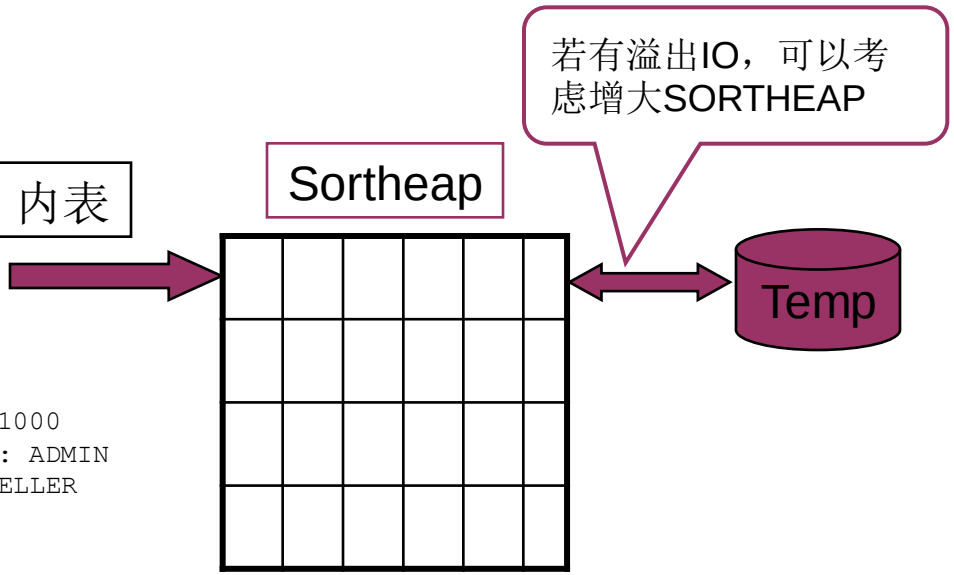
JOB_ID	FIRSTNAME		
3	Debbie	...	Bucket 1
4	Tad	...	
8	Gerhard	...	
8	Doreen	...	
1	Donna	...	Bucket 2
1	Doug	...	
2	David	...	
5	Debbie	...	
6	Dennis	...	Bu 3
7	Diane	...	

具有相同的Hashcode

```
SELECT  *
FROM  SIBLINGS  S,
OCCUPATIONS  O
WHERE
S.JOB_ID = O.JOB_ID
```

哈希连接访问计划






```
SELECT HISTORY.BRANCH_ID, TELLER.TELLER_NAME,
 HISTORY.ACCTNAME,
  HISTORY.ACCT_ID, HISTORY.BALANCE
  FROM HISTORY AS HISTORY, TELLER AS TELLER
  WHERE HISTORY.TELLER_ID = TELLER.TELLER_ID
AND HISTORY.BRANCH_ID = 25
  ORDER BY HISTORY.BRANCH_ID ASC,
HISTORY.ACCT_ID ASC ;
```

排序溢出

SELECT * FROM ACCT
WHERE ACCT_GRP BETWEEN 100 AND 150 order by balance desc

```
Access Plan:
-----------
            Total Cost: 51489.3
            Query Degree: 1

      Rows
     RETURN
     (   1)
      Cost
       I/O
       |
     51013.8
     TBSCAN
     (   2)
     51489.3
      28588
       |
     51013.8
     SORT
     (   3)
     51489.3
      28588
       |
     51013.8
     TBSCAN
     (   4)
     51449.8
      28588
       |
     1e+006
   TABLE: ADMIN
      ACCT
```

No I/O Cost To Sort 51K Rows

SELECT * FROM ACCT
WHERE ACCT_GRP BETWEEN 100 AND 800 order by balance desc

```
Access Plan:
-----------
            Total Cost: 345735
            Query Degree: 1

      Rows
     RETURN
     (   1)
      Cost
       I/O
       |
     700982
     TBSCAN
     (   2)
     345735
      68646
       |
     700982
     SORT
     (   3)
     309824
      48617
       |
     700982
     TBSCAN
     (   4)
     51556.8
      28588
       |
     1e+006
   TABLE: ADMIN
      ACCT
```

