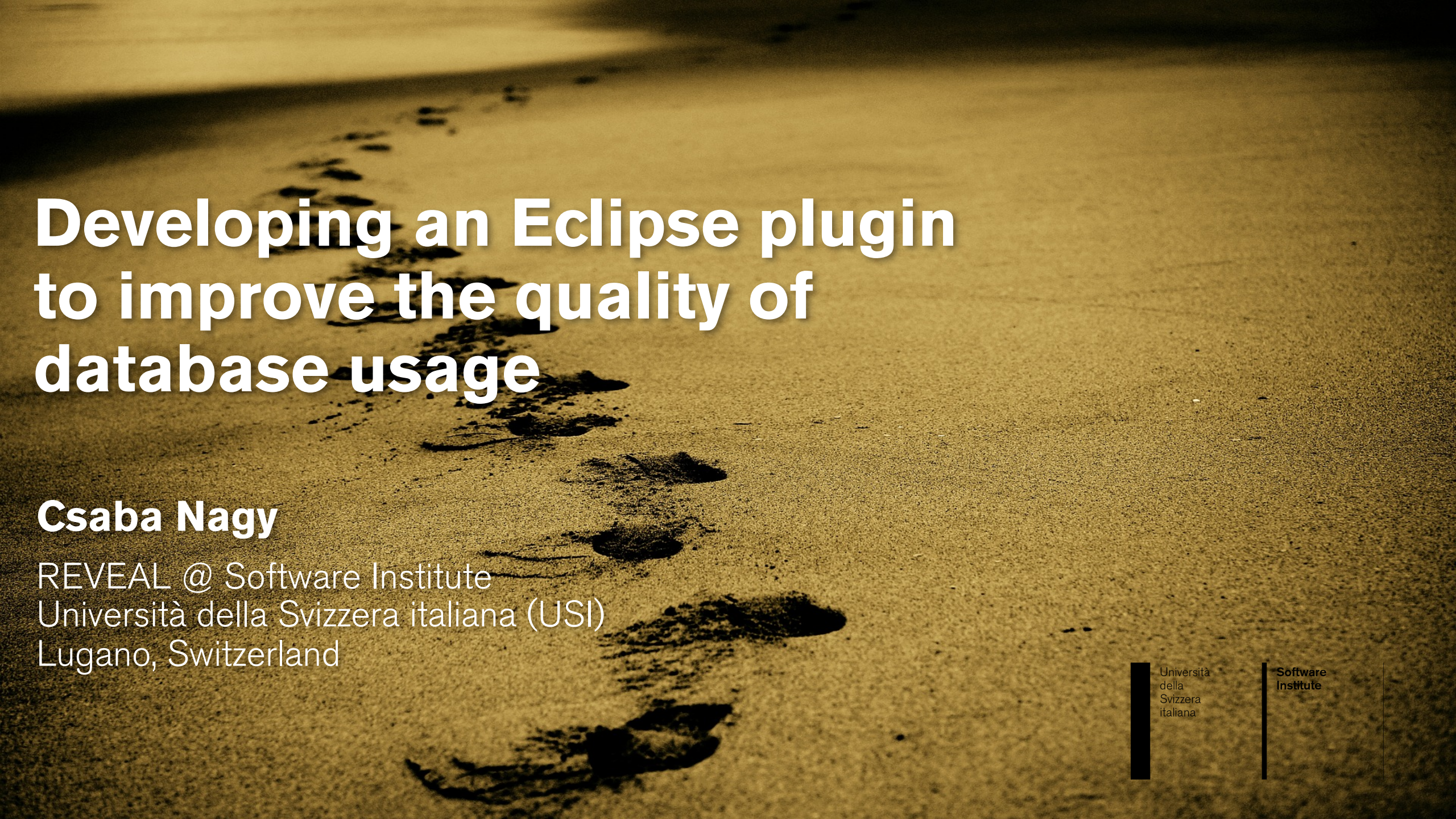


Developing an Eclipse plugin to improve the quality of database usage

Csaba Nagy

REVEAL @ Software Institute
Università della Svizzera italiana (USI)
Lugano, Switzerland

Università della Svizzera italiana

Software Institute


R E V E A L
USI
Università della Svizzera italiana
Software Institute
UNIVERSITÉ DE NAMUR
precise

What is wrong with this query?

```
SELECT a, *
FROM t1
JOIN t2 ON a = b
WHERE a <> NULL;
```

What is wrong with this query?

```
SELECT t1.a, *
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <> NULL;
```

What is wrong with this query?

```
SELECT t1.a, *
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <> IS NOT NULL;
```

What is wrong with this query?


```
SELECT t1.a, *
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <> IS NOT NULL;
```

```
You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to
use near '* from t1' at line 1
```

What is wrong with this query?


```
SELECT t1.a, * *, t1.a
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <=> IS NOT NULL;
```

What is wrong with this query?

```
          t1.a, t2.b, t2.c, …
SELECT ~~t1.a, * *, t1.a,~~
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <> IS NOT NULL;
```

