Developing an Eclipse plugin to improve the quality of database usage

Csaba Nagy
REVEAL @ Software Institute
Università della Svizzera italiana (USI)
Lugano, Switzerland
What is wrong with this query?

```sql
SELECT a, *
FROM t1
JOIN t2 ON a = b
WHERE a <> NULL;
```
What is wrong with this query?

```sql
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?

```
SELECT t1.a, t2.b, t2.c, ...
FROM t1.a
JOIN t2 ON t1.a = t2.b
WHERE t1.a <> IS NOT NULL;
```
Now can you find the same mistakes?

```java
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?

```sql
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?

```sql
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?

```sql
select appointment0.appointment_no as appointment1, 89_0, demographi1.demographic_no as demographi1_27_1,
appointmen0.appointment_date as appointment2 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._created as created89_0, appointmen0.demographic_no as demographi89_0, appointmen0._deleted as deleted89_0,
appointmen0._end_time as end9_89_0, appointmen0._imported as imported10_89_0, appointmen0._lastupdateuser as lastUpd43_27_1,
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._resources as resources89_0, appointmen0._style as style89_0,
appointmen0.type as type89_0, appointmen0._updatedatemetime as updatedatedate89_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.country_of_origin as country46_27_1, demographi1.demographi1.hin as hin27_1, demographi1.official_lang as official42_27_1,
demographi1.hc_code 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.patient_id as provider26_27_1, demographi1.end_date as end27_27_1, demographi1.ethnicity as ethnicity27_1,
demographi1.ethnicity as ethnicity27_1, demographi1.family_doctor as family33_27_1, demographi1.alias as alias27_1,
demographi1.alien as alien27_1, demographi1.age as age27_1, demographi1.is citizien38_27_1, demographi1.sin as sin27_1,
demographi1.anonymous as anonymous27_1, demographi1.spoken_lang as spokenlang27_1, demographi1.major_disease as major27_1,
demographi1.oficial_lang as oficial42_27_1, demographi1.lastupdateuser as lastUpd44_27_1, demographi1.newsletter as newsletter27_1,
demographi1.relationship as relationship28_27_1, demographi1.patient_status as patient29_27_1, demographi1.province as province27_1,
demographi1.date_of_birth as date30_27_1, demographi1.request as request31_27_1, demographi1.patient_status as patient32_27_1,
demographi1.label as label27_1, demographi1.character as character27_1, demographi1.gender as gender27_1, demographi1.patient_status as patient33_27_1,
demographi1.patient_status as patient34_27_1, demographi1.patient_status as patient35_27_1, demographi1.family_doctor as family36_27_1,
demographi1.family_doctor as family36_27_1, demographi1.patient_id as provider27_27_1, demographi1.patient_id as provider28_27_1,
demographi1.patient_id as provider29_27_1, demographi1.patient_id as provider30_27_1, demographi1.patient_id as provider31_27_1,
demographi1.patient_id as provider32_27_1, demographi1.patient_id as provider33_27_1, demographi1.patient_id as provider34_27_1,
demographi1.patient_id as provider35_27_1, demographi1.patient_id as provider36_27_1, demographi1.patient_id as provider37_27_1,
demographi1.patient_id as provider38_27_1, demographi1.patient_id as provider39_27_1, demographi1.patient_id as provider40_27_1,
demographi1.patient_id as provider41_27_1, demographi1.patient_id as provider42_27_1, demographi1.patient_id as provider43_27_1,
demographi1.patient_id as provider44_27_1, demographi1.patient_id as provider45_27_1, demographi1.patient_id as provider46_27_1,
demographi1.patient_id as provider47_27_1, demographi1.patient_id as provider48_27_1, demographi1.patient_id as provider49_27_1,
demographi1.patient_id as provider50_27_1, demographi1.patient_id as provider51_27_1, demographi1.patient_id as provider52_27_1,
demographi1.patient_id as provider53_27_1, demographi1.patient_id as provider54_27_1, demographi1.patient_id as provider55_27_1,
demographi1.patient_id as provider56_27_1, demographi1.patient_id as provider57_27_1, demographi1.patient_id as provider58_27_1,
