SQL Query Examples

Introduction

There are several facts you should be aware of when using the Polarion database, otherwise you may not get correct or expected results from your queries:

- Accessing the database from an external client requires that references to database tables include the schema name, which is POLARION in our case. So if you want to search in the WORKITEM table, you need to refer to it as **POLARION.WORKITEM**.

- Accessing the database from Polarion requires that database tables be referenced *without* the schema name, e.g. **WORKITEM**.

- If you want to search in a baseline via an external client, you need to connect to the historical database and reference tables, including schema name, so that the reference is composed of POLARION_B_ + revision number. For example: **POLARION_B_123.WORKITEM**. The particular baseline must exist in Polarion before you can search in it.

Joins

It is important to understand how the objects are identified in the database. For every object there are two columns: **C_PK** and **C_URI**.

- C_PK is the primary key, that also contains the information about the object version (revision)
- C_URI is the object ID, that does not contain the information about the object version.

To perform join queries you need to follow rules, to ensure that the queries work well both when you search the baseline and when you search the non-historical database.

- Tables that represent Polarion objects (i.e. their names *do not* start with CF, REL, or STRUCT) must be joined via the C_URI column, not by the C_PK column.
- Tables that *do not* represent Polarion prototypes (i.e. their names *do* start with CF, REL, or STRUCT) must always be joined by one C_PK column. Additional joins must be linked via the C_URI column.

Example:
Consider a test case work item that verifies a requirement. In Polarion data model, the link is an attribute of the test case work item and it is stored in the test case work item. The link then only points to the requirement that is show as the parent (outgoing) link on the test case detail. So for the SQL query to work correctly in the baselines, the test case work item must join the STRUCT_WORKITEM_LINKEDWORKITEMS by the C_PK column, while the outgoing relation to requirement must join via C_URI.

STRUCT_WORKITEM_LINKEDWORKITEMS.FK_URI_WORKITEM = REQUIREMENT.C_URI and
STRUCT_WORKITEM_LINKEDWORKITEMS.FK_P_WORKITEM = TESTCASE.C_PK

Table	Join with column	Example
REL_WORKITEM_USER_ASSIGNEE	FK_WORKITEM	Example: 4
REL_WORKITEM_CATEGORY_CATEGORIES	FK_WORKITEM	
REL_USER_WORKITEM_WATCHES	FK_USER	
REL_USER_WORKITEM_VOTES	FK_USER	
CF_WORKITEM	FK_WORKITEM	Example: 1
CF_TESTRUN	FK_TESTRUN	

STRUCT_*	FK_P_*	Example: 1

1. Requirements planned for "Release2" with implementing open defects

Queries all Work Items of type *requirement* in *MyProject* that have a target release value of *Release2* and that are implemented by some unresolved Work Item of type *defect*.

```
select
    WORKITEM.C_URI
from
    WORKITEM
    inner join PROJECT on WORKITEM.FK_URI_PROJECT = PROJECT.C_URI
    inner join CF_WORKITEM on CF_WORKITEM.FK_WORKITEM = WORKITEM.C_PK
where true
    and PROJECT.C_ID = 'myProject'
    and WORKITEM.C_TYPE = 'requirement'
    and CF_WORKITEM.C_NAME = 'targetRelease'
    and CF_WORKITEM.C_STRING_VALUE = 'Release2'
    and exists (
        select
            DEFECT.C_PK
        from
            WORKITEM DEFECT,
            STRUCT_WORKITEM_LINKEDWORKITEMS LINK
        where
            DEFECT.C_TYPE = 'defect' and
            LINK.C_ROLE = 'implements' and
            LINK.FK_URI_WORKITEM = WORKITEM.C_URI and
            LINK.FK_P_WORKITEM = DEFECT.C_PK and
            DEFECT.C_RESOLUTION IS NULL
    )
```

2. Requirements with linked test cases that failed in week 20

Queries all Work Items of type *requirement* in *MyProject* that are tested by some Work Item of type *testcase* which failed in the 20th week of year 2012.

```
select
    WORKITEM.C_URI
from
    WORKITEM
    inner join PROJECT on WORKITEM.FK_URI_PROJECT = PROJECT.C_URI
where true
    and PROJECT.C_ID = 'myProject'
    and WORKITEM.C_TYPE = 'requirement'
    and exists (
        select
            TESTCASE.C_PK
        from
            WORKITEM TESTCASE,
            TESTRUN TESTRUN,
            STRUCT_WORKITEM_LINKEDWORKITEMS LINK,
            STRUCT_TESTRUN_RECORDS TESTRECORD
        where
            LINK.FK_URI_WORKITEM = WORKITEM.C_URI AND
            LINK.FK_P_WORKITEM = TESTCASE.C_PK AND
            LINK.C_ROLE = 'tests' AND
            TESTCASE.C_TYPE = 'testcase' AND
            TESTRECORD.FK_URI_TESTCASE = TESTCASE.C_URI AND
            TESTRECORD.FK_P_TESTRUN = TESTRUN.C_PK AND
            TESTRECORD.C_RESULT = 'failed' AND
            TESTRECORD.C_EXECUTED > '2012-05-14 00:00:00' AND
            TESTRECORD.C_EXECUTED < '2012-05-20 00:00:00'
    )
```

3. Sum of time spent for tasks planned in "Iteration108"

Returns a sum of Time Spent values for all tasks that are assigned to Time Point *Iteration108*.