Large I/O Cost To Sort 700K Rows

排序溢出(续) – 操作符详情

SELECT * FROM ACCT
WHERE ACCT_GRP BETWEEN 100 AND 800 order by balance desc

```
3) SORT  : (Sort)
                Cumulative Total Cost:                  309824
                Cumulative CPU Cost:                    5.46755e+009
                Cumulative I/O Cost:                    48617
                Cumulative Re-Total Cost:     0
                Cumulative Re-CPU Cost:       0
                Cumulative Re-I/O Cost:       20029
                Cumulative First Row Cost:    309824
                Estimated Bufferpool Buffers:           48617

                Arguments:
                ---------
                DUPLWARN: (Duplicates Warning flag)
                          FALSE
                NUMROWS : (Estimated number of rows)
                          700982
                ROWWIDTH: (Estimated width of rows)
                          108
                SORTKEY : (Sort Key column)
                          1: Q1.BALANCE(D)
                SPILLED : (Pages spilled to bufferpool or disk)
                          20029
                TEMPSIZE: (Temporary Table Page Size)
                          4096
                UNIQUE  : (Uniqueness required flag)
                          FALSE
```

排序溢出到磁盘IO，IO代价很大？怎么解决？

排序溢出（续） -- 解决方案

- 调整数据库参数，增大排序堆SORTHEAP
- 在排序的列上创建index
  - 索引除了可以加速扫描外，合适的索引还可以消除访问计划的排序
  - 如果现在的计划是排序合并连接，可以通过建立索引是优化器选择NLJN
  - 唯一性索引还可以用来优化SQL的distinct查询
- 考虑应用逻辑，是否可以修改SQL
  - 去掉不必要的order by
  - 去掉不必要的distinct

提纲

- 监控找出问题SQL
- 获取SQL的访问计划
- 解读和分析访问计划
- 调优SQL语句的招式
- 小结与练习

调优SQL语句第一式 –整理维护

- 表或索引的统计信息是否过时,导致性能下降
  - 巧妇难为无米之炊，统计信息是优化器的米
  - 优化器估算的结果行数是否与实际结果差别大？
  - 做了大量更新操作后，可能会导致统计信息过时
- 表和索引的维护步骤
  - reorgchk：检查是否需要碎片整理
  - reorg： 执行碎片整理
  - runstats on schema.table with distribution and detailed indexes all
  - Rebind DB2的应用程序包

调优SQL语句第二式 – 建立索引

- 建立索引对SQL语句性能的好处:
  - 避免不必要的表扫描
  - 保证唯一性
  - 聚集索引保证表上的数据按该索引的键值顺序物理存放，减少IO
  - 避免排序
  - 显著提高查询的速度
  - 支持引用完整性，加速连接谓词的计算
  - 减少死锁的几率
- 建立索引的注意事项
  - 为每一个表建立主键
  - 建组合索引时，分析列的顺序
  - 不要建立冗余的索引
  - 不建无用的索引，保证你的索引物有所值
  - 在必要时用INCLUDE选项建索引，使得某些访问计划可使用INDEX-ONLY操作，无需访问基本表
- 理解索引的代价
  - 额外的存储空间
  - 插入、更新和删除(IUD)带来的额外开销
  - RUNSTATS, REORG, LOAD工具的代价

使用Design Advisor

命令行使用方法

```
db2advis -d sample -m MICP -i da.sql
```

-d database name

-m M-MQT I-Indexes C-MDC tables P-Partitioning

Workload type keyword: (choose one)

Create Explain Tables first

-s Single SQL statement

-i SQL from input file

-qp SQL from Query Patroller table

-w SQL from ADVISE_WORKLOAD table by workload name

-g Get workload from dynamic SQL snapshot

Other keywords:

-l number of MB available for indexes and MQTs (-1 for unlimited)

