Subject: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sat, 14 Apr 2007 22:28:28 GMT
View Forum Message <> Reply to Message

I have created experimental PostgreSQL classes and a dumb example.

The problems of it:

 it is incomplete yet, you can see the empty method bodies and comments
 blobs and reference constraints are untested.
 tested on windows only
 Auto increment field handling needs some unification in UPP. mysql has some API function to query the last generated value, just like sqlite. But Pg has a different architecture, as it uses per table sequences with names generated from table and field name.
 I created SERIAL and BIGSERIAL types in schema

Upp needs a modification to use this:
In Sql/SqlSchema.cpp in SqlSchema::FlushColumn():

```
   else if (dialect == SQLITE3)
    Upgrade() << Expand("alter table @t add ") << cd << ";\n";
   else if (dialect == POSTGRESS)
    Upgrade() << Expand("alter table @t add \n") << cd << "\n;\n\n";
   else
    Upgrade() << Expand("alter table @t add (\n") << cd << "\n);\n\n";
```

comments, patches are welcome.

File Attachments
1) PostgreSQLAndTest-01.zip, downloaded 517 times

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Sun, 15 Apr 2007 07:11:55 GMT
View Forum Message <> Reply to Message

Wonderful news! I was about to create this myself. I will test it soon!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by fallingdutch on Sun, 15 Apr 2007 07:19:32 GMT
View Forum Message <> Reply to Message

Thanks for that news!

Bas

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Sun, 15 Apr 2007 14:02:04 GMT
View Forum Message <> Reply to Message

unodgs wrote on Sun, 15 April 2007 03:11Wonderful news! I was about to create this myself. I will test it soon!

Please do and merge it into the uppsrc ASAP. (Of course, if we have Zsolt's persmission?)

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sun, 15 Apr 2007 15:44:28 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 15 April 2007 16:02unodgs wrote on Sun, 15 April 2007 03:11Wonderful news! I was about to create this myself. I will test it soon!

Please do and merge it into the uppsrc ASAP. (Of course, if we have Zsolt's persmission?)

Mirek
Of course you have

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Sun, 15 Apr 2007 18:25:04 GMT
View Forum Message <> Reply to Message

I tried the plugin. It works great but I had to modify the PostrgreSQLTest databse schema file. In this file there is:

```
TABLE_(TESTPARTNER)
  SERIAL_  (TESTPARTNER_ID) PRIMARY_KEY
  STRING_  (TESTPARTNER_NAME, 200) INDEX
  STRING_  (TESTPARTNER_ADDRESS, 200)
END_TABLE

TABLE_(TESTPRODUCT)
  SERIAL_  (TESTPRODUCT_ID) PRIMARY_KEY
  STRING_  (TESTPRODUCT_NAME, 200) INDEX
  STRING_  (TESTPRODUCT_UNIT, 20)
END_TABLE
```

In postrgre schema macros SERIAL is defined as
#define SERIAL(x) COLUMN("serial primary key", int64, x, 0, 0)
so in final sql file there is double primary key attribute:

```
create table TESTPARTNER (
  TESTPARTNER_ID       serial primary key primary key,
  TESTPARTNER_NAME     varchar(200),
  TESTPARTNER_ADDRESS  varchar(200)
);

create table TESTPRODUCT (
  TESTPRODUCT_ID       serial primary key primary key,
  TESTPRODUCT_NAME     varchar(200),
  TESTPRODUCT_UNIT     varchar(20)
);
```

I think we should remove primary key from SERIAL and BIGSERIAL macros and let the user explicitly define it in schema file.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Sun, 15 Apr 2007 18:49:58 GMT
View Forum Message <> Reply to Message

Ok. I removed the primary key from macros (if that is mistake we will restore it). I've added it to uvs (example and "driver")

