Subject: Fields
Posted by unodgs on Wed, 26 Jul 2006 07:41:32 GMT
View Forum Message <> Reply to Message

What I miss in current implementation of sql interface is some kind of Fields class. Let me explain what I mean:

Let's imagine that if an insert operation fails we want to do an update. At present I have to write:

```
sql * Insert(TABLE)(C0, "a")(C1, 1)(C3, 2)
if(duplicate())
  sql * Update(TABLE)(C0, "a")(C1, 1)(C3, 2)
```

The thing is I have to write fields names and its values twice. I would like to be able to write it like this:

```
Fields flds;
flds(C0, "a")(C1, 1)(C3, 2);

sql * Insert(TABLE).Fields(flds)
if(duplicate())
  sql * Update(TABLE).Fields(flds)
```

If that was possible it would be easier to build dynamic queries with sqlexp. Second case (similar to one in my app):

```
void InsertPerson(Sql &sql, bool flag)
{
   SqlInsert q = Insert(TABLE)
   if(flag)
   {
      q(ID, 1)
      q(STATUS, "A")
   }
   q(NAME, "Daniel")
   q(AGE, 27)
   ..... << more fields

   sql * q;
}

void UpdatePerson(Sql &sql, ...)
{
   SqlUpdate q = Update(TABLE)
```

```
   q(NAME, "Daniel")
   q(AGE, 27)
   .....

   sql * q.Where(ID == id);
}
```

With Fields class I could separate common fields and write:

```
void InsertPerson(Sql &sql, Fields &flds, bool flag)
{
   SqlInsert q = Insert(TABLE)
   if(flag)
   {
      q(ID, 1)
      q(STATUS, "A")
   }
   q.Fields(flds) //or in short form - q(flds)

   sql * q;
}

void UpdatePerson(Sql &sql, Fields &flds...)
{
   SqlUpdate q = Update(TABLE)
   q.Fields(flds)

   sql * q.Where(ID == id);
}
```

Now it is impossible and I have to do something like this:

```
template<typename T>
void Fields(T &q, DataCommon &dc)
{
    q(NAME, dc.name);
    q(AGE, dc.age);
}

int InsertPerson(Sql &sql, DataCommon &dc, void (*AddFields)(SqlInsert &, DataCommon &), ...)
{
    SqlInsert q = Update(TABLE)
    .....
    AddFields(q, dc);
    sql * q;
```

```
}

int UpdatePerson(Sql &sql, DataCommon &dc, void (*AddFields)(SqlUpdate &, DataCommon &),
...)
{
   SqlUpdate q = Update(TABLE)
   ....
   AddFields(q, dc);
   sql * q.Where(ID == id);
}
```

and finally...

```
InsertPerson(sql, dc, Fields<SqlInsert>, ...)
UpdatePerson(sql, dc, Fields<SqlUpdate>, ...)
```

and it all because both SqlInsert and SqlUpdate class have separate interfaces for adding fields to insert/update.

Can be in future similar functionality added?