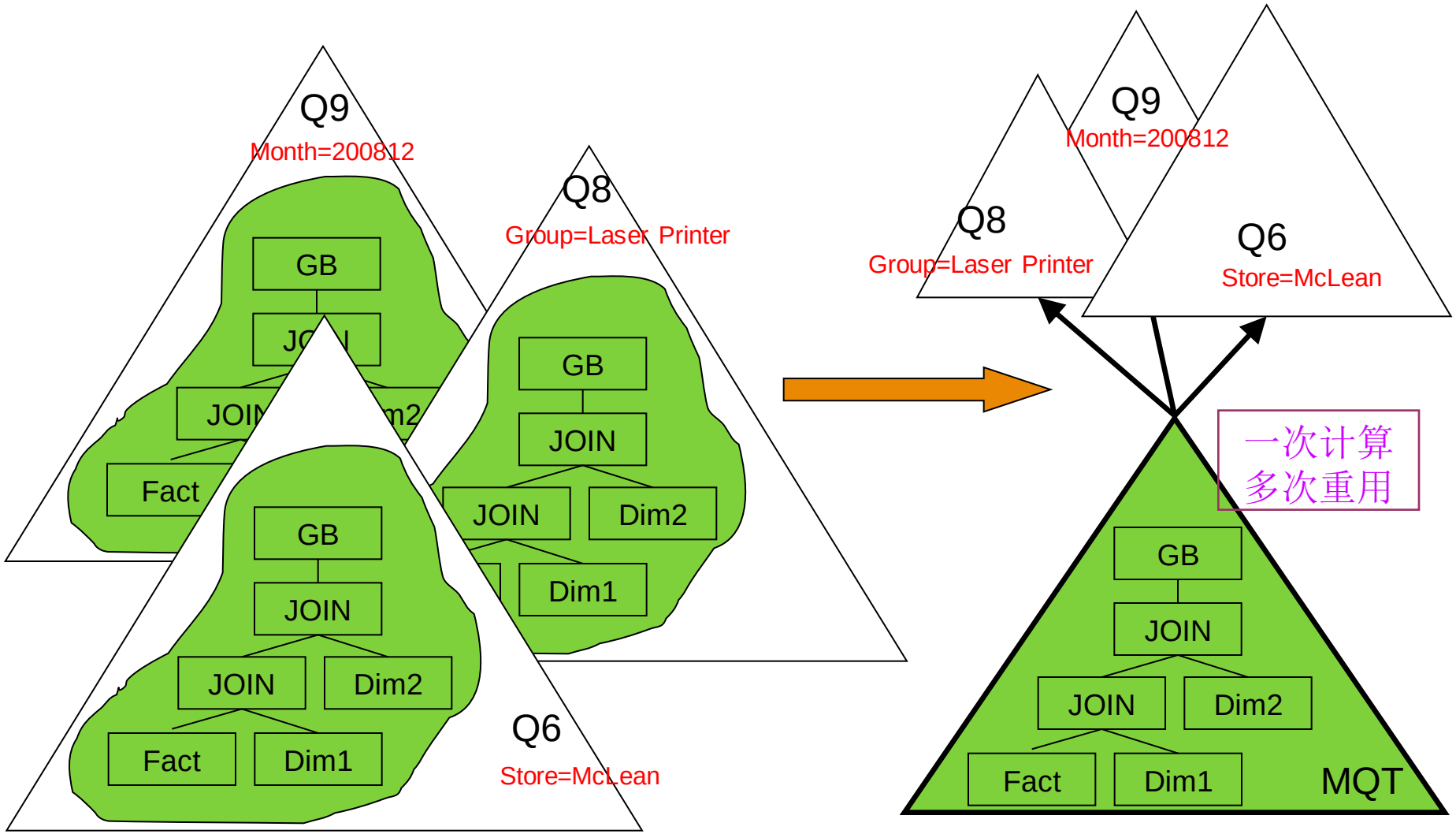
-t specifies the maximum time, in minutes, to complete the operation

