Хранилища

ORM

ES.40: Avoid complicated expressions

RDBMS

course

id	name
1	c++
2	python

group

id	name	course_id
1	2017-12	1
2	2018-02	1
3	2016-12	2
4	2017-12	2

DO

table	field	id	value
course	name	1	c++
course	name	2	python
group	course_id	1	1
group	course_id	2	1
group	course_id	3	2
group	course_id	4	2
group	name	1	2017-12
group	name	2	2018-02
group	name	3	2016-12
group	name	4	2017-12

KV

Key	Value
group_1	{“course”: {“id”: 1, “name”: “c++”}, “id”: 1, “name”: “2017-12”}
group_2	{“course_id”: 1, “id”: 2, “name”: “2018-02”}
group_3	{“course”: {“id”: 2, “name”: “python”}, “id”: 3, “name”: “2016-12”}
group_4	{“course”: {“id”: 2, “name”: “python”}, “id”: 4, “name”: “2017-12”}
course_1	{“id”: 1, “name”: “c++”}
course_2	{“id”: 2, “name”: “python”}

MR

```
$ hadoop fs -ls

630'505'450 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000067_0
632'146'221 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000068_0
629'464'052 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000069_0
631'540'595 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000070_0
634'579'190 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000071_0
632'425'581 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000072_0
633'255'459 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000073_0
```

```
631'412'163 YYY-MM-DD /user/.../sessions/dt=YYY-MM-DD/000074_0
```

```
$ hadoop fs -cat /user/.../sessions/dt=YYY-MM-DD/000067_0
```

```
{"http_user_agent":"PC / Windows 7 / Chrome 54.0.2840","mainvert":"ban"},"ruid":"0000001D5D3
{"http_user_agent":"Microsoft Internet Explorer / 6.0 | Windows","mainvert":"search","iruid"
{"http_user_agent":"Chrome / 56.0.2924.76 | Macintosh","mainvert":"search","iruid":"","spell
{"http_user_agent":"Firefox / 50.0 | Windows","mainvert":"search","iruid":"","spelled":"fals
{"http_user_agent":"Safari / 9.1.2 | Macintosh","mainvert":"search","iruid":"","spelled":"fa
{"http_user_agent":"Chrome / 56.0.2924.87 | Windows","mainvert":"search","iruid":"","spelled
{"http_user_agent":"Chrome / 64.0.3282.140 | Windows","mainvert":"search","iruid":"","spelle
{"http_user_agent":"Firefox / 50.0 | Windows","mainvert":"search","iruid":"","spelled":"fals
{"http_user_agent":"Firefox / 29.0 | Windows","mainvert":"search","iruid":"","spelled":"fals
{"http_user_agent":"Chrome / 54.0.2840.99 | Windows","mainvert":"search","iruid":"","spelled
```

SQL / MySQL Connector/C++

```
sql::mysql::MySQL_Driver *driver;
sql::Connection *con;
sql::Statement *stmt;

driver = sql::mysql::get_mysql_driver_instance();
con = driver->connect(
  "tcp://127.0.0.1:3306", "user", "password");

stmt = con->createStatement();
stmt->execute("USE " EXAMPLE_DB);
stmt->execute("DROP TABLE IF EXISTS test");
stmt->execute("CREATE TABLE test(id INT, label CHAR(1))");
stmt->execute("INSERT INTO test(id, label) VALUES (1, 'a\\'a'), (?, ?)", 2, "b'b"); // a'a, b'b

delete stmt;
delete con;
```

SQL / MySQL Connector/C++

```
stmt = con->createStatement();

res = stmt->executeQuery(
  "SELECT id, label FROM test ORDER BY id ASC");
while (res->next()) {
  cout << "id = " << res->getInt(1) << endl;
  cout << "label = " << res->getString("label") << endl;
}

delete res;
delete stmt;
```

