Subject: PATCH/BUGFIX Oracle global temporary table
Posted by wqcmaster on Wed, 14 Oct 2015 08:15:31 GMT
View Forum Message <> Reply to Message

Hi,

Oracle Definition for global temporary table: create global temporary table <tablename> on commit <...> as (<select_cmd>);
PostgreSQL Definition for global temporary table: create [global] temporary table <tablename> on commit <...> as (<select_cmd>);

so, it is not the great solution, but a simple sourcecode extension:

Sqlexp.h:

```
struct SqlCreateTable {
  SqlId  table;
  bool   permanent;
  bool   transaction;
  enum { TRANSACTION, SESSION, PERMANENT };
public:
  SqlCreateTable& Permanent()   { permanent = true; return *this; }
  SqlCreateTable& Transaction() { transaction = true; return *this; }
  SqlStatement As(const SqlSelect& select);
  SqlCreateTable(SqlId table) : table(table) { permanent = false; transaction = false; }
};
```

SqlStatement.cpp:

```
SqlStatement SqlCreateTable::As(const SqlSelect& select)
{
  String text = "create ";
  if(!permanent)
    text << SqlCase(ORACLE | PGSQL, "global temporary ")("temporary ");
  text << "table " << table.Quoted();
  if(!permanent){
    if (transaction)
      text << SqlCase(ORACLE | PGSQL, " on commit delete rows")("");
    else
      text << SqlCase(ORACLE | PGSQL, " on commit preserve rows")("");
  }
  text << " as (" + SqlStatement(select).GetText() + ")";
  return SqlStatement(text);
}
```

Subject: Re: PATCH/BUGFIX Oracle global temporary table
Posted by mirek on Wed, 14 Oct 2015 10:55:54 GMT
View Forum Message <> Reply to Message

OK, applied, but Oracle 'temporary' tables seem to be quite different semantically, so this is really ambiguos solution. It seems to me that for Oracle, temporary table support should be rather part of .sch file...

Subject: Re: PATCH/BUGFIX Oracle global temporary table
Posted by wqcmaster on Wed, 14 Oct 2015 13:45:36 GMT
View Forum Message <> Reply to Message

Yes, that would be great

Something like that?:

sch_model.h

```
#ifndef SESSIONTABLE
#define SESSIONTABLE(x)                TYPE(x)
#endif

#ifndef TRANSACTIONTABLE
#define TRANSACTIONTABLE(x)            TYPE(x)
#endif
```

sch_schema.h

```
#define SESSIONTABLE(x)\
void TABLE_##x(SqlSchema& schema) { schema.SessionTable(#x); SCHEMA_##x(schema);
schema.EndTable(); }

#define TRANSACTIONTABLE(x)\
void TABLE_##x(SqlSchema& schema) { schema.TransactionTable(#x); SCHEMA_##x(schema);
schema.EndTable(); }
```

SqlSchema.cpp:

```
String SqlSchema::Expand(const char *txt, int i) const
{
...
case '<1>': r.Cat(<temp_table_attribute_variable1>); break;   //<1> -> a letter
case '<2>': r.Cat(<temp_table_attribute_variable2>); break;   //<2> -> a letter
...
}
```

```
void SqlSchema::SessionTable(const char *name) {
  FlushTable();
  table = name;
  table_suffix = Null;
  <temp_table_attribute_variable1> = "global temporary ";
  <temp_table_attribute_variable2> = "on commit preserve rows "
  Schema() << Expand("create @<1> table @t <2> (\n");
  SchemaDrop() << Expand("drop table @t;\n");
  Attributes() << '\n';
  AttributesDrop() << '\n';
  firstcolumn = true;
}

void SqlSchema::TransactionTable(const char *name) {
  FlushTable();
  table = name;
  table_suffix = Null;
  <temp_table_attribute_variable1> = "global temporary ";
  <temp_table_attribute_variable2> = "on commit delete rows "
  Schema() << Expand("create @<1> table @t @<2> (\n");
  SchemaDrop() << Expand("drop table @t;\n");
  Attributes() << '\n';
  AttributesDrop() << '\n';
  firstcolumn = true;
}
```

last but not least:

```
void SqlSchema::FlushColumn()
{
...
Upgrade() << Expand("create @<1> table @t @<2> ( ...
```

or it should be a completely different solution?