Now can you find the same mistakes?

```
public static ResultSet queryTable(Connection con, String tab1,
String tab2, int filter) throws SQLException {
    String criteria = tab1 + ".a <> " +
                      (filter>0 ? Integer.toString(filter) : “NULL");
    String query = "SELECT a, * " +
                   " FROM " + tab1 +
                   " JOIN " + tab1 +
                   " ON " + tab1 + ".a= " + tab2 + ".b" +
                   " WHERE " + criteria;
    Statement stmt = con.createStatement();
    return stmt.executeQuery(query);
}
```

What if the query is like this?

```
select billingser0_.billingservice_no as billings1_373_,
billingser0_.anaesthesia as anaesthe2_373_,
billingser0_.billingservice_date as billings3_373_,
billingser0_.description as descript4_373_,
billingser0_.displaystyle as displays5_373_, billingser0_.gstFlag as
gstFlag373_, billingser0_.percentage as percentage373_,
billingser0_.region as region373_, billingser0_.service_code as
service9_373_, billingser0_.service_compositecode as service10_373_,
billingser0_.sliFlag as sliFlag373_, billingser0_.specialty as
specialty373_, billingser0_.termination_date as termina13_373_,
billingser0_.value as value373_ from billingservice billingser0_
where billingser0_.service_code='A001A' and
billingser0_.billingservice_date=(select
MAX(billingser1_.billingservice_date) from billingservice
billingser1_ where billingser1_.billingservice_date<>NULL and
billingser1_.service_code='A001A');
```

What if the query is like this?

```
select billingser0_.billingservice_no as billings1_373_,
billingser0_.anaesthesia as anaesthe2_373_,
billingser0_.billingservice_date as billings3_373_,
billingser0_.description as descript4_373_,
billingser0_.displaystyle as displays5_373_, billingser0_.gstFlag as
gstFlag373_, billingser0_.percentage as percentage373_,
billingser0_.region as region373_, billingser0_.service_code as
service9_373_, billingser0_.service_compositecode as service10_373_,
billingser0_.sliFlag as sliFlag373_, billingser0_.specialty as
specialty373_, billingser0_.termination_date as termina13_373_,
billingser0_.value as value373_ from billingservice billingser0_
where billingser0_.service_code='A001A' and
billingser0_.billingservice_date=(select
MAX(billingser1_.billingservice_date) from billingservice
billingser1_ where billingser1_.billingservice_date<>NULL and
billingser1_.service_code='A001A');
```

… or this?

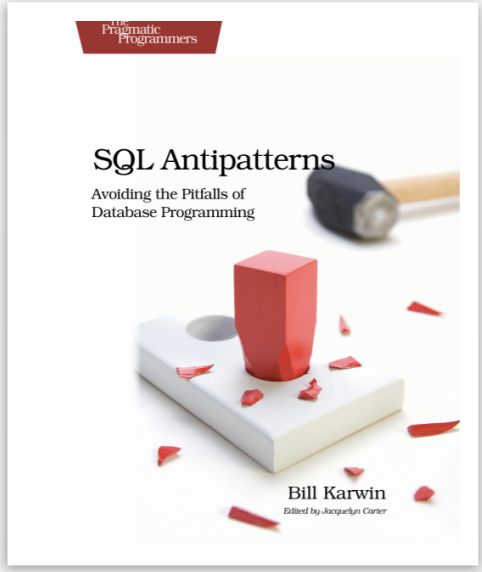
```
select appointmen0_.appointment_no as appointm1_89_0_, demographi1_.demographic_no as demograp1_27_1_,
appointmen0_.appointment_date as appointm2_89_0_, appointmen0_.billing as billing89_0_, appointmen0_.bookingSource as
bookingS4_89_0_, appointmen0_.createdatetime as createda5_89_0_, appointmen0_.creator as creator89_0_,
appointmen0_.creatorSecurityId as creatorS7_89_0_, appointmen0_.demographic_no as demograp8_89_0_, appointmen0_.end_time as
end9_89_0_, appointmen0_.imported_status as imported10_89_0_, appointmen0_.lastupdateuser as lastupd11_89_0_,
appointmen0_.location as location89_0_, appointmen0_.name as name89_0_, appointmen0_.notes as notes89_0_,
appointmen0_.program_id as program15_89_0_, appointmen0_.provider_no as provider16_89_0_, appointmen0_.reason as
reason89_0_, appointmen0_.reasonCode as reasonCode89_0_, appointmen0_.remarks as remarks89_0_, appointmen0_.resources as
resources89_0_, appointmen0_.start_time as start21_89_0_, appointmen0_.status as status89_0_, appointmen0_.style as
style89_0_, appointmen0_.type as type89_0_, appointmen0_.updatedatetime as updated25_89_0_, appointmen0_.urgency as
urgency89_0_, demographi1_.title as title27_1_, demographi1_.first_name as first3_27_1_, demographi1_.last_name as
last4_27_1_, demographi1_.sex as sex27_1_, demographi1_.month_of_birth as month6_27_1_, demographi1_.date_of_birth as
date7_27_1_, demographi1_.year_of_birth as year8_27_1_, demographi1_.address as address27_1_, demographi1_.city as
city27_1_, demographi1_.province as province27_1_, demographi1_.postal as postal27_1_, demographi1_.email as email27_1_,
demographi1_.phone as phone27_1_, demographi1_.phone2 as phone15_27_1_, demographi1_.myOscarUserName as myOscar16_27_1_,
demographi1_.hin as hin27_1_, demographi1_.ver as ver27_1_, demographi1_.hc_type as hc19_27_1_, demographi1_.hc_renew_date
as hc20_27_1_, demographi1_.roster_status as roster21_27_1_, demographi1_.patient_status as patient22_27_1_,
demographi1_.patient_status_date as patient23_27_1_, demographi1_.date_joined as date24_27_1_, demographi1_.chart_no as
chart25_27_1_, demographi1_.provider_no as provider26_27_1_, demographi1_.end_date as end27_27_1_, demographi1_.eff_date as
eff28_27_1_, demographi1_.roster_date as roster29_27_1_, demographi1_.roster_termination_date as roster30_27_1_,
demographi1_.roster_termination_reason as roster31_27_1_, demographi1_.pcn_indicator as pcn32_27_1_,
demographi1_.family_doctor as family33_27_1_, demographi1_.alias as alias27_1_, demographi1_.previousAddress as
previou35_27_1_, demographi1_.children as children27_1_, demographi1_.sourceOfIncome as sourceO37_27_1_,
demographi1_.citizenship as citizen38_27_1_, demographi1_.sin as sin27_1_, demographi1_.anonymous as anonymous27_1_,
demographi1_.spoken_lang as spoken41_27_1_, demographi1_.official_lang as official42_27_1_, demographi1_.lastUpdateUser as
lastUpd43_27_1_, demographi1_.lastUpdateDate as lastUpd44_27_1_, demographi1_.newsletter as newsletter27_1_,
demographi1_.country_of_origin as country46_27_1_, (select lst.description from lst_gender lst where
lst.code=demographi1_.sex) as formula21_1_, (select d.merged_to from demographic_merged d where d.deleted = 0 and
d.demographic_no = demographi1_.demographic_no) as formula22_1_, (select count(*) from admission a where
a.client_id=demographi1_.demographic_no and a.admission_status='current' and a.program_id in (select p.id from program p
where p.type<>NULL )) as formula23_1_, (select count(*) from health_safety h where
h.demographic_no=demographi1_.demographic_no) as formula24_1_ from appointment appointmen0_, demographic demographi1_ where
appointmen0_.demographic_no=demographi1_.demographic_no and demographi1_.hin<>'' and
appointmen0_.appointment_date>='2014-10-23' and appointmen0_.appointment_date<='2014-10-23' and
(upper(demographi1_.province)='ONTARIO' or demographi1_.province='ON') group by demographi1_.demographic_no order by
demographi1_.last_name;
```

