# Stored Procedures as an Implementation Tool Business Logic in Applications of Databases*

Tatyana N. Filimonenkova[1][0000-0002-7968-9028]

[1] V.I. Vernadsky Crimean Federal University, Yalta, Russia
tafil-nik@yandex.ru

**Abstract.** The article discusses a database object such as a stored procedure and the advantages of using such an approach when developing software in a client-server architecture. The definition of stored procedures is given, the features of their creation and compilation are listed. The syntax of creating stored procedures in the client-server database management system MySQL is considered. The examples of stored procedures developed for the database for accounting student performance, which demonstrate the feasibility of their use and advantages compared with the usual queries are given. The article provides an example of using the stored procedure on the client side of an application developed in the Embarcadero RAD Studio integrated environment using BDE (Borland Database Engine) technology as a tool for accessing the database.

**Keywords:** information systems, client-server architecture, databases, CASE, ER-modeling, SQL-query, stored procedure, MySQL.



## 1 Introduction

In applications that implement the client-server architecture, and now, this is the most common architecture for both web-based and desktop applications, data processing is distributed in such a way that the data presentation program is located on the user's machine (on the client), and the data management program and the data itself are placed on the server. The server software accepts requests from client software and returns to it the results of processing these requests.

The popularity of client-server technology is inextricably linked with IBM's invention of the query language to SQL relational databases – Structured Query Language, which is currently the universal standard for working with databases. Despite the fact that due to the growing amount of data and their complication, a new approach to data storage and processing is being developed – the NoSQL technology [8], still in the official DB-Engines rating (https://db-engines.com/en/ranking) as of May 2019, the first four places are occupied by relational databases: Oracle, MySQL, Microsoft SQL Server, PostgreSQL.

In client-server applications, it is the server that is responsible for managing the database, and client machines may have different applications using this data. Specialized software connects the client and the server, allowing the client to perform requests and access the database. A comparative analysis of popular client-server database management systems (DBMS) is given in [4]. One of the tools that provides the client-server architecture is an element of the SQL language as stored procedures.

## 2 Stored procedures as a data processing tool

The stored procedure is a compiled routine and is part of the database metadata. The stored procedure compiler is built into the server core. It is the use of stored procedures that allows implementing most of the application's business logic at the server level, which significantly improves application performance, centralizes data processing and minimizes errors, ensuring data integrity [1].

Stored procedures are a SQL tool that allows you to create queries with parameters first, because when creating a procedure, you can set a list of input parameters, which ensures that the corresponding arguments are accepted when it is called. Stored procedures can return information that the user expects, as well as in the case of incorrect data - an error message. Considering the fact that the stored procedure is compiled before it is stored as an object in the database, the form of the procedure is saved each time it is called. This fact reduces the amount of data involved in the exchange between the client application calling the procedure and the database server. The use of stored procedures significantly increases the level of security provided by the use of the GRANT and REVOKE instructions, by which users are granted different access privileges, since procedures can be used to control access authorization [9]. Thus, the user on the client side performs only the functions that are allowed to him. This allows you to tightly control the actions allowed by the client, what turn reduces the likelihood of data integrity problems due to client application errors.

Stored procedures are used in other tasks besides the ones stated above. For example, in [6], the use of MySQL stored procedures for working with iterators OCL (Object Constraint Language), a language that serves to define constraints and is designed to create navigation expressions, logical and other queries, is considered.

The syntax for creating and calling stored procedures is somewhat different in different client-server database management systems.

Consider the examples of stored procedures for the MySQL server, which is widely used both in web-based applications and in desktop ones. Not only hundreds of thousands of commercial websites work on this base, but also a large number of corporate applications use MySQL as a server-based DBMS. The MySQL distribution contains API (Application Programming Interface) modules for interacting with such programming languages as C, C ++, Java, PHP, Python, etc. MySQL provides support for various character sets, including Cyrillic. To integrate into the operating system and to access the database from the application software, the MySQL distribution kit contains a set of Open Database Connectivity (ODBC) drivers.

Stored procedures, like other database objects, are created using the DDL (Data Definition Language) language.

