CERN IT Department

COOL performance optimization using Oracle hints

Andrea Valassi and Romain Basset (IT-DM)

With many thanks to Luca Canali for his help!

IT-DM Database Developers Workshop, 8th July 2008

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Outline

CERN IT Department

- **COOL basics (only what is needed to understand the rest…)**
  - Data model basics
  - Use case for this talk: MV tags (relational schema and SQL query)
  - Performance plots (how we define 'good' performance)

- **Oracle performance optimization strategy**
  - Basic SQL optimization (fix indexes and joins)
  - Execution plan instabilities (same SQL, different plans)
    - Observe (causes: unreliable statistics, bind variable peeking)
    - Analyze (**10053 trace files** and the BEGIN_OUTLINE block)
    - Fix (rewrite queries to please the Optimizer; then add **hints**)

COOL – basics

CERN IT Department

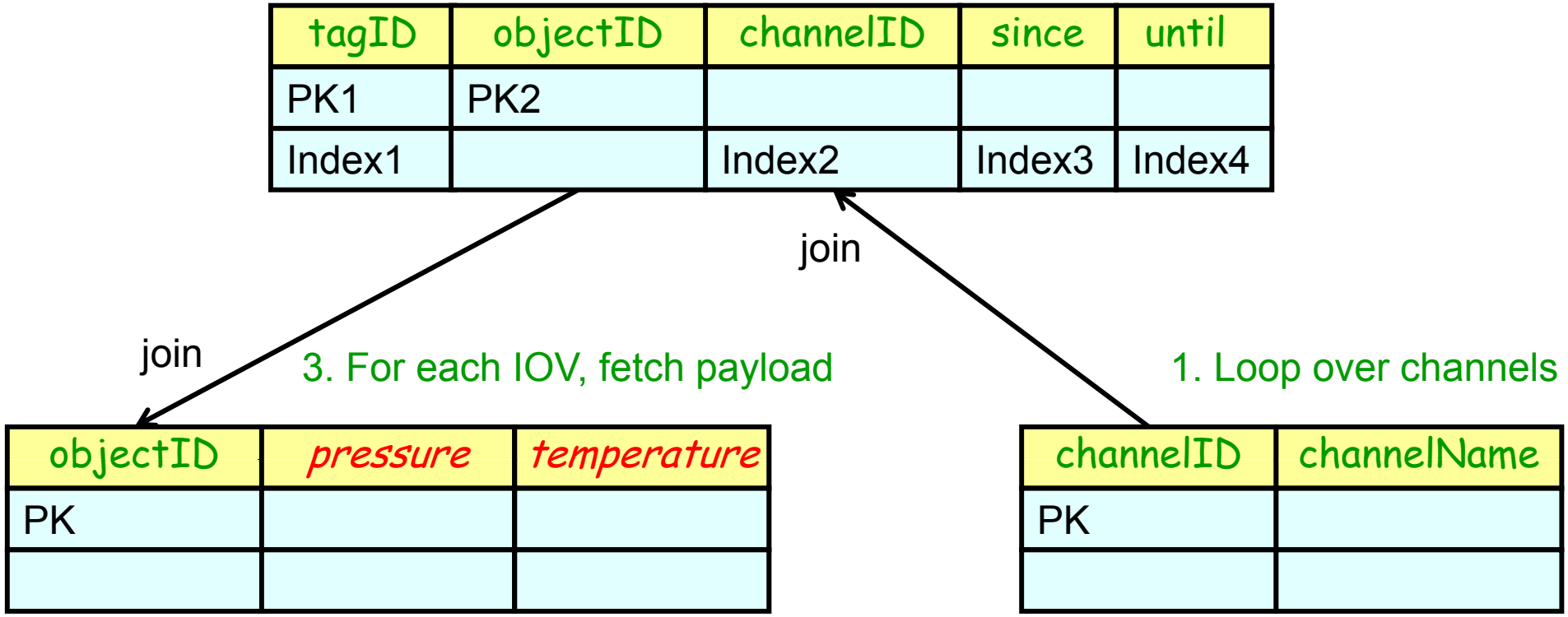
- **Conditions data**
  - *Detector data that vary in time and may be versioned*
  - Several use cases (different schemas and SQL queries to optimize)
    - Temperatures, voltages (measured – single version, SV)
    - Calibration, alignment (computed – multiple versions, MV)
- **COOL conditions objects ("IOV"s – interval of validity)**
  - Metadata: channel (c), IOV (t_{since} ,t_{until}), version or tag (v)
  - Data: user-defined "payload" (x1,x2,…)
  - *Typical query: retrieve the condition data payload X that was valid at time T in channel C for tag V*
- **COOL relational implementation (based on CORAL)**
  - Several backends (Oracle, MySQL…); C++ only (no PL/SQL)

COOL – test case: MV tag retrieval

CERN IT Department

Query: fetch all IOVs in [T1,T2] in tag PROD in all channels

2. For each channel C, select IOVs in tag PROD in [T1, T2]
(this is a *very large table* – and the most delicate part of the query to optimize)