... or this?

```
select appointmen0_.appointment_no as appointm1_89_0_, demographi1_.demographic_no as demograp1_27_1_,
appointmen0_.appointment_date as appointm2_89_0_, appointmen0_.billing as billing89_0_, appointmen0_.bookingSource as
bookingS4_89_0_, appointmen0_.createdatetime as createda5_89_0_, appointmen0_.creator as creator89_0_,
appointmen0_.creatorSecurityId as creatorS7_89_0_, appointmen0_.demographic_no as demograp8_89_0_, appointmen0_.end_time as
end9_89_0_, appointmen0_.imported_status as imported10_89_0_, appointmen0_.lastupdateuser as lastupd11_89_0_,
appointmen0_.location as location89_0_, appointmen0_.name as name89_0_, appointmen0_.notes as notes89_0_,
appointmen0_.program_id as program15_89_0_, appointmen0_.provider_no as provider16_89_0_, appointmen0_.reason as
reason89_0_, appointmen0_.reasonCode as reasonCode89_0_, appointmen0_.remarks as remarks89_0_, appointmen0_.resources as
resources89_0_, appointmen0_.start_time as start21_89_0_, appointmen0_.status as status89_0_, appointmen0_.style as
style89_0_, appointmen0_.type as type89_0_, appointmen0_.updatedatetime as updated25_89_0_, appointmen0_.urgency as
urgency89_0_, demographi1_.title as title27_1_, demographi1_.first_name as first3_27_1_, demographi1_.last_name as
last4_27_1_, demographi1_.sex as sex27_1_, demographi1_.month_of_birth as month6_27_1_, demographi1_.date_of_birth as
date7_27_1_, demographi1_.year_of_birth as year8_27_1_, demographi1_.address as address27_1_, demographi1_.city as
city27_1_, demographi1_.province as province27_1_, demographi1_.postal as postal27_1_, demographi1_.email as email27_1_,
demographi1_.phone as phone27_1_, demographi1_.phone2 as phone15_27_1_, demographi1_.myOscarUserName as myOscar16_27_1_,
demographi1_.hin as hin27_1_, demographi1_.ver as ver27_1_, demographi1_.hc_type as hc19_27_1_, demographi1_.hc_renew_date
as hc20_27_1_, demographi1_.roster_status as roster21_27_1_, demographi1_.patient_status as patient22_27_1_,
demographi1_.patient_status_date as patient23_27_1_, demographi1_.date_joined as date24_27_1_, demographi1_.chart_no as
chart25_27_1_, demographi1_.provider_no as provider26_27_1_, demographi1_.end_date as end27_27_1_, demographi1_.eff_date as
eff28_27_1_, demographi1_.roster_date as roster29_27_1_, demographi1_.roster_termination_date as roster30_27_1_,
demographi1_.termination_reason as roster31_27_1_, demographi1_.pcn_indicator as pcn32_27_1_,
demographi1_.spoken_lang as family33_27_1_, demographi1_.alias as alias27_1_, demographi1_.previousAddress as
lastUpd43_27_1_, demographi1_.children as children27_1_, demographi1_.sourceOfIncome as sourceOf37_27_1_,
demographi1_.country_of_citizen as citizen38_27_1_, demographi1_.sin as sin27_1_, demographi1_.anonymous as anonymous27_1_,
lst.code=demographi1_.spoken_lang and lst.lang='en' and spoken41_27_1_, demographi1_.official_lang as official42_27_1_, demographi1_.lastUpdateUser as
d.demographic_no = demographi1_.lastUpdateDate as lastUpd44_27_1_, demographi1_.newsletter as newsletter27_1_,
a.client_id=demographi1_.demographic_no and country_of_origin as country46_27_1_, (select lst.description from lst_gender lst where
where p.type<>NULL )) as formula21_1_, (select d.merged_to from demographic_merged d where d.deleted = 0 and
h.demographic_no=demographi1_.demographic_no) as formula22_1_, (select count(*) from admission a where
where appointmen0_.demographic_no and a.admission_status='current' and a.program_id in (select p.id from program p
appointmen0_.appointment_date>='2014-10-23' and appointmen0_.appointment_date<='2014-10-23' and
upper(demographi1_.province)='ONTARIO' or demographi1_.province='ON') group by demographi1_.demographic_no order by
demographi1_.last_name, demographi1_.first_name
```