The general syntax for creating a stored procedure in MySQL is:

```
CREATE [DEFINER = { user | CURRENT_USER }]
PROCEDURE имя_процедуры ([параметры_процедуры[,...]])
[характеристики ...] тело_подпрограммы
параметры_процедуры: [ IN | OUT | INOUT ] имя_параметра
type
type: Любой валидный тип данных MySQL
характеристики: COMMENT 'string'
| LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA
| MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
тело_подпрограммы:  Валидный оператор программы SQL
Call the stored procedure by the operator:
CALL имя_процедуры([параметры_процедуры[,...]])
```

## 3 Programming stored procedures on the server

As an example, consider the task of accounting student performance. The development of an information system begins with the construction of a conceptual database model. As a modeling tool, the model proposed by P. Chen in 1976 [5] is used. This model has been called the “entity-relationship” – (ER-model) and is a set of concepts for describing the logical structure of the database. This model is used for a wide range of tasks, as shown in [3, 7, 10] and for various DBMS [2].

The basic concepts of the ER-model – is the "entity" and "relationship." An entity is any object in the real world or a process that is significant in the context of the task, in fact it is a class of similar objects, information about which should be taken into account. Each entity is identified by a name and a set of attributes that characterize it. An entity key is a set of attributes that uniquely identify each entity instance. An entity is a projection of the notion “relation” of the relational model. Connection is a relationship of entities that defines their functional dependence. In this case, one of the entities acts as a parent in relation to the other, and the second – as a child. Between them there are two main types of links – identifying and non-identifying. When the type of link is identifying, the primary key of the parent entity becomes the primary key or part of the composite primary key of the child. With a non-identifying relationship, the primary key of the parent entity becomes the child's foreign key as one of the non-key attributes. As it is known from the conceptual apparatus of the relational model, the primary key of the entity is responsible for ensuring the categorical integrity of the database, which by definition cannot have a NULL value, i.e. have an undefined value. Relationships provide referential integrity, which means that no foreign key value in a child entity can have a value that is missing among the primary key values of the parent entity.

Several methodologies for constructing "entity-relationship" models are known. The most widespread methodology is IDEF1X.

The CASE-tool (Computer Aided Software / System Engineering) AllFusion ERwin Data Modeler is used as a modeling tool.

The logical model of database "student" is presented below (Fig. 1).

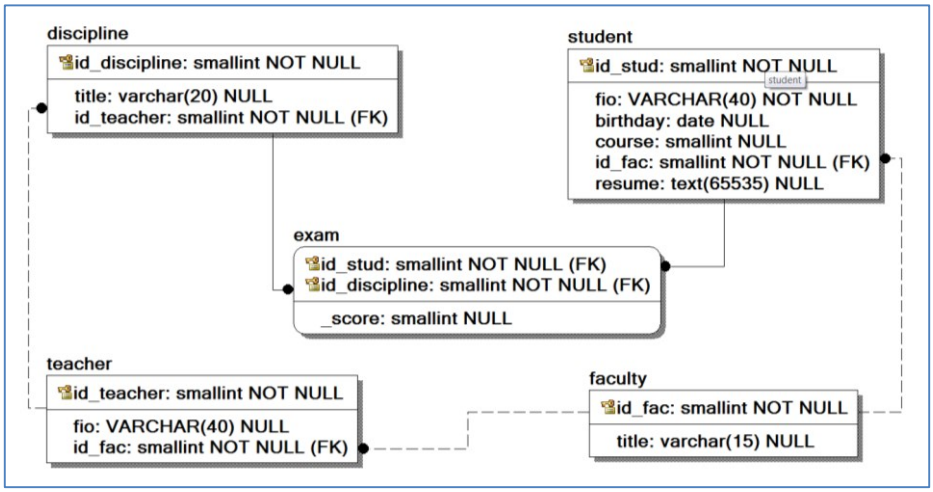


**Fig. 1.** ER-diagram of database "student"

The model has five entities: student (student), department (faculty), discipline (discipline), teacher (teacher), academic performance (exam). Relations between entities are determined by the following rules: each student studies at a department, each teacher works at a department, a teacher teaches every discipline, each student pass several disciplines, and several students pass each discipline. This many-to-many relationship is implemented as an entity "exam", which is identified through its foreign keys from the student and discipline entities and is subordinate.

The AllFusion ERwin Data Modeler package allows you not only to build a database model, but also to generate a database: tables with key fields and implement referential integrity constraints on foreign keys. To do this, you can use the “Forward Engineer / Schema Generation ...” command from the “Tools” menu.