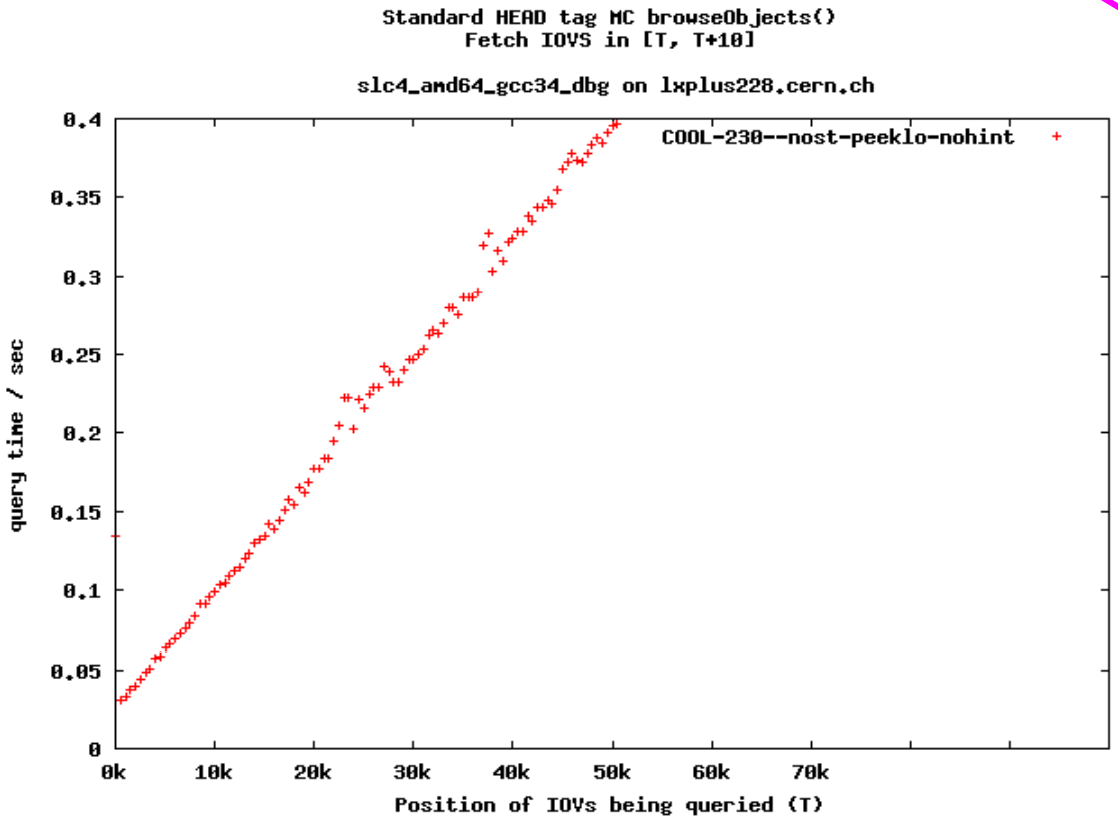
COOL – measuring performance

CERN IT Department

Is query time the same for all values of parameters T1, T2?

- *It was not in the initial COOL releases (≤ COOL 2.3.0)*
  - *Query time is higher for more recent IOVs than for older IOVs*

"tagId=PROD AND chId=C AND ((since ≤ T1< until) OR (T1 < since ≤ T2))"



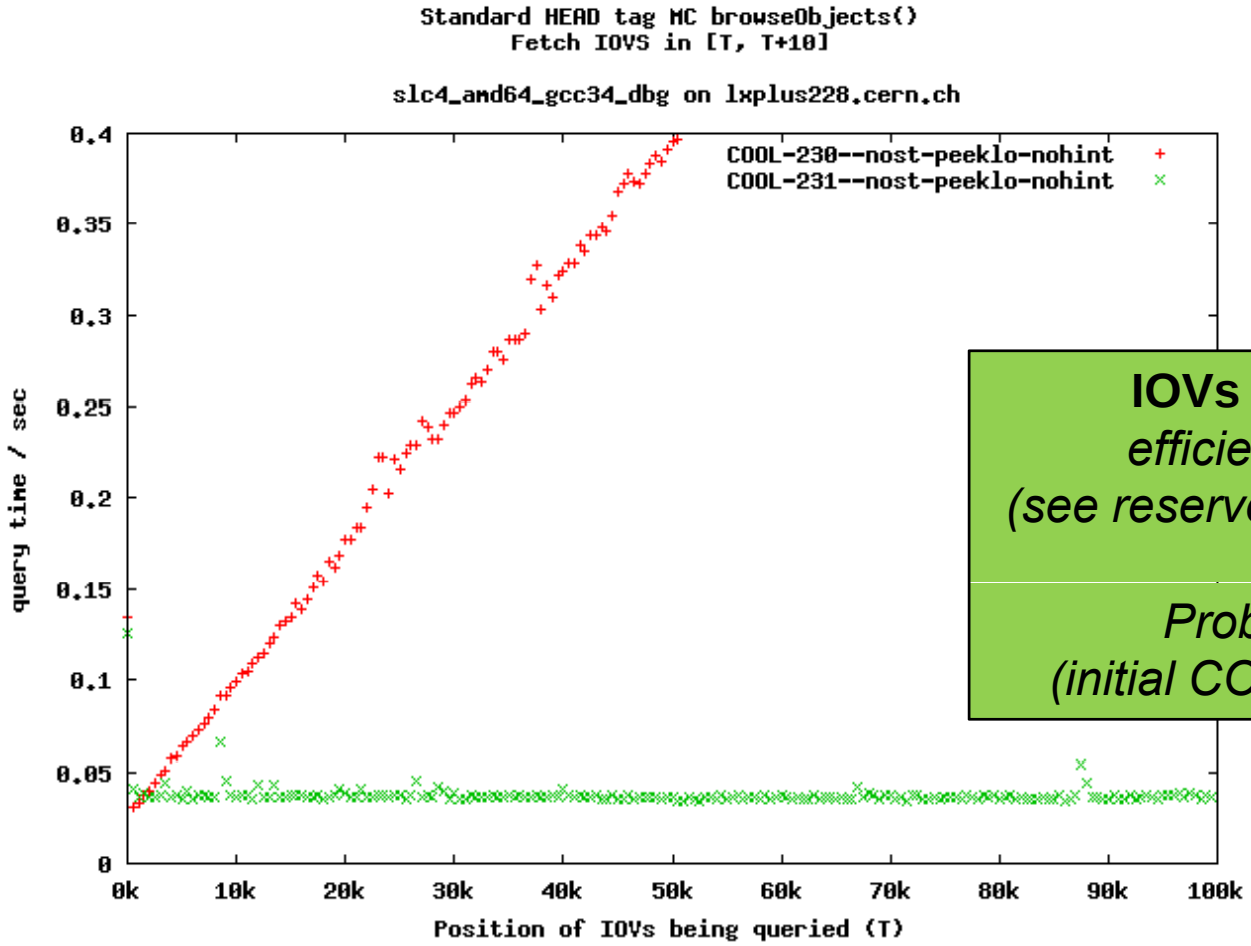
IOVs valid at t=T1 :
inefficient use of index for query on two columns since and until
(scan all IOVs with since ≤ T1, query time increases for high T1)

Basic optimization – better SQL

CERN IT Department

In tag PROD, in each channel at most one IOV is valid at T1

- Build a better SQL strategy from this constraint (unknown to Oracle)
  - The constraint is enforced in the C++ code, not in the database



IOVs valid at t=T1 :
efficient use of index
(see reserve slides for details...)

