

---

Subject: Re: PROPOSAL: Access to S\_\* Structure of TABLE crash Application.  
Posted by [Sender Ghost](#) on Sun, 20 May 2012 02:09:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Based on experience, when returned result set contains column names in the same order as constructed, I created following helper function and class:

Toggle Spoiler

NAMESPACE\_UPP

```
template<> String AsString(const SqlSet& set) {
    StringBuffer text;
    const char *s = ~set;

    for(;;) {
        const char *b = s;

        while((byte)*s >= 32)
            s++;

        text.Cat(b, s);
        int c = *s;

        if (c == SQLC_AS) {
            text << ' ';
        }
        else
        if (c == SQLC_OF)
            text << '.';
        else
        if (c == SQLC_COMMA)
            text << ", ";

        s++;

        if (c == '\0' || c == SQLC_ID)
            break;
    }

    return text;
}

class SqlSetIndex {
private:
    VectorMap<String, int> list;
    inline void Parse(const SqlSet& set);
public:
    SqlSetIndex(const SqlSet& set)      { Parse(set); }
```

```

void Clear()                { list.Clear(); }
int operator[](const SqlId& id) const { return list.Get(~id); }
int operator[](const String& id) const { return list.Get(id); }
SqlSetIndex& operator=(const SqlSet& set) { list.Clear(); Parse(set); return *this; }
const VectorMap<String, int>& operator~() { return list; }
};

```

```

void SqlSetIndex::Parse(const SqlSet& set)

```

```

{
    int index = 0;
    StringBuffer text;
    bool isNext = true, isAs = false;
    const char *s = ~set;

    for(;;) {
        const char *b = s;

        while((byte)*s > 32 && (byte)*s != ',')
            s++;

        text.Cat(b, s);
        int c = *s;

        if (c == SQLC_OF) { // ','
            text << *s;
            isNext = false;
        }
        else
            if (c == ' ')
                isNext = false;
            else
                if (c == SQLC_AS)
                    isAs = true;

        if (isNext) {
            if (!isAs)
                list.GetAdd(text, index++);
            else {
                list.GetAdd(text, index);
                isAs = false;
            }
        }

        text.Clear();
    }
    else
        isNext = true;

    s++;
}

```

```

if (c == '\0' || c == SQLC_ID)
    break;
}
}

```

END\_UPP\_NAMESPACE

- AsString function for SqlSet, which replaces special SQLC\_\* characters, almost the same as SqlCompile function does for full SQL statement.
- SqlSetIndex class to parse SqlSet to VectorMap<String, int> container with column names and their indexes.

Now, the part of example will look like follows:

Toggle Spoiler

```

LOG("Inserting values:");
SQL * Insert(WORKER)(ID, 0)(NAME, "Joe")(LASTNAME, "Smith")(PLANT_ID, 0);
LOG(SQL.ToString());
SQL * Insert(WORKER)(ID, 1)(NAME, "Mike")(LASTNAME, "Smith")(PLANT_ID, 0);
LOG(SQL.ToString());
SQL * Insert(WORKER)(ID, 2)(NAME, "Jon")(LASTNAME, "Goober")(PLANT_ID, 1);
LOG(SQL.ToString());

```

```

SQL * Insert(PLANT)(ID, 0)(NAME, "First Plant")(ADDRESS, "First st.");
LOG(SQL.ToString());
SQL * Insert(PLANT)(ID, 1)(NAME, "Second Plant")(ADDRESS, "Second st.");
LOG(SQL.ToString());

```

```

SQLID(PLANT_NAME); // Just to show parsing of "AS" clause here.
LOG("\n' << "Testing SqlSetIndex:");
SqlSet testing_set(WORKER(ID, NAME), ID.Of(WORKER), PLANT(NAME).As(PLANT_NAME),
PLANT(ID, NAME, ADDRESS));
DUMP(testing_set);
SqlSetIndex testing_list(testing_set);
DUMPM(~testing_list);

```

```

LOG("\n' << "Selecting values:");
SqlSet set(WORKER(NAME, LASTNAME), PLANT(NAME).As(PLANT_NAME),
PLANT(ADDRESS));
SqlSetIndex list(set);

```

```

SQL * Select(set).From(WORKER).LeftJoin(PLANT).On(WORKER(PLANT_ID) == PLANT(ID));
LOG(SQL.ToString());
while (SQL.Fetch()) {
    LOG("-----");
    DUMP(SQL[list[WORKER(NAME)]];

```

```

DUMP(SQL[list[WORKER(LASTNAME)]]);
DUMP(SQL[list[PLANT(NAME)]]);
//DUMP(SQL[list[PLANT_NAME]]); // The same as previous value
DUMP(SQL[list[PLANT(ADDRESS)]]);
}

```

With following output:

Toggle Spoiler

Inserting values:

```

insert into WORKER(ID, NAME, LASTNAME, PLANT_ID) values (0, 'Joe', 'Smith', 0)
insert into WORKER(ID, NAME, LASTNAME, PLANT_ID) values (1, 'Mike', 'Smith', 0)
insert into WORKER(ID, NAME, LASTNAME, PLANT_ID) values (2, 'Jon', 'Goober', 1)
insert into PLANT(ID, NAME, ADDRESS) values (0, 'First Plant', 'First st.')
insert into PLANT(ID, NAME, ADDRESS) values (1, 'Second Plant', 'Second st.')

```

Testing SqlSetIndex:

```

testing_set = WORKER.ID, WORKER.NAME, WORKER.ID, PLANT.NAME PLANT_NAME,
PLANT.ID, PLANT.NAME, PLANT.ADDRESS

```

~testing\_list:

```

[0] = (WORKER
ID) 0
[1] = (WORKER
NAME) 1
[2] = (PLANT
NAME) 3
[3] = (PLANT_NAME) 3
[4] = (PLANT
ID) 4
[5] = (PLANT
ADDRESS) 6

```

Selecting values:

```

select WORKER.NAME, WORKER.LASTNAME, PLANT.NAME PLANT_NAME,
PLANT.ADDRESS from WORKER left outer join PLANT on WORKER.PLANT_ID = PLANT.ID

```

-----

```

SQL[list[WORKER(NAME)]] = Joe
SQL[list[WORKER(LASTNAME)]] = Smith
SQL[list[PLANT(NAME)]] = First Plant
SQL[list[PLANT(ADDRESS)]] = First st.

```

-----

```

SQL[list[WORKER(NAME)]] = Mike
SQL[list[WORKER(LASTNAME)]] = Smith
SQL[list[PLANT(NAME)]] = First Plant
SQL[list[PLANT(ADDRESS)]] = First st.

```

-----

```

SQL[list[WORKER(NAME)]] = Jon

```

SQL[list[WORKER(LASTNAME)]] = Goober  
SQL[list[PLANT(NAME)]] = Second Plant  
SQL[list[PLANT(ADDRESS)]] = Second st.

The full example with SqlSetIndex class you could find in the attachment.

Edit:  
Fixed some typos in the text.

### File Attachments

---

1) [SqlSetIndex.zip](#), downloaded 462 times

---