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sun, 15 Apr 2007 18:59:15 GMT
View Forum Message <> Reply to Message

sorry, I didn't update test app the content of .sch file should be

```
TABLE_(TESTPARTNER)
   SERIAL_  (TESTPARTNER_ID)
   STRING_  (TESTPARTNER_NAME, 200) INDEX
   STRING_  (TESTPARTNER_ADDRESS, 200)
END_TABLE

TABLE_(TESTPRODUCT)
   SERIAL_  (TESTPRODUCT_ID)
   STRING_  (TESTPRODUCT_NAME, 200) INDEX
   STRING_  (TESTPRODUCT_UNIT, 20)
END_TABLE
```

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sun, 15 Apr 2007 19:04:42 GMT
View Forum Message <> Reply to Message

My original idea was to have a database independent app.
So I created this SERIAL field to generate a serial id.

I put SERIAL, BIGSERIAL and CLOB into my sqlite schema also:
#define SERIAL(x)   COLUMN("integer primary key autoincrement", int64, x, 0, 0) //int is not enough, as it is unsigned
#define SERIAL_ARRAY(x, items)   COLUMN_ARRAY("integer primary key autoincrement", int64, x, 0, 0, items)
#define SERIAL_(x)   COLUMN_("integer primary key autoincrement", int64, x, 0, 0)
#define SERIAL_ARRAY_(x, items)   COLUMN_ARRAY_("integer primary key autoincrement", int64, x, 0, 0, items)

#define BIGSERIAL(x)   COLUMN("integer primary key autoincrement", int64, x, 0, 0)
#define BIGSERIAL_ARRAY(x, items) COLUMN_ARRAY("integer primary key autoincrement", int64, x, 0, 0, items)
#define BIGSERIAL_(x)   COLUMN_("integer primary key autoincrement", int64, x, 0, 0)
#define BIGSERIAL_ARRAY_(x, items) COLUMN_ARRAY_("integer primary key autoincrement", int64, x, 0, 0, items)

#define CLOB(x)   COLUMN("text", String, x, 0, 0)
#define CLOB_(x)   COLUMN_("text", String, x, 0, 0)

I don't know if this is correct or not, but we have to find some common schema requirement for all kinds of databases to achieve portability.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sun, 15 Apr 2007 19:18:54 GMT
View Forum Message <> Reply to Message

So more clearly, what I would like to see in UPP:

Common typename for automatically incremented id fields, used as primary key (SERIAL, AUTOINC or something else)
Common typename for text fields with size limit (CLOB, TEXT or something else)
A common infrastructure to handle SERIAL like fields in SqlArray like cases

Subject: Re: PostgreSQL Support Classes [Experimental]

Posted by mirek on Mon, 16 Apr 2007 07:20:53 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 15 April 2007 15:18So more clearly, what I would like to see in UPP:

Common typename for automatically incremented id fields, used as primary key (SERIAL, AUTOINC or something else)
Common typename for text fields with size limit (CLOB, TEXT or something else)
A common infrastructure to handle SERIAL like fields in SqlArray like cases

What is wrong with GetSerialId?

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Mon, 16 Apr 2007 08:22:51 GMT
View Forum Message <> Reply to Message

I think one should manualy write in schema file that a field is primary key, even if serial generates unique values. That's better IMO. Someone else without deep knowledge about given database know immediately what is primary key. And we should remember that we provide a COMMON way to define databse structure. In others databases serial not awlays means primary key (rather auto increment)

I think we should solve the problem of multi-field primary key.
Does it work in upp?:

```
TABLE(Test)
  INT ID PRIMARY_KEY
  INT VERSION PRIMARY_KEY
  ...
END_TABLE
```

Antoher thing are database scripts. There should be ONE funtcion to create/update database schema like:

```
OleDBSession db;
db.Prepare() - creates db or update it
```

and some additional funtions

```
db.Create() - only creating db
db.Update() - only updating db
db.Drop() - removing all tables
```

rather than

```
SqlSchema sch(SQLITE3);
StdStatementExecutor se(sqlite3);
All_Tables(sch);
if(sch.ScriptChanged(SqlSchema::UPGRADE))
 Sqlite3PerformScript(sch.Upgrade(),se);
if(sch.ScriptChanged(SqlSchema::ATTRIBUTES)) {
 Sqlite3PerformScript(sch.Attributes(),se);
}
if(sch.ScriptChanged(SqlSchema::CONFIG)) {
 Sqlite3PerformScript(sch.ConfigDrop(),se);
 Sqlite3PerformScript(sch.Config(),se);
}
sch.SaveNormal();
```