ORM / ODB

```
#pragma db object
class person
{
  friend class odb::access;
  #pragma db id auto
  unsigned long id_;
public:
  const std::string& first () const {return first_; }
  const std::string& last () const {return last_; }
  unsigned short age () const {return age_; }
  void age (unsigned short age) {age_ = age; }
};
```

```
odb -d mysql --generate-query  --generate-schema person.hxx
```

```
/* This file was generated by ODB, object-relational mapping (ORM)
 * compiler for C++.
 */

DROP TABLE IF EXISTS `person`;

CREATE TABLE `person` (
  `id` BIGINT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
  `first` TEXT NOT NULL,
  `last` TEXT NOT NULL,
  `age` SMALLINT UNSIGNED NOT NULL)
 ENGINE=InnoDB;
```

ORM / ODB

```
person joe("Joe", "Dirt", 30);
joe_id = db->persist(joe);
```

```
const char access::object_traits_impl< ::person, id_mysql >::persist_statement[] =
"INSERT INTO `person` "
"(`id`, "
"`first`, "
"`last`, "
"`age`) "
"VALUES "
"(?, ?, ?, ?)";
```

ORM / ODB

```
result r (db->query<person> (query::age > 30));
for (result::iterator i (r.begin ()); i != r.end (); ++i) {
    cout << "Hello, " << i->first () << "!" << endl;
}
```

```
const char access::object_traits_impl< ::person, id_mysql >::query_statement[] =
"SELECT "
"`person`.`id`, "
"`person`.`first`, "
"`person`.`last`, "
"`person`.`age` "
"FROM `person`";
```

ORM / ODB

```
auto_ptr<person> joe(db->load<person>(joe_id));
joe->age(joe->age() + 1);
db->update(*joe);
```

```
const char access::object_traits_impl< ::person, id_mysql >::update_statement[] =
  "UPDATE `person` "
  "SET "
  "`first`=?, "
  "`last`=?, "
  "`age`=? "
  "WHERE `id`=?";
```

ORM / ODB

```
#pragma db view object(person)
struct person_stat
{
  #pragma db column("count(" + person::id_ + ")")
```

```
  std::size_t count;
  #pragma db column("min(" + person::age_ + ")")
  unsigned short min_age;
  #pragma db column("max(" + person::age_ + ")")
  unsigned short max_age;
};
```

Object-Relational Mapping (ORM) / ODB

```
query_statement (const query_base_type& q) {
  query_base_type r (
    "SELECT "
    "count(`person`.`id`), "
    "min(`person`.`age`), "
    "max(`person`.`age`) ");
  r += "FROM `person`";
  if (!q.empty ()) {
    r += " ";
    r += q.clause_prefix ();
    r += q;
  }
  return r;
}
```

Миграция

```
#pragma db object
class person
{
  friend class odb::access;
  #pragma db id auto
  unsigned long id_;
public:
  const std::string& first () const {return first_; }
+ const std::string& second () const {return second_; }
  const std::string& last () const {return last_; }
  unsigned short age () const {return age_; }
  void age (unsigned short age) {age_ = age; }
};
```

Data Access Object (DAO)

```
struct CTest {
  int id;
  std::string label;
}

class CTestDAO {
  void create(const CTest &test) {
    stmt->execute("INSERT INTO test (label) VALUES (?)", test.label);
  }
  void update(const CTest &test);
  CTest get(int id) {
    res = stmt->executeQuery(
      "SELECT id, label FROM test WHERE id=?", id);
    // ...
  }
  void remove(int id);
}
```

Repository / DDD

```
struct CTest {
  int id;
  std::string label;
  std::vector<CSlave> slaves;
}
class CTestRepository {
  void create(const CTest &test) {
    stmt->execute("INSERT INTO test (label) VALUES (?)", test.label);
    stmt->execute("INSERT INTO slave (test, value) VALUES (?, ?)", test.id, 42);
  }
  void update(const CTest &test);
  CTest get(const CTest &test) {
    res = stmt->executeQuery(
      "SELECT id, label FROM test WHERE id=?", test.id);
    // ...
  }
  void remove(const CTest &test);
}
```

Почитать

- https://codesynthesis.com/products/odb/
- Предметно-ориентированное проектирование. Структуризация сложных программных систем (Эрик Эванс)

Спасибо за внимание!