Problem fixed (?)
(initial COOL231 candidate)

Execution plan instabilities

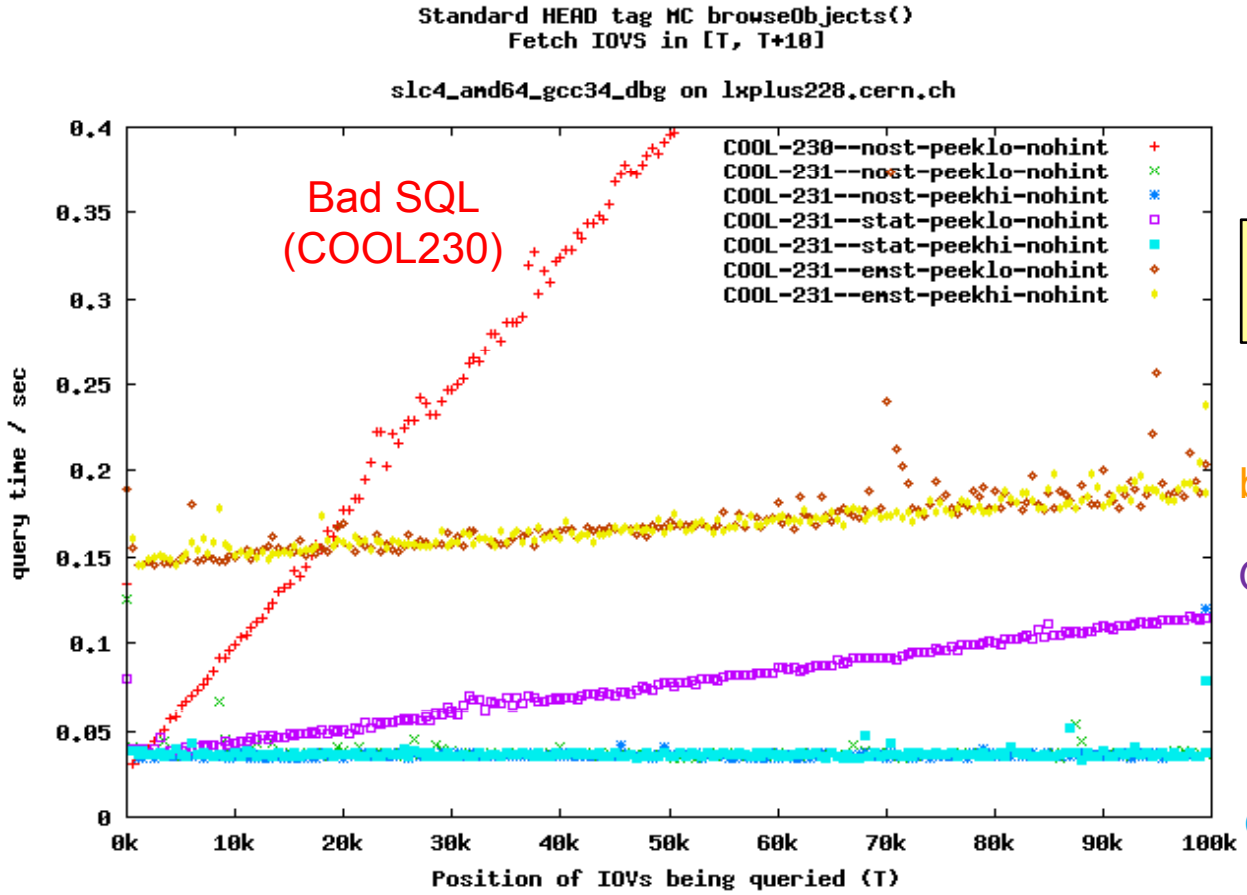
CERN IT Department

- **So, we thought the job was done...**
  - Query time used to increase, we managed to make it flat
- **But... every now and then our tests or our users reported performance issues again (...?...)**
  - Example: different performance in ATLAS tests at CNAF and LYON
- **Symptoms: same SQL, different execution plan**
  - In time, we identified two possible causes for this:
    - *Bind variable peeking*
      - *Optimal exec plan for finding old IOVs and recent IOVs are different*
      - *Problem if optimal plan for old IOVs is used for finding recent IOVs*
    - *Missing or unreliable statistics*
      - *Optimal exec plan is computed starting from wrong assumptions*

Execution plan instabilities – plots

CERN IT Department

- **Systematic study of 6 (2x3) cases**
  - 2 cases for b.v. peeking: peek "low" (old IOVs) or "high" (recent IOVs)
  - 3 cases for statistics: none, full, unreliable (empty tables)



Same 'good' SQL (COOL231), three different exec plans!

Good SQL (COOL231), bad stats (empty tables).

Good SQL (COOL231) and stats, peek 'low' (bad plan for 'high').

Good SQL (COOL231) and stats, peek 'high' (plan OK for all).
Good SQL (COOL231), NO stats.

Analyze plans – 10053 trace files

CERN IT Department

- **Look at the plan that was used for *your* query execution**
  - More reliable than 'explain plan', 'set autotrace' and other methods...
- **Look at how and why the Optimizer chose this plan**
  - Bind variable values
  - Alternative plans attempted
  - *Were user-supplied hints understood and used?*
    - *The "Dumping Hints" section at the end*
- **Look at the Optimizer's outline for the chosen plan**
  - *Get inspiration from the outline to prepare your user-supplied hints*
    - *The "BEGIN_OUTLINE_DATA" section towards the end*

Stabilize plans – add hints

CERN IT Department

- **This is an iterative process! In summary:**
  - 1. Execute your query for many cases (peek high/low...)
  - 2. Get plan and outline for a case with good performance
    - You want your plan to look like this in the end for *all* cases
  - 3. Do you need some query rewrite?
    - Are query blocks not named? Add QB_NAME and go to 1.
    - Is Oracle rewriting your query? Change SQL and go to 1.
    - Is Oracle using a different join order? Change SQL and go to 1.
  - 4. Is there a case with bad performance? Get its outline.
    - What is different in 'good' outline? Add as a hint and go to 1.
    - Was your hint not used or not useful? Try another and go to 1.
  - 5. Do all cases have good performance? You made it!

CERN

10053 technicalities

CERN IT Department

- **Generate a 10053 trace file 'myfile.trc'**
  - From SQL*Plus
    - ALTER SESSION SET EVENTS
      '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
    - ALTER SESSION SET tracefile_identifier='myfile'
  - From CORAL:
    - **export CORAL_ORA_SQL_TRACE_ON="10053"**
    - **export CORAL_ORA_SQL_TRACE_IDENTIFIER="myfile"**

- **Retrieve the trace file**
  - Ask your friendly DBA to get it from the server's udump...
  - *But please avoid generating (and asking for) trace files unless you need them... ;-)*

10053 technicalities – exec plan flush

CERN IT Department

- **You should invalidate existing exec plans between tests**
  - To remove the effect of bind variable peeking (e.g. when testing the effect of different bind variable values)
  - To make sure that execution plans are recomputed and ORA-10053 trace files are as complete as possible
- **To invalidate existing execution plans you may:**
  - Flush the shared pool (DBA only – affects the whole DB)
  - Simpler hack: alter a relevant table in a dummy way
    - e.g. **"ALTER TABLE mytable LOGGING;"**

Query rewrite – are *you* in control?