It is too complicated.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Mon, 16 Apr 2007 08:37:35 GMT

View Forum Message <> Reply to Message

unodgs wrote on Mon, 16 April 2007 04:22
And we should remember that we provide a COMMON way to define databse structure. In others databases serial not awlays means primary key (rather auto increment)

IMO, RDBMS are too different to provide single common .sch "syntax&semantics" for all of them.

Quote:
I think we should solve the problem of multi-field primary key.
Does it work in upp?:

In Oracle, yes:

Quote:
```
TABLE_(O_POLOZKA)
 INT_    (O_POLOZKA_SEQ)     INDEX
 INT     (O_DOKLAD_SEQ)      REFERENCES(O_DOKLAD) INDEX
 STRING  (POPIS, 2000)
 DOUBLE_ (CENAJ)
 INT_    (MNC)
 INT_    (MNJ)
 DUAL_PRIMARY_KEY(O_POLOZKA_SEQ, O_DOKLAD_SEQ)
```

END_TABLE

Obviously, not as sexy as simply puttin more "PRIMARY_KEY" columns, OTOH at least Oracle SQL syntax is similar - you have to add it in single contraint, not as several "PRIMARY_KEY" comlums.

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Mon, 16 Apr 2007 10:13:35 GMT
View Forum Message <> Reply to Message

Quote:Obviously, not as sexy as simply puttin more "PRIMARY_KEY" columns, OTOH at least Oracle SQL syntax is similar - you have to add it in single contraint, not as several "PRIMARY_KEY" comlums.
If databse is SQL 92 compatible there is posibility to define primary key in single line. It seems that most of db engines support it.

```
create table t
(
  id integer,
  ver integer,
  type integer,
  ...,
  primary key(id, ver, type)
)
```

I proposed putting PRIMARY_KEY attribute next to every column to avoid several macros like:
SINGLE_PRIMARY_KEY
DUAL_PRIMARY_KEY
TRIO_PRIMARY_KEY etc..

Of course I don't know if it is possible to implement it using c preprocessor. If not - I suggest to get rid of primary_key attribute as well as DUAL_PRIMARY_KEY and define new macro PRIMARY_KEY(...) which allow you to define primary key for maximum 5 columns.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by jibe on Mon, 16 Apr 2007 16:46:51 GMT
View Forum Message <> Reply to Message

Hi zsolt,

Sorry to be a little out of topic... I'm studying if I could develop an interface for Firebird, and experience some difficulties to get started. I'm downloading your work, maybe it will help. But I worry if you have some docs or notes that could help me ?

Thanks.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Mon, 16 Apr 2007 17:43:52 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 16 April 2007 10:22I think one should manualy write in schema file that a field is primary key, even if serial generates unique values. That's better IMO. Someone else without deep knowledge about given database know immediately what is primary key. And we should remember that we provide a COMMON way to define databse structure. In others databases serial not awlays means primary key (rather auto increment)

I agree.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Mon, 16 Apr 2007 17:44:42 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 16 April 2007 06:13Quote:Obviously, not as sexy as simply puttin more "PRIMARY_KEY" columns, OTOH at least Oracle SQL syntax is similar - you have to add it in single contraint, not as several "PRIMARY_KEY" comlums.
If databse is SQL 92 compatible there is posibility to define primary key in single line. It seems that most of db engines support it.

```
create table t
(
  id integer,
  ver integer,
  type integer,
  ...,
  primary key(id, ver, type)
)
```

Sure, above is a direct equivalent of what have now. SQL allows to define single-column keys in the column and multi-column keys as additional contraint...