利用db2advis推荐索引

db2advis -d TPCD -i cost.sql -m I > indexadv.txt

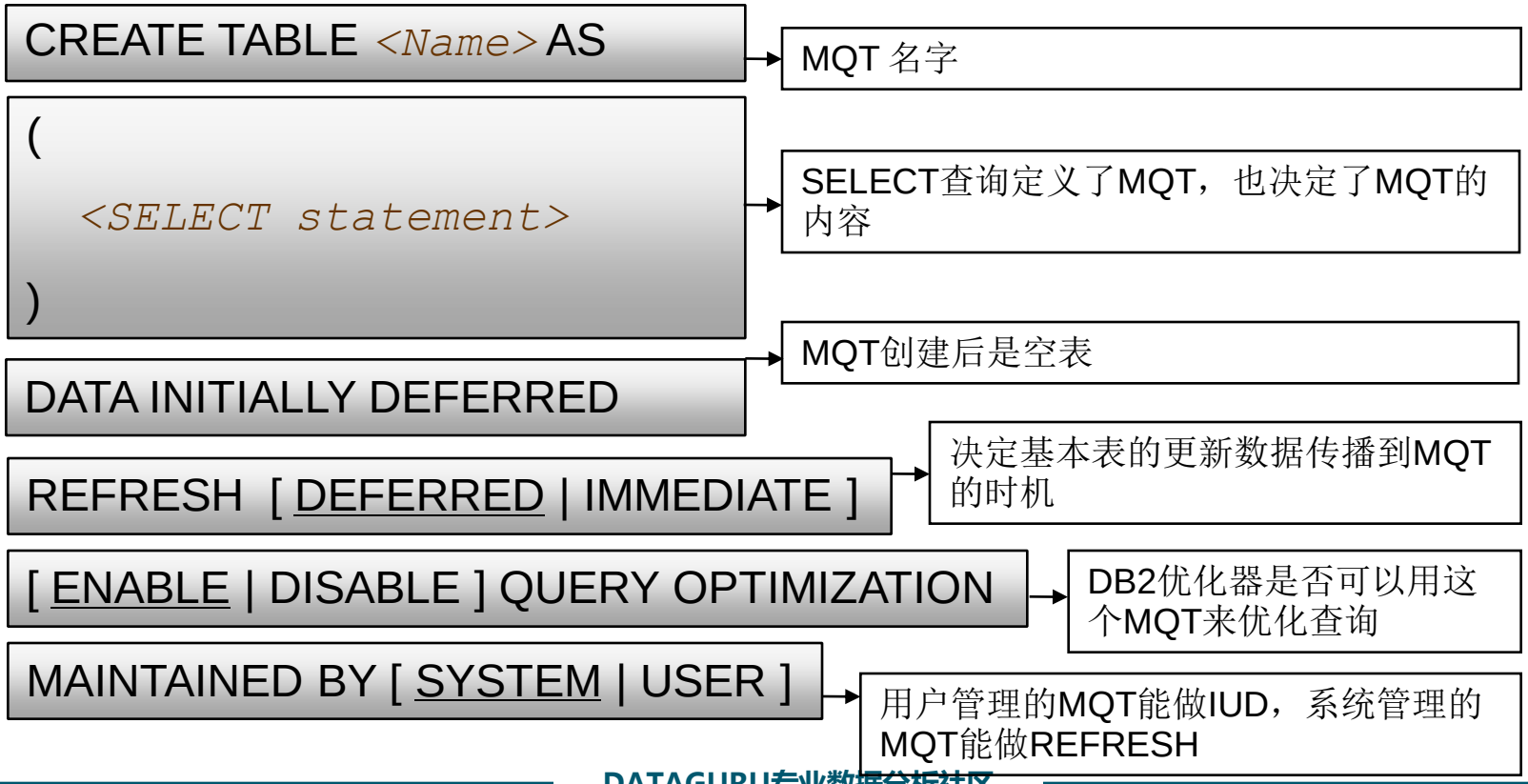
```
-- LIST OF RECOMMENDED INDEXES

CREATE INDEX "DB2ADMIN"."IDX509062043470000"
ON "TPCD    "."LINEITEM"
("L_RETURNFLAG" ASC, "L_DISCOUNT" ASC,
 "L_EXTENDEDPRICE" ASC, "L_ORDERKEY" ASC)
ALLOW REVERSE SCANS ;

RUNSTATS ON TABLE "TPCD    "."LINEITEM"
FOR INDEX "DB2ADMIN"."IDX509062043470000" ;

CREATE UNIQUE INDEX "DB2ADMIN"."IDX509062044160000"
ON "TPCD    "."ORDERS"
("O_ORDERDATE" ASC, "O_ORDERKEY" ASC, "O_CUSTKEY" ASC)
ALLOW REVERSE SCANS ;

RUNSTATS ON TABLE "TPCD    "."ORDERS"
FOR INDEX "DB2ADMIN"."IDX509062044160000" ;
```

调优SQL语句第三式 – MQT物化视图



MQT(物化视图)

- MQT将一些复杂查询（OLAP，大表的连接和聚集）的结果预先计算出来并存在磁盘上，如果后面的查询匹配或者部分匹配了MQT的定义，则可以重用MQT的结果数据，避免重复计算
- DB2优化器会自动的用MQT重写用户的查询，如果使用MQT比基本表更优的话



MQTs – 定义和查询示例

```
CREATE TABLE TP1SUM AS
(SELECT ACCT_GRP, COUNT(*) AS COUNTS,
SUM(BALANCE) AS GROUP_BALANCE
FROM ACCT
GROUP BY ACCT_GRP)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

REFRESH TABLE TP1SUM;
RUNSTATS ON TABLE ADMIN.TP1SUM;
```

```
SELECT ACCT_GRP, SUM(BALANCE) AS SUM_BALANCE
FROM ACCT
WHERE ACCT_GRP BETWEEN 20 AND 80
GROUP BY ACCT_GRP
HAVING SUM(BALANCE) > 80000;
```

```
Access Plan:
-----------

        Rows
      RETURN
      (   1)
       Cost
        I/O
        |
    32.2464
     TBSCAN
     (   2)
    50.3867
        2
        |
       100
TABLE: DB2ADMIN
     TP1SUM
```