demographi1.patient_id as provider59_27_1, demographi1.patient_id as provider60_27_1, demographi1.patient_id as provider61_27_1,
demographi1.patient_id as provider62_27_1, demographi1.patient_id as provider63_27_1, demographi1.patient_id as provider64_27_1,
demographi1.patient_id as provider65_27_1, demographi1.patient_id as provider66_27_1, demographi1.patient_id as provider67_27_1,
demographi1.patient_id as provider68_27_1, demographi1.patient_id as provider69_27_1, demographi1.patient_id as provider70_27_1,
...
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

- String Analysis
- Data-flow analysis
- Control-flow analysis
- Handle incompleteness

SQL Extractor

- MySQL parser
- Apache Impala parser
- ID resolution

SQL Analyzer

- Table/Column access analysis
- Query finder

SQL smells

Metrics

Eclipse IDE

More cool stuff
An Eclipse plugin from scratch

- String Analysis
- Data-flow analysis
- Control-flow analysis
- Handle incompleteness

SQL Extractor

- MySQL parser
- Apache Impala parser
- SQLite parser
- ID resolution

SQL Analyzer

- Table/Column access analysis
- Query finder

SQL smells

Metrics

More cool stuff

Eclipse IDE
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
Slice of a query

Metrics and smells

More cool stuff 😊

Fragments of a query

DEMO

Fragments of a query

Metrics and smells

Slice of a query

More cool stuff 😊

Fragments of a query

DEMO
We analyzed 1000 top-rated Android projects on GitHub.

Found some interesting stuff and reported PRs.
```java
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
                   + " FROM " + Tables.SESSIONS + " s "
                   + " LEFT OUTER JOIN"
                   + " // Subquery resulting in session_id, speaker_id, speaker_name
                   + "(SELECT " + Sessions.SESSION_ID + "," + Qualified.SPEAKERS_SPEAKER_ID
                   + " 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
```java
static void updateSessionCom(hIndex SQLiteDatabase db) {
    db.execSQL("DELETE FROM " + Qualified.SESSIONS_SEARCH);
    db.execSQL("INSERT INTO " + Qualified.SESSIONS_SEARCH
        + Sessions.SESSION_ID + ","
    );

    // Full text body
    + Sessions.SESSION_TITLE + ":\""\"\";
    + Sessions.SESSION_ABSTRACT + ":\""\"\";
    + IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",\"\"),\"\")\"
    + ");";
    + ");";
    + FROM " + Table.SESSIONS + ";";
    + " INNER JOIN " + Tables.SPEAKERS
    + " ON " + Qualified.SESSIONS_SPEAKERS_SPEAKER_ID + "="
    + Qualified.SPEAKERS_SPEAKER_ID
    + ");";

    // 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 + ",(" + Sessions.SESSION_TITLE + "||'',''||" + Sessions.SESSION_ABSTRACT + "||'',''||" + "IFNULL(GROUP_CONCAT(t." + Speakers.SPEAKER_NAME + ",',''))'',''||" + "')" + " FROM " + Tables.SESSIONS + " s " + " LEFT OUTER JOIN " + Sessions.SESSION_ID + " = " + Qualified.SESSIONS_SPEAKERS + " ON " + Qualified.SESSIONS_SPEAKERS_SPEAKER_ID + " = " + Sessions.SESSION_ID + " GROUP BY s." + Sessions.SESSION_ID + ");
}
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;
```

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
https://github.com/google/iosched
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
private static final String SQL_INDEX_ENTRIES =
- NotificationTable.INDEX_CREATE_NOTIFICATION_ID +
- NotificationTable.INDEX_CREATE_ANDROID_NOTIFICATION_ID +
- NotificationTable.INDEX_CREATE_GROUP_ID +
- NotificationTable.INDEX_CREATE_COLLAPSE_ID +
- NotificationTable.INDEX_CREATE_CREATED_TIME;

private static final String[] SQL_INDEX_ENTRIES = {
    NotificationTable.INDEX_CREATE_NOTIFICATION_ID,
    NotificationTable.INDEX_CREATE_ANDROID_NOTIFICATION_ID,
    NotificationTable.INDEX_CREATE_GROUP_ID,
    NotificationTable.INDEX_CREATE_COLLAPSE_ID,
    NotificationTable.INDEX_CREATE_CREATED_TIME
};

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_ENTRIES);
    db.execSQL(SQL_INDEX_ENTRIES);
    for (String ind : SQL_INDEX_ENTRIES) {
        db.execSQL(ind);
    }
}
Thanks for the catch! For reference this the multiple statements per `execSQL` is noted below.

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

Thanks!

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