Goals

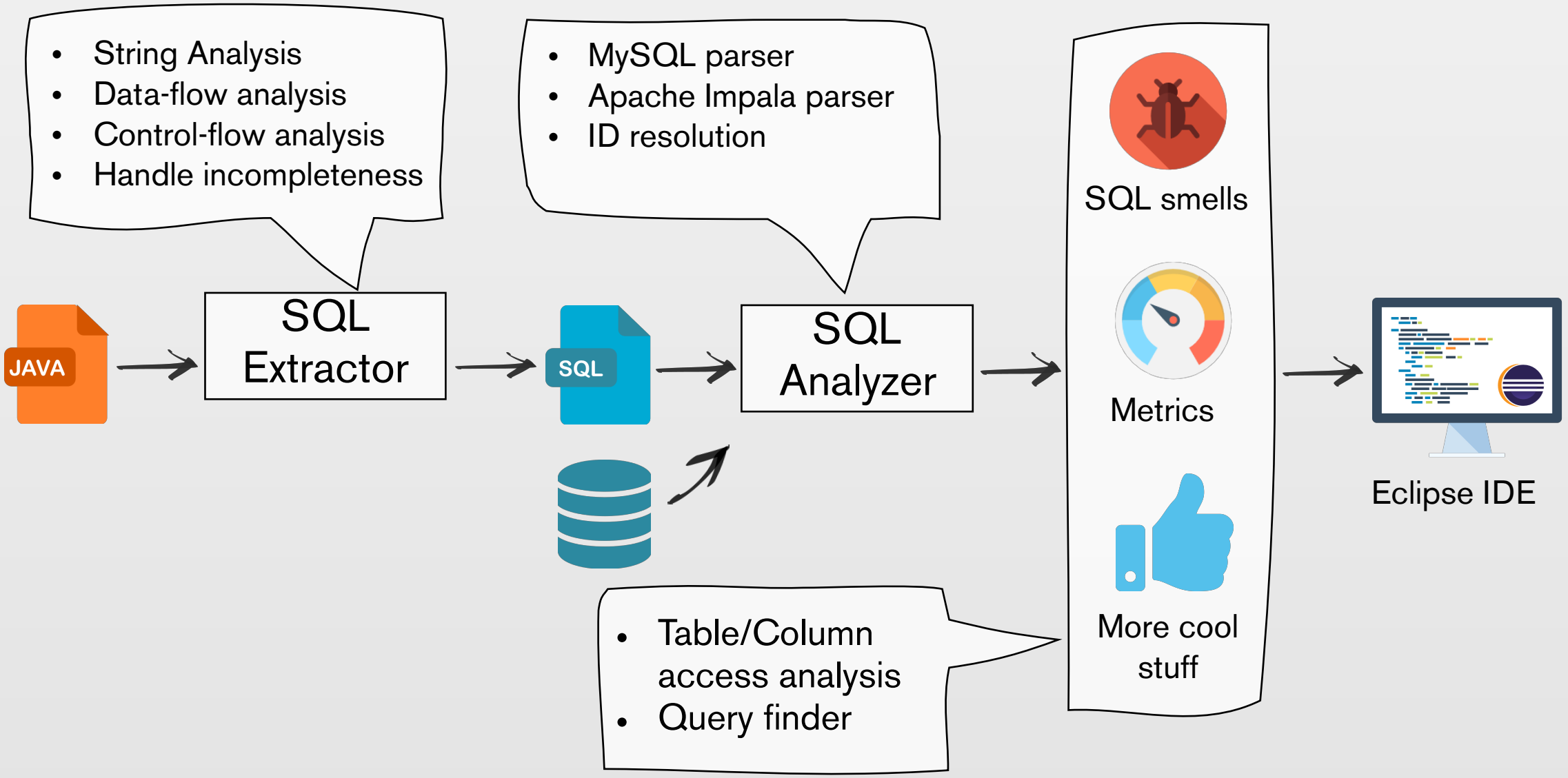
- To support working with SQL *embedded in Java*
- Avoid potential mistakes
- Write more efficient queries
- Help in maintenance tasks
- Refactoring
- Concept location



