Subject: PostgreSql does not handle BOOL correctly
Posted by omari on Sun, 11 Jun 2023 15:12:50 GMT
View Forum Message <> Reply to Message

Test case:

with this schema:

```
TABLE_(MY_TABLE)
  INT_   (ID) PRIMARY_KEY
  BOOL_  (IS_OK)
END_TABLE
```

this code:

```
SqlStatement s = Select(ID).From(MY_TABLE).WHERE(IS_OK == true);
LOG(s.Get(PGSQL));
```

print :

```
select ID from MY_TABLE where IS_OK = 1
```

PostgreSql does not accept this sql statement.

the correct one is:

```
select ID from MY_TABLE where IS_OK = '1'
```

File Attachments

1) pgsql_bool.PNG, downloaded 867 times

The following table shows the valid literal values for `TRUE` and `FALSE` in PostgreSQL.

True	False
true	false
't'	'f '
'true'	'false'
'y'	'n'
'yes'	'no'
'1'	'0'

Note that the leading or trailing whitespace does not matter and all the constant values except for `true` and `false` must be enclosed in single quotes.

Subject: Re: PostgreSql does not handle BOOL correctly
Posted by omari on Thu, 27 Jul 2023 14:09:55 GMT
View Forum Message <> Reply to Message

In order to provide a test case, i have modified PostgreSql referance example by adding a bool field:

```
TABLE_(TESTPARTNER)
  SERIAL_  (TESTPARTNER_ID) PRIMARY_KEY
  STRING_  (TESTPARTNER_NAME, 200) INDEX
  STRING_  (TESTPARTNER_ADDRESS, 200)
  BOOL_    (IS_OK)                <---------- field added
END_TABLE
```

and i have added a Where condition to the Query:

```
 m_array.Query(IS_OK == true);
```

the result is: at runtime, it show this ERROR messages :

The operation has failed

SQL error:
ERREUR: l'opérateur n'existe pas : character = integer
LINE 1: ...RTNER_ADDRESS, IS_OK from TESTPARTNER where IS_OK = 1 order ...
                                                             ^
HINT: Aucun opérateur ne correspond au nom donné et aux types d'arguments.
Vous devez ajouter des conversions explicites de type.

Error statement:
select TESTPARTNER_ID, TESTPARTNER_NAME, TESTPARTNER_ADDRESS, IS_OK from TESTPARTNER where IS_OK = 1 order by TESTPARTNER_NAME

File Attachments

1) SQL_PostgreSql.7z, downloaded 253 times

Subject: Re: PostgreSql does not handle BOOL correctly
Posted by omari on Thu, 27 Jul 2023 14:15:41 GMT
View Forum Message <> Reply to Message

Running this statement :

select TESTPARTNER_ID, TESTPARTNER_NAME, TESTPARTNER_ADDRESS, IS_OK from TESTPARTNER where IS_OK = 1 order by TESTPARTNER_NAME

in postgresql Sql console, print the same error message.

the correct statement is:

select TESTPARTNER_ID, TESTPARTNER_NAME, TESTPARTNER_ADDRESS, IS_OK from TESTPARTNER where IS_OK = '1' order by TESTPARTNER_NAME

where IS_OK = '1'
instead of
where IS_OK = 1

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by omari on Wed, 04 Oct 2023 16:50:57 GMT
View Forum Message <> Reply to Message

I managed to resolve this bug,
here is my solution:

Sql/SqlExp.h:
line 67:

```
enum {
 SQLC_IF = 1,
 SQLC_ELSEIF = 2,
 SQLC_ELSE = 3,
 SQLC_ENDIF = 4,
 SQLC_DATE = 5,
 SQLC_TIME = 6,
 SQLC_STRING = 7,
 SQLC_BINARY = 8,
 SQLC_ID = 9, // '\t'
 SQLC_OF = 10,
 SQLC_AS = 11,
 SQLC_COMMA = 12,
 SQLC_BOOL = 13,        +++
};
```

Line 99:

```
String SqlFormat(int x);
String SqlFormat(bool x);         +++
String SqlFormat(double x);
String SqlFormat(int64 x);
```

Line 212:

```
class SqlVal : public SqlS, Moveable<SqlVal> {
public:
 ...

 SqlVal(int x);
 SqlVal(bool x);       +++
 SqlVal(int64 x);
 SqlVal(double x);
 ...
};
```

Sql/SqlVal.cpp:
line 128: ADD

```
SqlVal::SqlVal(bool x) {
 if(UPP::IsNull(x))
  SetNull();
 else
  SetHigh(SqlFormat(x));
}
```

Sql\SqlCode.cpp
Line 146: ADD

```
  case SQLC_BOOL: {
   LTIMING("SqlCompile BOOL");
   bool x;
   ReadSqlValue(x, s);
   if(!r) break;
   if(IsNull(x)) {
    *r << "NULL";
    break;
   }
   switch(dialect) {
   case PGSQL:
    *r << ( x ? "'1'" : "'0'");
    break;
   default:
    *r << ( x ? "1" : "0");
   }
   break;
  }
  case SQLC_DATE: {
```

Line 429: ADD

```
String SqlFormat(bool x)
{
 return MakeSqlValue(SQLC_BOOL, x);
}
```

and change the next function:

```
String SqlFormat(const Value& x)
{
 if(x.IsNull()) return "NULL";
 switch(x.GetType()) {
 case BOOL_V:
  return SqlFormat((bool) x);     +++
 case INT_V:
  return SqlFormat((int) x);
 case INT64_V:
  return SqlFormat((int64) x);
 case DOUBLE_V:
```

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by mirek on Fri, 20 Oct 2023 07:41:45 GMT
View Forum Message <> Reply to Message

I think this is a bit of misunderstanding. Schema BOOL is defined as char(1), for better or worse, and thus follows text operations rules.