Quote:
I proposed putting PRIMARY_KEY attribute next to every column to avoid several macros.
Of course I don't know if it is possible to implement it using c preprocessor.

You would have radically change schema code, but I guess it is possible somehow.

Quote:
 If not - I suggest to get rid of primary_key attribute as well as DUAL_PRIMARY_KEY and define new macro
PRIMARY_KEY(...) which allow you to define primary key for maximum 5 columns.

Well, but C++ does not have variadic macros (yet?).

BTW, I have just checked and I have used DUAL_PRIMARY_KEY once in my "grand IDIS DB schema" of 500 tables... (OTOH, it is also true that I perhaps deliberately avoid it).

My suggestion is to use current way for now; maybe we can do better when we have all RDBMs implemented and have some experience with it; maybe we will then develop some "common .sch language" to have single .sch for all of them.

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Mon, 16 Apr 2007 17:57:21 GMT

View Forum Message <> Reply to Message

Hi jibe,
it is not very complicated. I used SQLite classes as a starting point.
I opened sqlite and postgresql library docs and started implementation.
For checking the created things I made a dumb app.

You can start, based on my classes as well, but PostgreSQL datatype handling is a little bit complicated, so you don't have to use that part, I think.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Mon, 16 Apr 2007 18:11:02 GMT

View Forum Message <> Reply to Message

luzr wrote on Mon, 16 April 2007 13:44
BTW, I have just checked and I have used DUAL_PRIMARY_KEY once in my "grand IDIS DB schema" of 500 tables... (OTOH, it is also true that I perhaps deliberately avoid it).

Sure, in my project the biggest primary key consist of 3 columns. But there is no macro to define it and probably there is someone out there who needs more

---

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Mon, 16 Apr 2007 18:37:25 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 16 April 2007 14:11
Sure, in my project the biggest primary key consist of 3 columns. But there is no macro to define it

INLINE_ATTRIBUTE(", primary key (KEY1, KEY2, KEY3, KEY4)")

OTOH, perhaps not a bad idea is this:

#define PRIMARY_KEYS(keys)   INLINE_ATTRIBUTE(", primary key (" keys ")")

PRIMARY_KEYS("KEY1, KEY2, KEY3")

does not sound that bad...

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by unodgs on Mon, 16 Apr 2007 20:42:28 GMT
View Forum Message <> Reply to Message

It looks fine to me! Maybe except S letter at the end sugessting multiple primary keys. But if you could add it to sql schema do it please!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Mon, 16 Apr 2007 22:01:13 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 16 April 2007 16:42It looks fine to me! Maybe except S letter at the end sugessting multiple primary keys. But if you could add it to sql schema do it please!

'S' to avoid ambiguity with PRIMARY_KEY.

Also, this is RDBMS specific (so it would have to be added for each RDBMS).

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by jibe on Tue, 17 Apr 2007 07:08:30 GMT
View Forum Message <> Reply to Message

Hi zsolt,

Thanks for your reply ! Maybe I worry only because I'm just discovering UPP, and have no experience with it !

To go back to the topic :

luzr wrote on Mon, 16 April 2007 20:37PRIMARY_KEYS("KEY1, KEY2, KEY3")

does not sound that bad...

Yes, it looks like very much what I use with Firebird, and is also common with some other databases...

unodgs wrote on Mon, 16 April 2007 22:42It looks fine to me! Maybe except S letter at the end sugessting multiple primary keys. But if you could add it to sql schema do it please!
Yes, I think also that this S is not so usefull... Why not

PRIMARY KEY ("UNIQUE_KEY")
or
PRIMARY KEY ("KEY1, KEY2, ...")
It seems logical, and I don't see any ambiguity ? But maybe I don't know enough yet about all consequences in upp...

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Tue, 17 Apr 2007 09:53:17 GMT
View Forum Message <> Reply to Message

---

jibe wrote on Tue, 17 April 2007 03:08
PRIMARY KEY ("KEY1, KEY2, ...")
It seems logical, and I don't see any ambiguity ? But maybe I don't know enough yet about all consequences in upp...