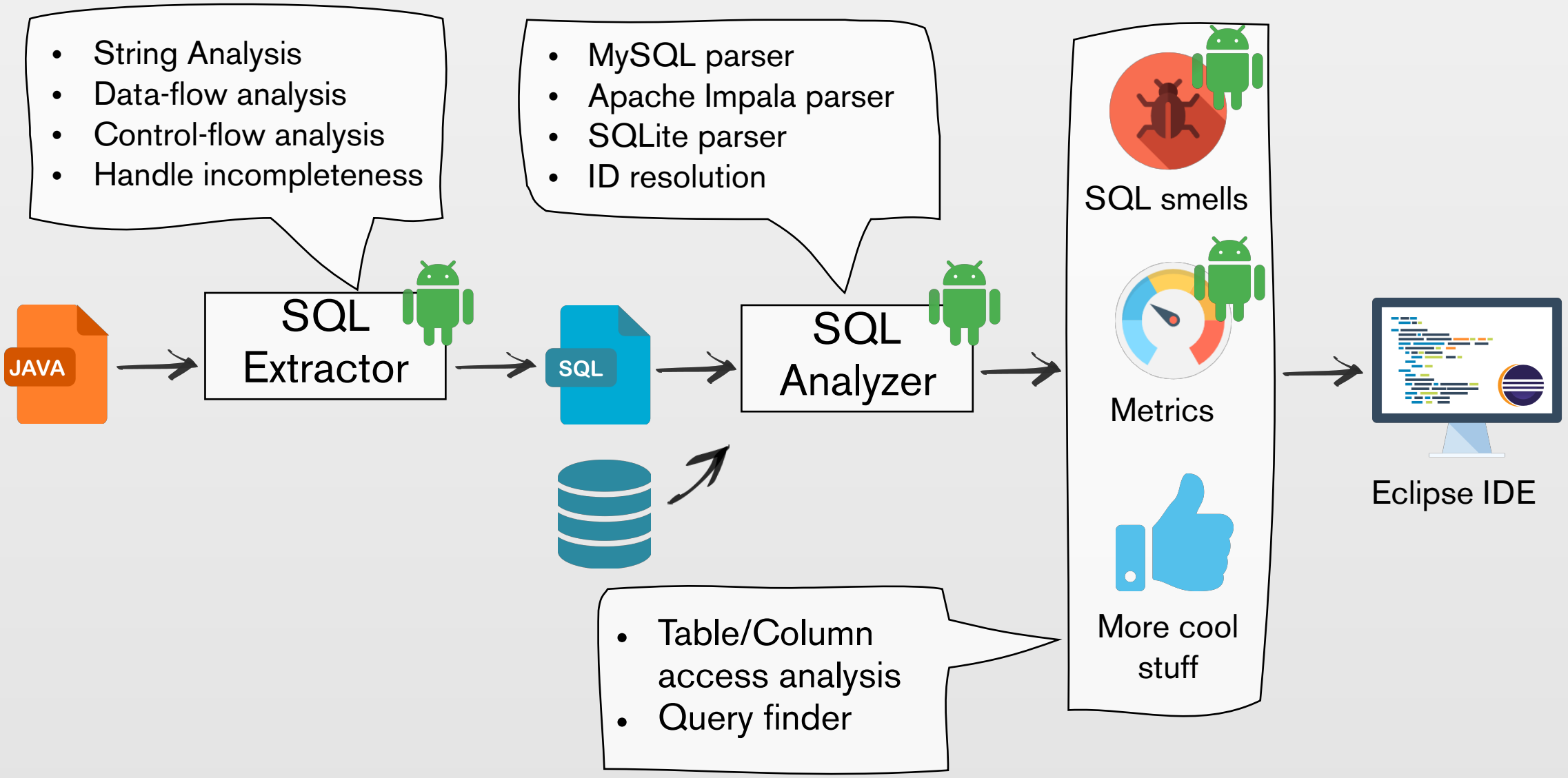
"SQL Antipatterns describes the most frequently made missteps I've seen people naively make while using SQL." – Bill Karwin

"SQL has its own particular habits that will alert the programmer to the need to refactor code." – Phil Factor @Redgate