The reason for this is that we try to be "universal" and not all RDBMS support bool columns, so char(1) is the safe bet.

Maybe we should introduce another column type, defined only when possible, which would only work with some databases.

But then again, as long as you know that BOOL is actually char(1), it is trivial to adjust your SQL coding with the benefit of begin more RDBMS agnostic (arguably small, but hey - I really did have a well paid job converting some of my apps to use MSSQL instead of Oracle, thanks to this, it was quite easy).

(BTW, you made me check whether the documentation mentions it; it does: https://www.ultimatepp.org/srcSqlsch_en-us.html)

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by mirek on Fri, 20 Oct 2023 07:52:35 GMT
View Forum Message <> Reply to Message

After rereading, I think you actually suggest something a little bit different (interpreting C++ bool as Sql text).

Well, I feel bad about that. bool is too close to integers, I do not see it safe to test Value on bool inside and returning something fundamentally different (text literal instead of number literal).

As I said, BOOL is char(1), adjust your SQL. It works this way since 2003, so it is not a bug, feature request at best

---

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by omari on Fri, 20 Oct 2023 22:33:17 GMT
View Forum Message <> Reply to Message

Hi Mirek,

the actual solution traite BOOL field like INT, with value 0 for false and 1 for true.
this work well for all dialect, except PGSQL.(I will provide a test case in the following post)

my patch, create a new type that is like INT (0: false, 1: true) for all dialect, but like char(1) for PGSQL ('0': false, '1': true)

Quote:
```
  switch(dialect) {
  case PGSQL:
   *r << ( x ? "'1'" : "'0'");
   break;
  default:
   *r << ( x ? "1" : "0");
  }
```

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by omari on Fri, 20 Oct 2023 23:14:00 GMT
View Forum Message <> Reply to Message

This test case (attached) FAIL without the patch, and PASS with the patch.

```
 if(!OpenDB()) return;

 Sql sql(session);
 sql * Insert(TEST1) (ID, 1) (B, true);
 sql * Insert(TEST1) (ID, 2) (B, false);

 sql*Select(SqlAll()).From(TEST1).Where(B == true);
 LOG(sql.ToString());
 if (sql.Fetch()) {
  ASSERT(sql[ID] == 1);
 }
 else {
```

```
 ASSERT(false); // Failed : 'Select' should return one row
}

sql*Select(SqlAll()).From(TEST1).Where(B == false);
LOG(sql.ToString());
if (sql.Fetch()) {
 ASSERT(sql[ID] == 2);
}
else {
 ASSERT(false); // Failed : 'Select' should return one row
}
```

the schema file:

```
TABLE_(TEST1)
 INT_   (ID) PRIMARY_KEY
 BOOL_  (B)
END_TABLE
```

File Attachments

1) SQL_BOOL_TEST.7z, downloaded 250 times

Subject: Re: PostgreSql does not handle BOOL correctly [BUG+PATCH]
Posted by mirek on Sat, 21 Oct 2023 15:22:51 GMT
View Forum Message <> Reply to Message

omari wrote on Sat, 21 October 2023 01:14This test case (attached) FAIL without the patch, and PASS with the patch.

```
if(!OpenDB()) return;

Sql sql(session);
sql * Insert(TEST1) (ID, 1) (B, true);
sql * Insert(TEST1) (ID, 2) (B, false);

sql*Select(SqlAll()).From(TEST1).Where(B == true);
LOG(sql.ToString());
if (sql.Fetch()) {
 ASSERT(sql[ID] == 1);
}
else {
 ASSERT(false); // Failed : 'Select' should return one row
```

```
}

sql*Select(SqlAll()).From(TEST1).Where(B == false);
LOG(sql.ToString());
if (sql.Fetch()) {
 ASSERT(sql[ID] == 2);
}
else {
 ASSERT(false); // Failed : 'Select' should return one row
}
```

the schema file:

```
TABLE_(TEST1)
 INT_   (ID) PRIMARY_KEY
 BOOL_  (B)
END_TABLE
```

That is all good and fine, just not the way it was intended to work. (And I do not claim that the intended way is the best but it worked fine for 20+ years.)

sql*Select(SqlAll()).From(TEST1).Where(B == '1')

is not all that harder to do and IMO it is more "honest" (we do not pretend that B is not char).

Frankly, the only differense between BOOL and STRING(1) apart from documentation purposes is that gets converted to bool in S_ structures. If you do not like that, do not use BOOL.

Mirek