CERN IT Department

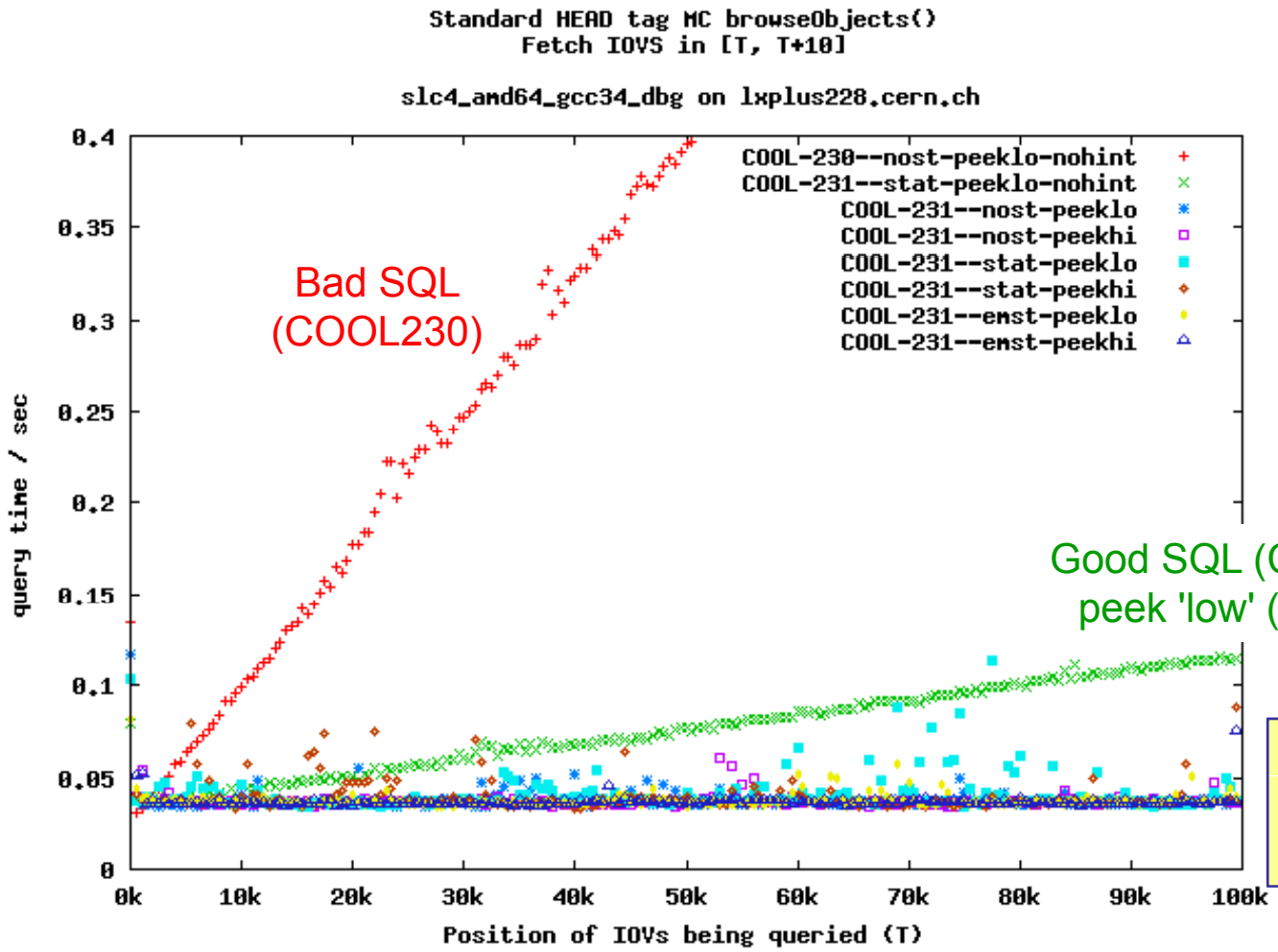
- **Master your query blocks**
  - *Name your query blocks – syntax is "/*+ QB_NAME(xxx) */"*
    - Else the Optimizer will name them for you (e.g. "SEL$1")
  - *The Optimizer rewrites your query blocks? Do it yourself!*
    - Symptoms: query block names like "SEL$3F979EFD", keywords like "MERGE" (remove inline views) or "CONCAT" (expand as union all)
    - Solution: do what the Optimizer would do (e.g. remove MERGE by expanding subqueries in WHERE clause into normal joins)
- **Master the order of your joins**
  - *The Optimizer reorders your joins? Do it yourself!*
    - Copy the Optimizer's favorite order from the "LEADING" keyword

Stabilized plan – results

CERN IT Department

Default hints added in COOL 2.3.1 release

– *Stable good plans in all 6 cases (2 bind var peeking x 3 statistics)*



Good SQL (COOL231) and stats, peek 'low' (bad plan for 'high').

Good SQL (COOL231), good/missing/bad stats, peek 'low' or ‘high’