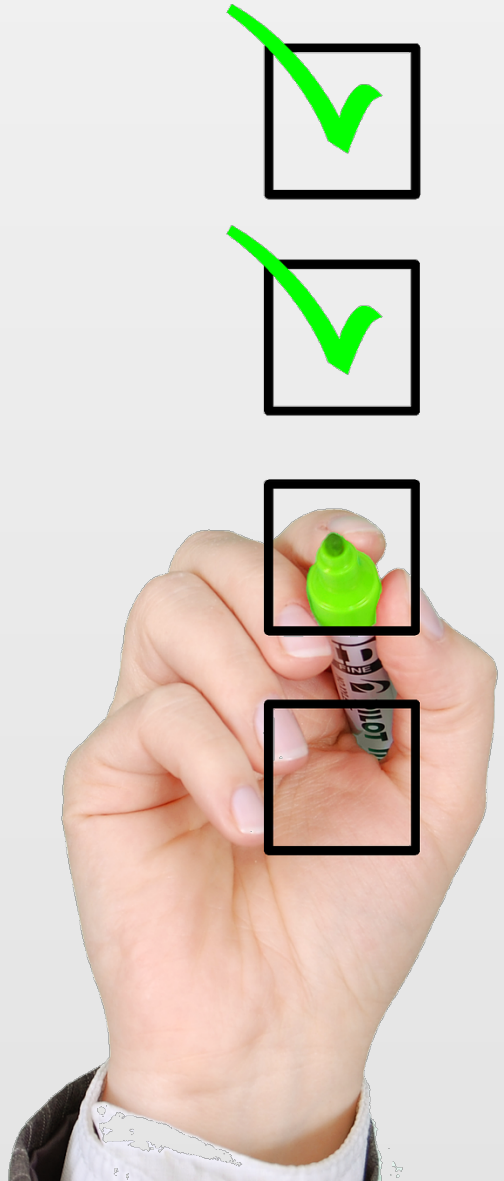
An Eclipse plugin from scratch



An Eclipse plugin from scratch

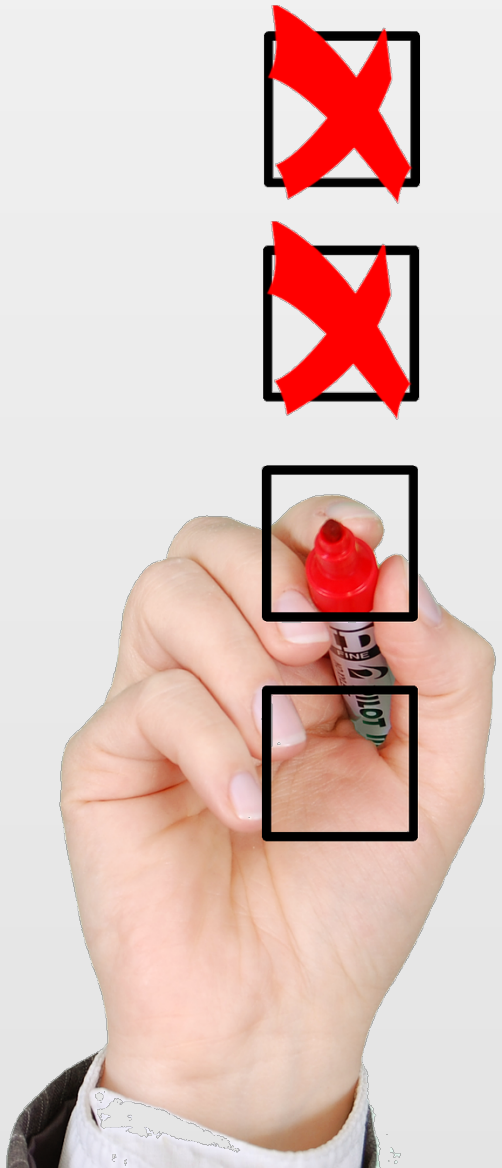


Eclipse JDT, because…



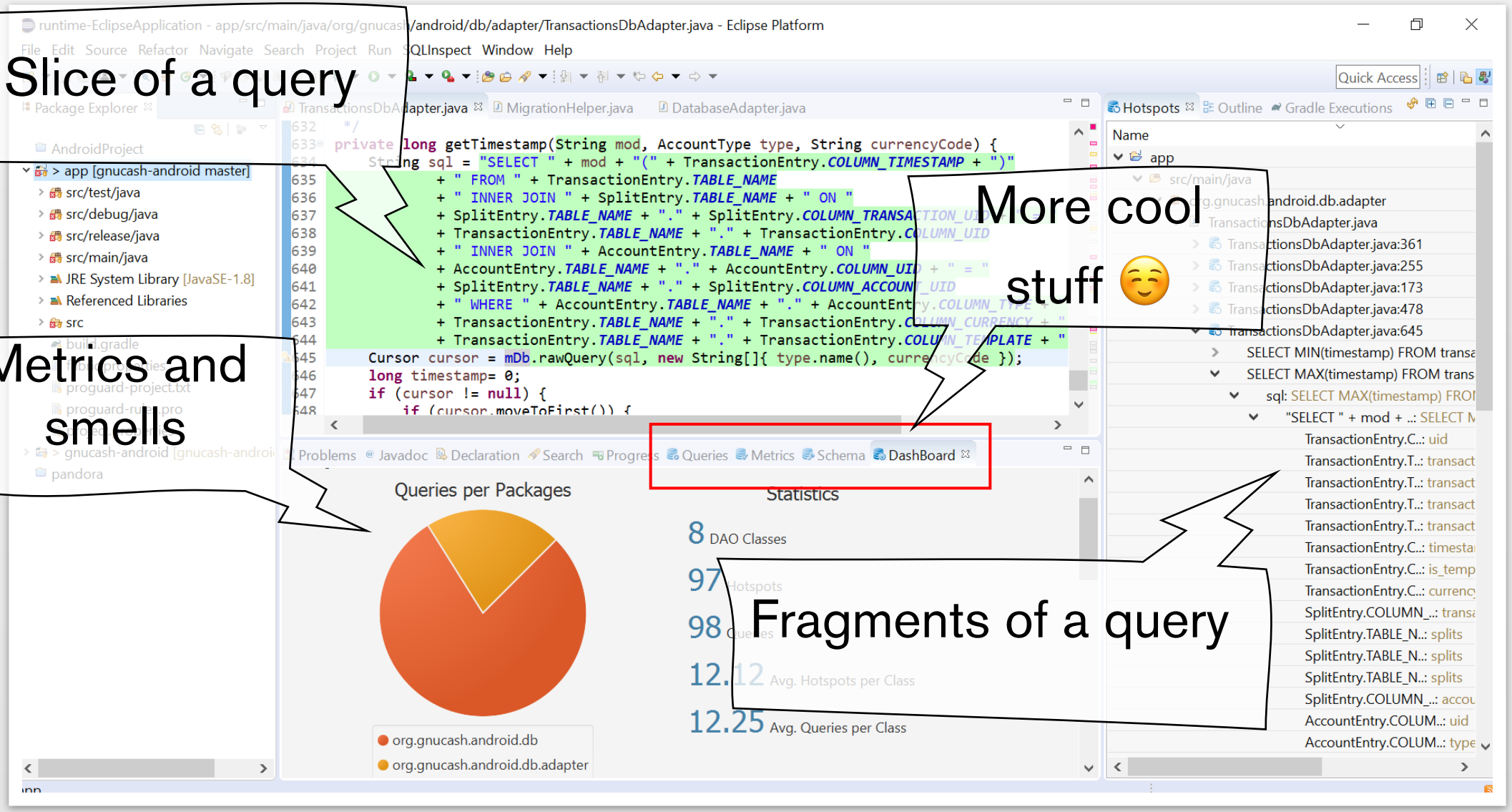
- Fault tolerance
  - Works with the AST in the editor
- Complete AST, comfortable API
- Handles the project as a whole
  - Possible inter-procedural analysis
- Basic analyses
  - Identifier resolution
- IDE integration
- Works for standalone application too
- Ready for code manipulation, refactorings

But...



- Lacks support for advanced static analysis
- Cool stuff are internal
  - `org.eclipse.jdt.internal.compiler.flow`
  - `org.eclipse.jdt.internal.corext.callhierarchy`
- No normalization
- No control-flow/data-flow analysis
  - Plugins implement their own
    - Early Security Vulnerability Detector - ESVD
    - Control Flow Graph Factory

DEMO


Slice of a query
Metrics and smells
More cool stuff
Fragments of a query
Queries per Packages
org.gnucash.android.db
org.gnucash.android.db.adapter
Statistics
8 DAO Classes
97 Hotspots
98 Queries
12.12 Avg. Hotspots per Class
12.25 Avg. Queries per Class

We analyzed 1000 top-rated Android projects on GitHub.

Found some interesting stuff and reported PRs.

Common Android mistakes (Project X)

```
static void updateSessionSearchIndex(SQLiteDatabase db) {
    db.execSQL("DELETE FROM " + Tables.SESSIONS_SEARCH);

    db.execSQL("INSERT INTO " + Qualified.SESSIONS_SEARCH
            + " SELECT s." + Sessions.SESSION_ID + ",("

            // Full text body
            + Sessions.SESSION_TITLE + "||'; '||"
            + Sessions.SESSION_ABSTRACT + "||'; '||"
            + "IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",' '),'')||'; '||"