There is already macro PRIMARY_KEY for single column primary keys. Well, in theory, it could be replaced by this new macro, but

a) as single-column primary keys are so much widely use, having it as special case is a good idea

b) would break all of my commercial stuff...

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by jibe on Wed, 18 Apr 2007 06:57:14 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 17 April 2007 11:53
a) as single-column primary keys are so much widely use, having it as special case is a good idea
Yes, maybe... But I don't see the problem if the same macro is used the same way for one or several colomn keys...
Seems a lot easier if there is no difference ?

luzr wrote on Tue, 17 April 2007 11:53
b) would break all of my commercial stuff...

Yes, this is a good reason !

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by tojocky on Thu, 01 May 2008 14:53:50 GMT
View Forum Message <> Reply to Message

I have an error! Sorry for my incompetence but I need help!

c:\myapps\postgresql\PostgreSQL.h(7) : fatal error C1083: Cannot open include file: 'libpq-fe.h': No such file or directory

Thanks a lot!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Thu, 01 May 2008 20:55:57 GMT
View Forum Message <> Reply to Message

tojocky wrote on Thu, 01 May 2008 10:53I have an error! Sorry for my incompetence but I need help!

c:\myapps\postgresql\PostgreSQL.h(7) : fatal error C1083: Cannot open include file: 'libpq-fe.h': No such file or directory

Thanks a lot!

You are missing the correct include path (and quite likely library path, that would hit later...) in build method.

You need to have PostgreSQL installed and you need to put proper paths into the build method (in Setup menu) you are using.

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by tojocky on Sun, 04 May 2008 08:15:33 GMT
View Forum Message <> Reply to Message

Thank you! its worked! I will testing in continuation!
I have a situation: I need to make a server and client application. that in code i need decide where it will execute (server or client). Only application server can communicate with DB server, but client is connected with server. Server and client appliction need becouse:
1. The client station can/may be not so performace as server station.
2. Connection from DB server and application server can be more speedily.
If I wrong tell me please how is correct! How I can create this?

Thank you!
John!

P.S. The ultimate ++ is the most impression me!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by zsolt on Sun, 04 May 2008 09:24:44 GMT
View Forum Message <> Reply to Message

I think, if you do not have more than 2000 clients, use simple client-server architecture:
- SQL database on server
- SQL clients on clients
It is not a good idea to create an over-complicated system without any reason.
Additionally, U++ doesn't implement muli-tier techhologies currently (AFAIK).

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by tojocky on Mon, 05 May 2008 15:12:26 GMT
View Forum Message <> Reply to Message

On Application server I will execute massive operations with data! and client station can be not so performance. About count of clients will be more 100 (not 2000)!
Thank you for advice, Zsolt!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mr_ped on Mon, 05 May 2008 18:59:09 GMT
View Forum Message <> Reply to Message

Massive operations with data = simulation of A-bomb or 3D game or HD video encoding or what?

Unless it is top secret, I would be interested what is so "massive" that ordinary 1GHz CPU is not enough to solve it within 0.1-0.5s.

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Mon, 05 May 2008 20:14:02 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Mon, 05 May 2008 14:59Massive operations with data = simulation of A-bomb or 3D game or HD video encoding or what?

Unless it is top secret, I would be interested what is so "massive" that ordinary 1GHz CPU is not enough to solve it within 0.1-0.5s.

Uhmps, while I think where you are heading, I have a lot of selects in my apps that cannot be optimized any further way and take more than 10s on 2Ghz machine

Anyway, this of course is irrelevant here.

Mirek

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by tojocky on Tue, 06 May 2008 15:24:35 GMT
View Forum Message <> Reply to Message

Massive operations it will be with data from database! The application consists in automation evidence of firm (salary, accounting, production)!

Subject: Re: PostgreSQL Support Classes [Experimental]
Posted by mirek on Tue, 06 May 2008 17:45:55 GMT
View Forum Message <> Reply to Message

Yep, that is what SQL database are created for....

Mirek