Optimal query and hints

```
CORAL/RelationalPlugins/oracle    Debug Prepared statement : "SELECT /*+ QB_NAME
(MAIN) INDEX_RS_ASC(@MAIN COOL_I3@MAIN (TAG_ID "CHANNEL_ID" IOV_SINCE IOV_UNTIL)
) INDEX_RS_ASC(@MAIN COOL_I4@MAIN (OBJECT_ID)) LEADING(@MAIN COOL_C2@MAIN COOL_I
3@MAIN COOL_I4@MAIN) USE_NL(@MAIN COOL_I3@MAIN) USE_NL(@MAIN COOL_I4@MAIN) INDEX
(@MAX1 COOL_I1@MAX1 (TAG_ID "CHANNEL_ID" IOV_SINCE IOV_UNTIL)) */ COOL_I4."OBJE
CT_ID", COOL_I4."CHANNEL_ID", COOL_I4."IOV_SINCE", COOL_I4."IOV_UNTIL", COOL_I4.
"USER_TAG_ID", COOL_I4."SYS_INSTIME", COOL_I4."LASTMOD_DATE", COOL_I4."ORIGINAL_
ID", COOL_I4."NEW_HEAD_ID", COOL_I4."I" FROM LCG_COOL."S5820231_F0001_CHANNELS"
COOL_C2, LCG_COOL."S5820231_F0001_IOV2TAG" COOL_I3, LCG_COOL."S5820231_F0001_IOV
S" COOL_I4 WHERE COOL_I3."TAG_ID"=:"tagid3" AND COOL_I3."CHANNEL_ID"=COOL_C2."CH
ANNEL_ID" AND COOL_I3."IOV_SINCE">=COALESCE(( SELECT /*+ QB_NAME(MAX1)  */ MAX(C
OOL_I1."IOV_SINCE") FROM LCG_COOL.S5820231_F0001_IOV2TAG COOL_I1 WHERE COOL_I1."
TAG_ID"=:"tagid1" AND COOL_I1."CHANNEL_ID"=COOL_C2."CHANNEL_ID" AND COOL_I1."IOV
_SINCE"<=:"since1" ),:"sinc3s") AND COOL_I3."IOV_SINCE"<=:"until3" AND COOL_I3."
IOV_UNTIL">:"sinc3u" AND COOL_I4."OBJECT_ID"=COOL_I3."OBJECT_ID" ORDER BY COOL_I
3."CHANNEL_ID" ASC, COOL_I3."IOV_SINCE" ASC"
```

Good plan with hints (peek low)

```
| Id  | Operation                              | Name                          |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                               |
|   1 |  SORT ORDER BY                         |                               |
|   2 |   NESTED LOOPS                         |                               |
|   3 |    NESTED LOOPS                        |                               |
|   4 |     INDEX FULL SCAN                    | S5820231_F0001_CHANNELS_PK    |
|   5 |      SORT AGGREGATE                    |                               |
|   6 |       FIRST ROW                        |                               |
|   7 |        INDEX RANGE SCAN (MIN/MAX)      | S5820231_F0001_IOV2TAG_4INDX  |
|   8 |     TABLE ACCESS BY INDEX ROWID        | S5820231_F0001_IOV2TAG        |
|   9 |      INDEX RANGE SCAN                  | S5820231_F0001_IOV2TAG_4INDX  |
|  10 |       SORT AGGREGATE                   |                               |
|  11 |        FIRST ROW                       |                               |
|  12 |         INDEX RANGE SCAN (MIN/MAX)     | S5820231_F0001_IOV2TAG_4INDX  |
|  13 |    TABLE ACCESS BY INDEX ROWID         | S5820231_F0001_IOVS           |
|  14 |     INDEX UNIQUE SCAN                  | S5820231_F0001_IOVS_PK        |
-------------------------------------------------------------------------------
```

Bad plan (peek low)

```
| Id  | Operation                              | Name                          |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                               |
|   1 |  NESTED LOOPS                          |                               |
|   2 |   NESTED LOOPS                         |                               |
|   3 |    TABLE ACCESS BY INDEX ROWID         | S5820231_F0001_IOV2TAG        |
|   4 |     INDEX SKIP SCAN                    | S5820231_F0001_IOV2TAG_4INDX  |
|   5 |    INDEX UNIQUE SCAN                   | S5820231_F0001_CHANNELS_PK    |
|   6 |     SORT AGGREGATE                     |                               |
|   7 |      FIRST ROW                         |                               |
|   8 |       INDEX RANGE SCAN (MIN/MAX)       | S5820231_F0001_IOV2TAG_4INDX  |
|   9 |   TABLE ACCESS BY INDEX ROWID          | S5820231_F0001_IOVS           |
|  10 |    INDEX UNIQUE SCAN                   | S5820231_F0001_IOVS_PK        |
-------------------------------------------------------------------------------
```

Good plan with hints (peek low)

```
  /*+
    BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
      ALL_ROWS
      OUTLINE_LEAF(@"MAX1")
      OUTLINE_LEAF(@"MAIN")
      OUTLINE(@"MAX1")
      OUTLINE(@"MAIN")
      INDEX(@"MAIN" "COOL_C2"@"MAIN" ("S5820231_F0001_CHANNELS"."CHANNEL_ID"))
      INDEX_RS_ASC(@"MAIN" "COOL_I3"@"MAIN" ("S5820231_F0001_IOV2TAG"."TAG_ID" "S5
820231_F0001_IOV2TAG"."CHANNEL_ID" "S5820231_F0001_IOV2TAG"."IOV_SINCE" "S582023
1_F0001_IOV2TAG"."IOV_UNTIL"))
      INDEX_RS_ASC(@"MAIN" "COOL_I4"@"MAIN" ("S5820231_F0001_IOVS"."OBJECT_ID"))
      LEADING(@"MAIN" "COOL_C2"@"MAIN" "COOL_I3"@"MAIN" "COOL_I4"@"MAIN")
      USE_NL(@"MAIN" "COOL_I3"@"MAIN")
      USE_NL(@"MAIN" "COOL_I4"@"MAIN")
      INDEX(@"MAX1" "COOL_I1"@"MAX1" ("S5820231_F0001_IOV2TAG"."TAG_ID" "S5820231
_F0001_IOV2TAG"."CHANNEL_ID" "S5820231_F0001_IOV2TAG"."IOV_SINCE" "S5820231_F0001
_IOV2TAG"."IOV_UNTIL"))
    END_OUTLINE_DATA
  */
```