            + "'')"

            + " FROM " + Tables.SESSIONS + " s "
            + " LEFT OUTER JOIN"

            // Subquery resulting in session_id, speaker_id, speaker_name
            + "(SELECT " + Sessions.SESSION_ID + "," + Qualified.SPEAKERS_SPEAKER_ID
            + "," + Speakers.SPEAKER_NAME
            + " FROM " + Tables.SESSIONS_SPEAKERS
            + " INNER JOIN " + Tables.SPEAKERS
            + " ON " + Qualified.SESSIONS_SPEAKERS_SPEAKER_ID + "="
            + Qualified.SPEAKERS_SPEAKER_ID
            + ") t"

            // Grand finale
            + " ON s." + Sessions.SESSION_ID + "=t." + Sessions.SESSION_ID
            + " GROUP BY s." + Sessions.SESSION_ID);
}
```

https://github.com/google/iosched

```
static void updateSessionSearchIndex(SQLiteDatabase db) {
    db.execSQL("DELETE FROM " + Tables.SESSIONS_SEARCH);

    db.execSQL("INSERT INTO " + Qualified.SESSIONS_SEARCH
            + " SELECT s." + Sessions.SESSION_ID + ",("

            // Full text body
            + Sessions.SESSION_TITLE + "||'; '||"
            + Sessions.SESSION_ABSTRACT + "||'; '||"
            + "IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",' '),'')||'; '||"
            + "''')"

            + " FROM " + Tables.SESSIONS + " s "
            + " FROM " + Tables.SESSIONS_SPEAKERS
            + " INNER JOIN " + Tables.SPEAKERS
            + " ON " + Qualified.SESSIONS_SPEAKERS_SPEAKER_ID + "="
            + Qualified.SPEAKERS_SPEAKER_ID
            + ") t"

            // Grand finale
            + " ON s." + Sessions.SESSION_ID + "=t." + Sessions.SESSION_ID
            + " GROUP BY s." + Sessions.SESSION_ID);
}
```

```
execSQL(String sql, Object[] bindArgs)
```