往往极低的代价！

db2exfmt输出中MQT评估和使用信息

```
Extended Diagnostic Information:

--------------------------------

Diagnostic Identifier:        1

Diagnostic Details:    EXP0148W  The following MQT or statistical view was
        considered in query matching: "DB2ADMIN "."TP1SUM".


Diagnostic Identifier:        2

Diagnostic Details:    EXP0149W  The following MQT was used (from those considered) in
   query matching: "DB2ADMIN "."TP1SUM".
```

使用db2advis推荐MQT

db2advis -d TPCD -file sum.sql -m IM > imadv.txt

```
execution started at timestamp 2004-11-18-16.42.05.405000
Using the default table space name USERSPACE1
found [1] SQL statements from the input file
Recommending indexes...
Recommending MQTs...
Found 4 user defined views in the catalog table
Found [2] candidate MQTs
Getting cost of workload with MQTs
total disk space needed for initial set [  30.237] MB
total disk space constrained to         [  33.062] MB
Trying variations of the solution set.
Optimization finished.
  1  indexes in current solution
  1  MQTs in current solution
 [52972.0000] timerons  (without recommendations)
 [ 13.0000] timerons  (with current solution)
 [99.98%] improvement
--
```

使用db2advis推荐MQT(续)

```
-- LIST OF RECOMMENDED MQTs
-- ========================
-- MQT MQT411182143400000 can be created as a refresh immediate MQT
-- mqt[1],    0.032MB
   CREATE SUMMARY TABLE "DB2ADMIN"."MQT411182143400000" AS
(SELECT Q3.C0 AS "C0", Q3.C1 AS "C1", Q3.C2 AS "C2"
FROM TABLE(SELECT Q2.C0 AS "C0", SUM(Q2.C1) AS "C1", COUNT(* ) AS "C2"
FROM TABLE(SELECT Q1.ACCT_GRP AS "C0", Q1.BALANCE AS "C1"
FROM ADMIN.ACCT AS  Q1) AS Q2 GROUP BY Q2.C0) AS Q3)
DATA INITIALLY DEFERRED REFRESH IMMEDIATE IN TP1DMSAD          ;
   COMMIT WORK ;
   REFRESH TABLE "DB2ADMIN"."MQT411182143400000" ;
   COMMIT WORK ;
   RUNSTATS ON TABLE "DB2ADMIN"."MQT411182143400000" ;
   COMMIT WORK ;
-
-- LIST OF RECOMMENDED INDEXES
-- ===========================
-- index[1],    0.044MB
   CREATE INDEX "ADMIN"."IDX411182144040000" ON "ADMIN"."MQT411182143400000"
  ("C0" ASC, "C1" DESC) ALLOW REVERSE SCANS ;
   COMMIT WORK ;
   RUNSTATS ON TABLE "DB2ADMIN"."MQT411182143400000" FOR INDEX
   "ADMIN"."IDX411182144040000" ;
   COMMIT WORK ;
```

调优SQL第四式 – 语句集中器

- 语句集中器 – Statement Concentrator，可指定含有常量的动态语句能否使用语句缓存
- 数据库配置参数: stmt_conc OFF | LITERALS
- 对于并发比较大的OLTP系统，如果编译SQL的开销太大，特别是CPU开销，可考虑启用语句集中器
- 示例如下：

- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000020'
- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000070‘
- SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO=:L0

常量的位置一样

包缓存中的语句

DB2 将根据原始语句中的常量为:L0 提供具体的值(‘000020’ 或 ‘000070’) 。

调优SQL第五式—优化器profile

- 用Profile告诉优化器该怎么做
  - 再聪明的机器始终还是机器，人在某些时候需要直接给出指示
- db2set DB2_OPTPROFILE=YES
- 重新启动数据库
- 在SQL中内嵌Profile提示，或者注册优化Profile（无需修改应用）
- 告诉优化器使用HashJoin来连接两个表

```
SELECT SUM(O.AMOUNT)
FROM CUSTOMER C, ORDER O
WHERE C.C_NATIONKEY = 5 AND
      O.O_CUSTKEY = C.C_CUSTKEY
 /* <OPTGUIDELINES>
  <HSJOIN>
   <ACCESS TABLE ="CUSTOMER"/>
   <ACCESS TABLE =" ORDER "/>
  </HSJOIN>
 </OPTGUIDELINES> */;
```