Bad plan (peek low)

```
  /*+
    BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
      ALL_ROWS
      OUTLINE_LEAF(@"MAX1")
      OUTLINE_LEAF(@"MAIN")
      OUTLINE(@"MAX1")
      OUTLINE(@"MAIN")
      INDEX_SS(@"MAIN" "COOL_I3"@"MAIN" ("S5820231_F0001_IOV2TAG"."TAG_ID" "S5820
231_F0001_IOV2TAG"."CHANNEL_ID" "S5820231_F0001_IOV2TAG"."IOV_SINCE" "S5820231_F0
001_IOV2TAG"."IOV_UNTIL"))
      INDEX(@"MAIN" "COOL_C2"@"MAIN" ("S5820231_F0001_CHANNELS"."CHANNEL_ID"))
      INDEX_RS_ASC(@"MAIN" "COOL_I4"@"MAIN" ("S5820231_F0001_IOVS"."OBJECT_ID"))
      LEADING(@"MAIN" "COOL_I3"@"MAIN" "COOL_C2"@"MAIN" "COOL_I4"@"MAIN")
      USE_NL(@"MAIN" "COOL_C2"@"MAIN")
      USE_NL(@"MAIN" "COOL_I4"@"MAIN")
      INDEX(@"MAX1" "COOL_I1"@"MAX1" ("S5820231_F0001_IOV2TAG"."TAG_ID" "S5820231
_F0001_IOV2TAG"."CHANNEL_ID" "S5820231_F0001_IOV2TAG"."IOV_SINCE" "S5820231_F0001
_IOV2TAG"."IOV_UNTIL"))
    END_OUTLINE_DATA
  */
```

```
Dumping Hints
=============
  atom_hint=(@=0xb7185dd4 err=0 resol=1 used=0 token=1123 org=1 lvl=3 txt=INDEX_RS_ASC ("COOL_I3" "S5820231_F0001_IOV2TAG_4INDX") )
  atom_hint=(@=0xb7185bc0 err=0 resol=1 used=0 token=1123 org=1 lvl=3 txt=INDEX_RS_ASC ("COOL_I4" "S5820231_F0001_IOVS_PK") )
  atom_hint=(@=0xb71859e4 err=0 resol=1 used=0 token=501 org=1 lvl=4 txt=LEADING ("COOL_C2" "COOL_I3" "COOL_I4") )
  atom_hint=(@=0xb7185820 err=0 resol=1 used=0 token=924 org=1 lvl=3 txt=USE_NL ("COOL_I3") )
  atom_hint=(@=0xb7187748 err=0 resol=1 used=0 token=924 org=1 lvl=3 txt=USE_NL ("COOL_I4") )
  atom_hint=(@=0xb7185540 err=0 resol=1 used=0 token=83 org=1 lvl=3 txt=INDEX ("COOL_I1" "S5820231_F0001_IOV2TAG_4INDX") )
  atom_hint=(@=0xb71860c8 err=0 resol=1 used=1 token=1003 org=1 lvl=2 txt=QB_NAME ("MAIN") )
  atom_hint=(@=0xb718461c err=0 resol=1 used=1 token=924 org=1 lvl=3 txt=USE_NL ("COOL_I4") )
  atom_hint=(@=0xb7180668 err=0 resol=1 used=1 token=1123 org=1 lvl=3 txt=INDEX_RS_ASC ("COOL_I4" "S5820231_F0001_IOVS_PK") )
  atom_hint=(@=0xb718057c err=0 resol=1 used=1 token=924 org=1 lvl=3 txt=USE_NL ("COOL_I3") )
  atom_hint=(@=0xb71807cc err=0 resol=1 used=1 token=1123 org=1 lvl=3 txt=INDEX_RS_ASC ("COOL_I3" "S5820231_F0001_IOV2TAG_4INDX") )
  atom_hint=(@=0xb71802b0 err=0 resol=1 used=1 token=501 org=1 lvl=4 txt=LEADING ("COOL_C2" "COOL_I3" "COOL_I4") )
  atom_hint=(@=0xb7183918 err=0 resol=1 used=1 token=1003 org=1 lvl=2 txt=QB_NAME ("MAX1") )
  atom_hint=(@=0xb7180380 err=0 resol=1 used=1 token=83 org=1 lvl=3 txt=INDEX ("COOL_I1" "S5820231_F0001_IOV2TAG_4INDX") )
```

Good plan with hints (peek low)

CORAL workarounds

CERN IT Department

- **No support for hints**
  - Implemented in COOL queries using SQL injection
    - Prepend the hint "/*+...*/" to the 1st item in the SELECT list
  - *This hack does not work for UPDATE, INSERT, DELETE*
  - CORAL support request sr #103420

- **No support for subqueries in WHERE clause**
  - Implemented in COOL queries using SQL injection
    - CORAL receives a WHERE clause that explicitly contains a fully qualified "(SELECT ... FROM ...) " subquery
    - COOL needs to know if it is talking to Oracle or MySQL (quotes)
  - CORAL support request sr #103547

COOL performance – in progress

CERN IT Department

- **Handle all use cases consistently in C++ code**
  - SV, MV tags (~CVS tags) or 'user tags' (~CVS branches)
  - *Goal: same performance optimization in all use cases*
    - Share a single C++ method to define the general SQL strategy (with internal switches for use-case-dependent SQL fragments)
    - So far each use case was optimized separately
- **Evaluate Oracle partitioning**
  - Goal: ease data management (long-term archiving)
    - Partitioned tables with partitioned (local) indexes
  - *Evaluate impact (benefits?) for performance too*
- **Performance for non-Oracle backends**
  - Using the same SQL is not always possible
    - MySQL performance is bad with subqueries
  - Lower priority

CERN

Summary

CERN IT Department

COOL strategy for optimizing Oracle performance

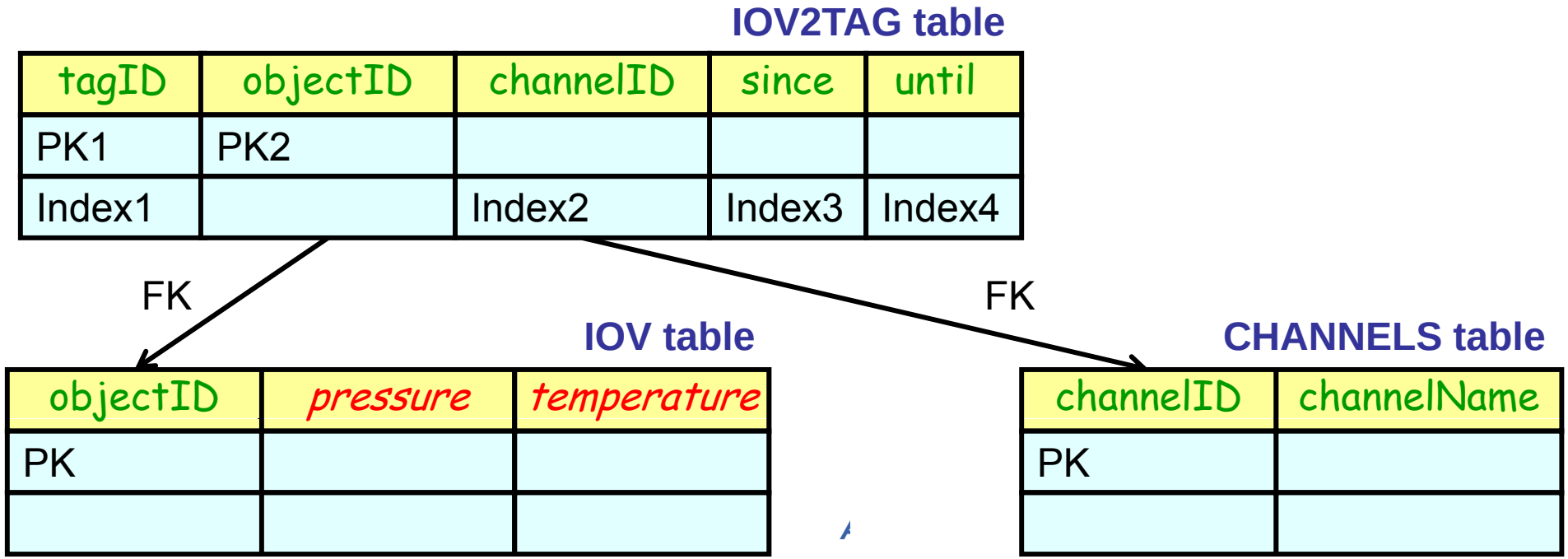
- Basic SQL optimization (fix indexes and joins)
- Execution plan instabilities (same SQL, different plans)
  - Observe (causes: unreliable statistics, bind variable peeking)
  - Analyze (**10053 trace files** and the BEGIN_OUTLINE block)
  - Fix (rewrite queries to please the Optimizer; then add **hints**)