Execute a single SQL statement that is NOT a SELECT/INSERT/UPDATE/DELETE.

https://github.com/google/iosched

```
static void updateSessionSearchIndex(SQLiteDatabase db) {
    db.execSQL("DELETE FROM " + Tables.SESSIONS_SEARCH);

    db.execSQL("INSERT INTO " + Qualified.SESSIONS_SEARCH
            + " SELECT s." + Sessions.SESSION_ID + ",("

            // Full text body
            + Sessions.SESSION_TITLE + "||'; '||"
            + Sessions.SESSION_ABSTRACT + "||'; '||"
            + "IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",' '),'')||'; '||"
            + "''')"

            + " FROM " + Tables.SESSIONS + " s "
            + " LEFT OUTER JOIN"

            // Subquery resulting in session_id, speaker_id, speaker_name
            + "(SELECT " + Sessions.SESSION_ID + "," + Qualified.SPEAKERS_SPEAKER_ID
            + "," + Speakers.SPEAKER_NAME
            + " FROM " + Tables.SESSIONS_SPEAKERS
            + " INNER JOIN " + Tables.SPEAKERS
            + " ON " + Qualified.SESSIONS_SPEAKERS_SPEAKER_ID + "="
            + Qualified.SPEAKERS_SPEAKER_ID
            + ") t"

            // Grand finale
            + " ON s." + Sessions.SESSION_ID + "=t." + Sessions.SESSION_ID
            + " GROUP BY s." + Sessions.SESSION_ID);
}
```

https://github.com/google/iosched

```
static void updateSessionSearchIndex(SQLiteDatabase db) {
    db.execSQL("DELETE FROM " + Tables.SESSIONS_SEARCH);

    db.execSQL("INSERT INTO " + Qualified.SESSIONS_SEARCH
            + " SELECT s." + Sessions.SESSION_ID + ",("

            // Full text body
            + Sessions.SESSION_TITLE + "||'; '||"
            + Sessions.SESSION_ABSTRACT + "||'; '||"
            + "IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",' '),'')||'; '||"
            + "'')"

            // Grand finale
            + " GROUP BY s." + Sessions.SESSION_ID);
}
```

https://github.com/google/iosched

A result column which contains a column name that is not within an aggregate function and that does not appear in the GROUP BY clause (if one exists) is called a "bare" column.

SELECT a, b, sum(c) FROM tab1 GROUP BY a; ← b is missing!

...

The problem is that you usually do not know which input row is used to compute "b", and so in many cases the **value for "b" is undefined.**

https://sqlite.org/lang_select.html

Common Android mistakes (Project Y)

commented on May 25 • edited by

Hello,
I've spotted a misuse of multiple SQL statements in an execSQL(). In fact, after the first index, the others do not get created by SQLite like this. Please consider reviewing and fixing it.
Best,
Csaba

```
69      -    private static final String SQL_INDEX_ENTRIES =
70      -            NotificationTable.INDEX_CREATE_NOTIFICATION_ID +
71      -            NotificationTable.INDEX_CREATE_ANDROID_NOTIFICATION_ID +
72      -            NotificationTable.INDEX_CREATE_GROUP_ID +
73      -            NotificationTable.INDEX_CREATE_COLLAPSE_ID +
74      -            NotificationTable.INDEX_CREATE_CREATED_TIME;
        69  +    private static final String[] SQL_INDEX_ENTRIES = { NotificationTable.INDEX_CREATE_NOTIFICATION_ID,
        70  +            NotificationTable.INDEX_CREATE_ANDROID_NOTIFICATION_ID,
        71  +            NotificationTable.INDEX_CREATE_GROUP_ID,
        72  +            NotificationTable.INDEX_CREATE_COLLAPSE_ID,
        73  +            NotificationTable.INDEX_CREATE_CREATED_TIME };
```

```
116     115         @Override
117     116         public void onCreate(SQLiteDatabase db) {
118     117             db.execSQL(SQL_CREATE_ENTRIES);
119         -           db.execSQL(SQL_INDEX_ENTRIES);
        118 +           for (String ind : SQL_INDEX_ENTRIES) {
        119 +               db.execSQL(ind);
        120 +           }
120     121         }
```

commented on May 25 • edited ▾ Member

Thanks for the catch! For reference this the multiple statements per `execSQL` is noted below.
https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase#execsql_3

We will review our SQL statements for any other cases where this could be happening. Well will leave this PR open until then so feel free to add to this PR if you find any other cases.

Thanks!

commented 17 days ago Member

I checked our code base and was not able to find any other cases of this issue, just the index create which you fixed Merging this PR in. Thanks again!

“It is good to have an end to journey toward;
but it is the journey that matters, in the end.”
Ernest Hemingway

https://bitbucket.org/csnagy/sqlinspect