Info: This example can be executed only via an external client!

```
SELECT
    SUM(TASK.C_TIMESPENT)
FROM
    POLARION.WORKITEM TASK,
    POLARION.PROJECT PROJECT,
    POLARION.TIMEPOINT TIMEPOINT
WHERE
    TASK.FK_URI_PROJECT = PROJECT.C_URI AND
    PROJECT.C_ID = 'MyProject' AND
    TASK.C_TYPE = 'task' AND
    TASK.FK_URI_TIMEPOINT = TIMEPOINT.C_URI AND
    TIMEPOINT.C_ID = 'Iteration108'
```

4. Tasks assigned to "rProject" with "must_have" severity

Returns all Work Items of type *task* in *MyProject* that are assigned to *rProject* and that have *must_have* severity.

```
select
    WORKITEM.C_URI
from
    WORKITEM
    inner join PROJECT on WORKITEM.FK_URI_PROJECT = PROJECT.C_URI
    inner join REL_WORKITEM_USER_ASSIGNEE on WORKITEM.C_PK = REL_WORKITEM_USER_ASSIGNEE.FK_WORKITEM
    inner join USER on REL_WORKITEM_USER_ASSIGNEE.FK_URI_USER = USER.C_URI
where true
    and PROJECT.C_ID = 'drivepilot'
    and WORKITEM.C_TYPE = 'task'
    and WORKITEM.C_SEVERITY = 'must_have'
    and USER.C_ID = 'rProject'
```

Note: the table "USER" was renamed for PostgreSQL to "T_USER", so please adjust this example query accordingly, for running against PostgreSQL for Polarion. Use "T_USER", not "USER" when referring to the table.

5. Combining Lucene query with SQL query

Returns all Work Items of type *requirement* in *Playground* that has linked (role *tests*) at least one test case (type *testcase*).

```
select WORKITEM.C_URI
from WORKITEM
inner join LUCENE_QUERY('WorkItem', 'project.id:playground AND type:requirement', 'id') REQUIREMENT
        on WORKITEM.C_PK=REQUIREMENT.C_PK
where true
and exists (
        select
            TEST.C_PK
        from
            WORKITEM TEST,
            STRUCT_WORKITEM_LINKEDWORKITEMS LINK
        where
            LINK.FK_WORKITEM = REQUIREMENT.C_PK and
            LINK.FK_P_WORKITEM = TEST.C_PK and
            LINK.C_ROLE = 'tests' and
            TEST.C_TYPE = 'testcase'
    )
```

6. Using custom fields in SQL Query

Returns all Work Items of type *testcase* in *playground* project that are planned for sprint (custom field *plannedForSprint* is *true*) and duration (custom field *duration*) of WI is between 1 - 2 hours

```
select WORKITEM.C_URI
from WORKITEM
inner join PROJECT on PROJECT.C_URI = WORKITEM.FK_URI_PROJECT
inner join CF_WORKITEM CF1 on CF1.FK_WORKITEM = WORKITEM.C_PK
inner join CF_WORKITEM CF2 on CF2.FK_WORKITEM = WORKITEM.C_PK
where true
and PROJECT.C_ID = 'drivepilot'
and WORKITEM.C_TYPE = 'testcase'
and CF1.C_NAME = 'plannedForSprint'
and CF1.C_BOOLEAN_VALUE IS TRUE
and CF2.C_NAME = 'duration'
and CF2.C_DURATIONTIME_VALUE BETWEEN 1 AND 2
```

7. Distinct values in SQL Query

Collect all *System requirements* from *Drive Pilot* project that are covered by some *Test case* with linked *Issue*.

Keyword *group by* was used instead of *distinct* keyword.

```
select WORKITEM.C_URI
from WORKITEM
inner join PROJECT on PROJECT.C_URI = WORKITEM.FK_URI_PROJECT
inner join STRUCT_WORKITEM_LINKEDWORKITEMS LINKTEST on LINKTEST.FK_URI_WORKITEM = WORKITEM.C_URI
inner join WORKITEM TEST on TEST.C_URI = LINKTEST.FK_URI_P_WORKITEM
inner join STRUCT_WORKITEM_LINKEDWORKITEMS LINKISSUE on LINKISSUE.FK_URI_WORKITEM = TEST.C_URI
inner join WORKITEM ISSUE on ISSUE.C_URI = LINKISSUE.FK_URI_P_WORKITEM
where true
and PROJECT.C_ID = 'drivepilot'
and WORKITEM.C_TYPE = 'systemRequirement'
AND LINKTEST.C_ROLE = 'verifies'
AND ISSUE.C_TYPE = 'issue'
GROUP BY WORKITEM.C_URI
```

8. SQL query using "Like" or "Similar to" with a regular expression

Collect all Work Items with *Lucene* in their Description

```
select item.C_URI
from WORKITEM item
where item.C_DESCRIPTION
like '%Lucene%'
```