Reserve slides

COOL – relational schema (simplified)

CERN IT Department

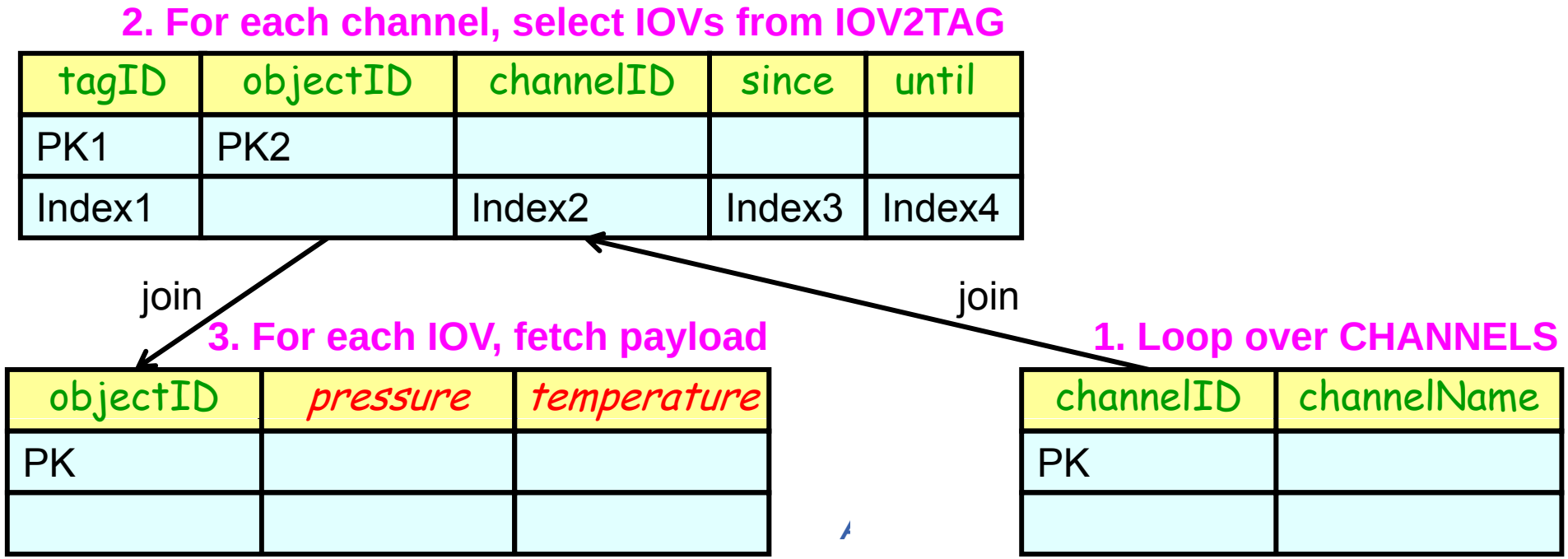
- **Metadata (green)**
  - System-controlled
  - Different sets of tables for different versioning modes (here: MV tags)
- ***Data payload (red)***
  - User-defined schema
  - Different sets of tables for different data channel categories ('folders')



COOL – test case: MV tag retrieval

CERN IT Department

- **Query: fetch all IOVs in [T1,T2] in tag PROD in all channels**
  - 1. Loop over all channels in table CHANNELS
  - 2. For each channel, select IOVs from table IOV2TAG
    - *In tag PROD in [T1, T2] – this is the most complex part of the query*
    - *Simplest (suboptimal): "(since ≤ T1< until) OR (T1 < since ≤ T2)"*
  - 3. For each selected IOV, fetch payload from table IOV