提纲

- 监控找出问题SQL
- 获取SQL的访问计划
- 解读和分析访问计划
- 调优SQL语句的招式
- 小结与练习

小结

- 监控是调优SQL的出发点
  - 通过快照、事件监控器和管理视图等工具找到问题SQL
  - 优先对消耗资源最多的SQL调优
- 了解DB2处理SQL的内部机制，有三种工具可以获取访问计划
  - Visual Explain：图形化说明工具
  - db2expln：便捷说明工具
  - db2exfmt：全面的说明工具
- 分析SQL语句的访问计划需要了解DB2优化器
  - 优化器基于代价的优化模型：总代价，IO，CPU，通信和内存
  - 表访问：TABSACN vs IXSCAN
  - 连接方式:嵌套循环(nested-loop) vs. 排序合并(sorted-merge) vs. 哈希连接(hash join)
  - 排序及其溢出
- 调优SQL有多种方法，但一般基于访问计划和监控数据的分析
  - Reorg和runstats：碎片整理和即时收集最新的统计信息
  - 建立合适的索引
  - 使用MQT处理复杂的查询
  - 在高并发的OLTP中启用语句集中器
  - 优化器的Profile在某些情况可以指导优化器做选择

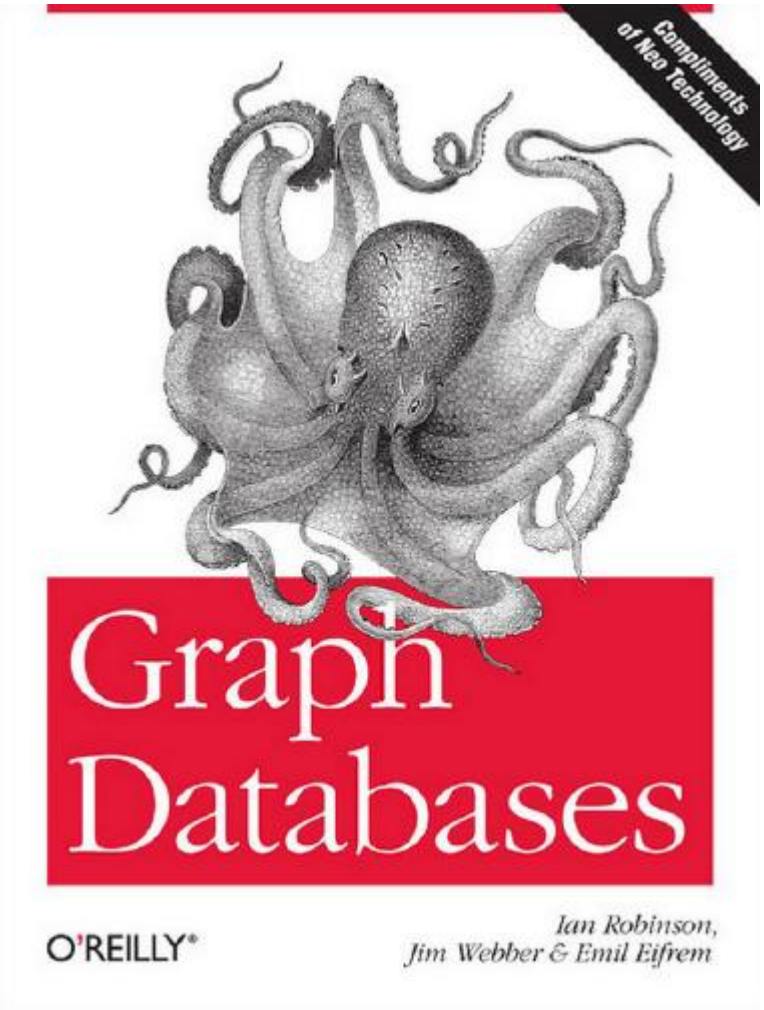
法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

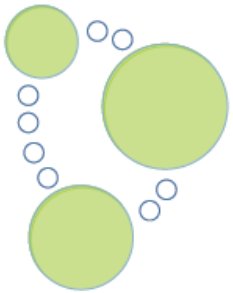
课程详情访问炼数成金培训网站

http://edu.dataguru.cn

图数据库参考书



The Neo4j Manual



neotechnology
value in relationships

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间