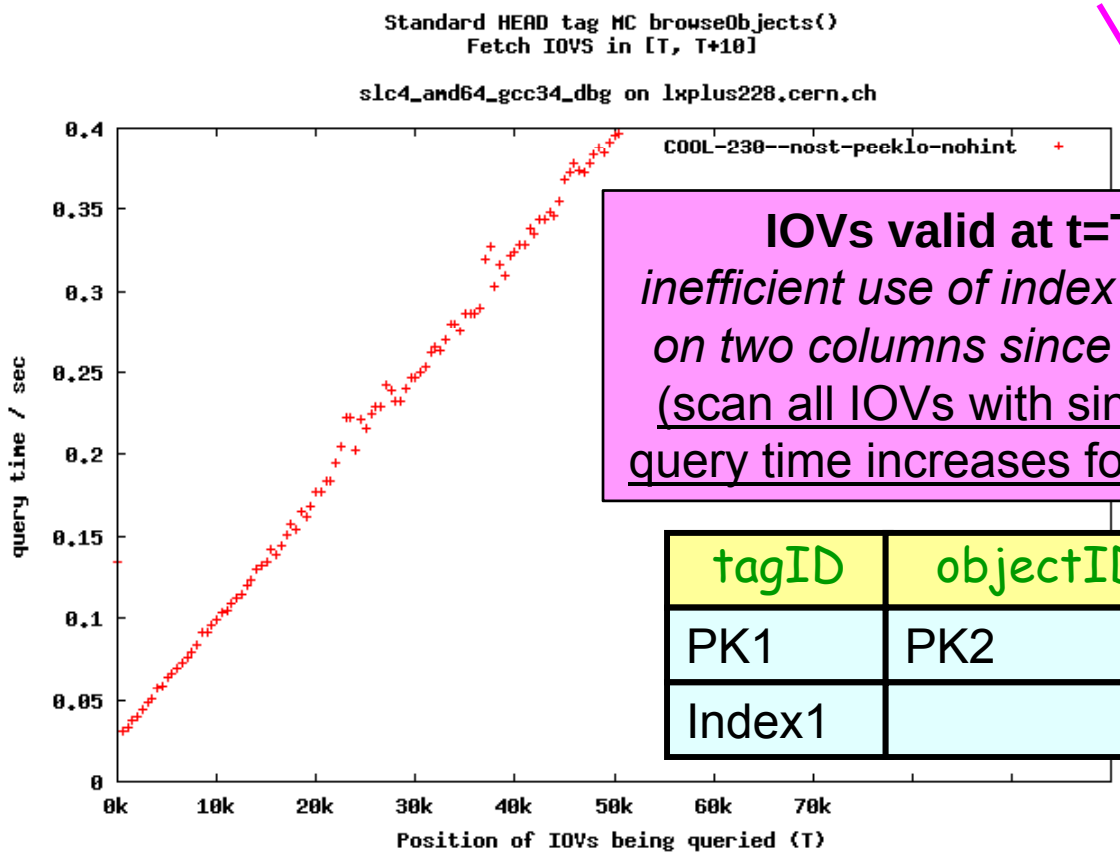
COOL – measuring performance

CERN IT Department

Is query time the same for all values of parameters T1, T2?

- *Not in releases ≤ COOL 2.3.0 : query time increases for more recent IOVs*
- *Simplest **(suboptimal):***

"tagId=PROD AND chId=C AND ((since ≤ T1< until) OR (T1 < since ≤ T2))"



IOVs valid at t=T1 :
inefficient use of index for query on two columns since and until
(scan all IOVs with since ≤ T1, query time increases for high T1)

IOVs valid from t>T1 to t ≤ T2 : efficient use of index

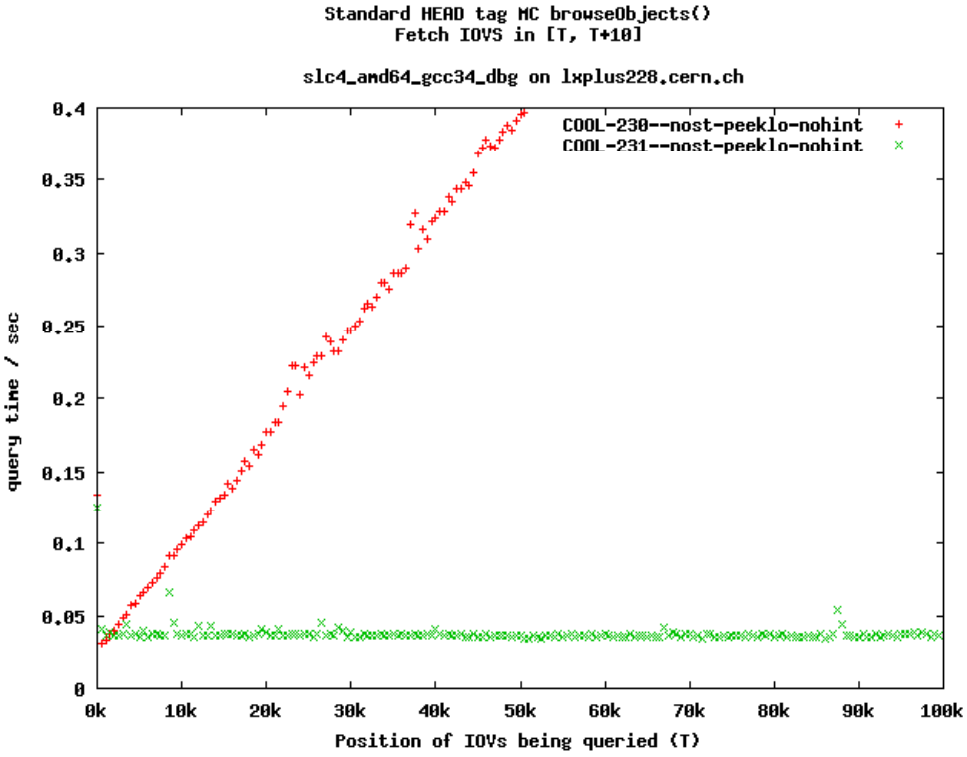
tagID	objectID	channelID	since	until
PK1	PK2			
Index1		Index2	Index3	Index4

Basic optimization – better SQL

CERN IT Department

In tag PROD, in each channel at most one IOV is valid at T1

- General definition of MV tags
  - This constraint is enforced in the C++ code, not in the database
- Find s_{MAX} = MAX(s) WHERE s<T1 in tag PROD and the loop channel
  - Accept (s = s_{MAX} OR T1 < s ≤ T2)
  - Remove 'OR' using 'COALESCE'




```
CORAL/RelationalPlugins/oracle    Debug Prepared statement : "SELECT /*+ QB_NAME
(MAIN)   */ COOL_I4."OBJECT_ID", COOL_I4."CHANNEL_ID", COOL_I4."IOV_SINCE", COOL
_I4."IOV_UNTIL", COOL_I4."USER_TAG_ID", COOL_I4."SYS_INSTIME", COOL_I4."LASTMOD_
DATE", COOL_I4."ORIGINAL_ID", COOL_I4."NEW_HEAD_ID", COOL_I4."I" FROM LCG_COOL."
N5820231_F0001_CHANNELS" COOL_C2, LCG_COOL."N5820231_F0001_IOV2TAG" COOL_I3, LCG
_COOL."N5820231_F0001_IOVS" COOL_I4 WHERE COOL_I3."TAG_ID"=:"tagid3" AND COOL_I3
."CHANNEL_ID"=COOL_C2."CHANNEL_ID" AND COOL_I3."IOV_SINCE">=COALESCE(( SELECT /*
+ QB_NAME(MAX1)  */ MAX(COOL_I1."IOV_SINCE") FROM LCG_COOL.N5820231_F0001_IOV2TA
G COOL_I1 WHERE COOL_I1."TAG_ID"=:"tagid1" AND COOL_I1."CHANNEL_ID"=COOL_C2."CHA
NNEL_ID" AND COOL_I1."IOV_SINCE"<=:"since1" ),:"sinc3s") AND COOL_I3."IOV_SINCE"
<=:"until3" AND COOL_I3."IOV_UNTIL">:"sinc3u" AND COOL_I4."OBJECT_ID"=COOL_I3."O
BJECT_ID" ORDER BY COOL_I3."CHANNEL_ID" ASC, COOL_I3."IOV_SINCE" ASC"
```