The following are SQL instructions for creating tables of database "student".

```
CREATE TABLE discipline(
  id_discipline  smallint NOT NULL,
  title  varchar(20) NULL,
  id_teacher  smallint NOT NULL );
CREATE UNIQUE INDEX PRIMARY ON discipline(
  id_discipline );
ALTER TABLE discipline ADD  PRIMARY KEY (id_discipline);
CREATE INDEX id_teacher ON discipline ( id_teacher);
```

```
CREATE TABLE exam(
  id_stud  smallint NOT NULL,
  id_discipline  smallint NOT NULL,
  _score  smallint NULL );
CREATE UNIQUE INDEX PRIMARY ON exam(
  id_stud,
  id_discipline);
ALTER TABLE exam ADD  PRIMARY KEY (id_stud,id_disci-
pline);
CREATE INDEX id_subj ON exam (id_discipline);

CREATE TABLE faculty(
  id_fac  smallint NOT NULL,
  title  varchar(15) NULL );
CREATE UNIQUE INDEX PRIMARY ON faculty( id_fac);
ALTER TABLE faculty ADD  PRIMARY KEY (id_fac);

CREATE TABLE student(
  id_stud  smallint NOT NULL,
  fio  VARCHAR(40) NOT NULL,
  birthday  date NULL,
  course  smallint NULL,
  id_fac  smallint NOT NULL,
  resume  text(65535) NULL );
CREATE UNIQUE INDEX PRIMARY ON student(
  id_stud );
ALTER TABLE student
  ADD  PRIMARY KEY (id_stud);
CREATE INDEX id_fac ON student (id_fac);

CREATE TABLE teacher(
  id_teacher  smallint NOT NULL,
  fio  VARCHAR(40) NULL,
  id_fac  smallint NOT NULL );
CREATE UNIQUE INDEX PRIMARY ON teacher(
  id_teacher );
ALTER TABLE teacher
  ADD  PRIMARY KEY (id_teacher);
CREATE INDEX id_fac ON teacher( id_fac );

ALTER TABLE discipline
  ADD FOREIGN KEY subject_ibfk_1 (id_teacher)
REFERENCES teacher(id_teacher);

ALTER TABLE exam
```

```
  ADD FOREIGN KEY exam_ibfk_3 (id_stud)
REFERENCES student(id_stud)ON DELETE CASCADE;

ALTER TABLE exam
  ADD FOREIGN KEY exam_ibfk_2 (id_discipline)
REFERENCES discipline(id_discipline);

ALTER TABLE student
  ADD FOREIGN KEY student_ibfk_1 (id_fac)
REFERENCES faculty(id_fac);

ALTER TABLE teacher
  ADD FOREIGN KEY teacher_ibfk_1 (id_fac) REFERENCES
faculty(id_fac);
```

The stored procedure is associated with a specific database, so before creating it, you must go to this database using the standard instruction: "use database_name".

Consider an example of a procedure that allows you to get the results of intermediate certification of a particular student by his last name, including the title of the disciplines that he passed, and the resulting assessments.

```
delimiter ^
create procedure stud_fio_scores(in in_fio varchar(20))
begin

declare stud_id integer;

select id_stud into stud_id from student where
fio=in_fio;
select id_stud, title, _score from exam inner join disci-
pline using (id_discipline) where id_stud=stud_id;

end; ^
```

Since inside the procedure operators are separated by a semicolon, and in SQL this character is a separator for closing an operator, the first instruction of the procedure sets another separator character. In this case, this "^". Quite often, the symbol "//" or "$" is used as a separator.

As an input parameter, the procedure takes the student's last name, according to which the corresponding primary key of the student table id_stud is determined using the query. This value is stored in a local variable, which is declared inside the procedure by the declare statement. As students are identified in the table with the results of the exams using the key field (id_stud), it is this calculated parameter that is transmitted to the second query of the procedure body, allowing you to get the desired result.

The following stored procedure allows you to get a list of students who have passed the session on the "good" and "excellent", which can apply for scholarships. The procedure displays information in the format of last name of student, course and name of faculty.

```
delimiter ^
create procedure student_score_not3( )
begin

select fio, course, title from exam inner join (student
inner join faculty using(id_fac)) using(id_stud) group by
id_stud having min(_score)>3;

end; ^
```

Below is an example of a stored procedure that determines the student with the highest average score (if there are several such students, the entire list will be displayed) and the student with the lowest score based on the results of the intermediate attestation.

```
delimiter ^
create procedure max_min_avg()
begin

declare min_ball  double(6,4);
declare max_ball double(6,4);

select max(sr_b) into max_ball from sr_ball;
select min(sr_b) into min_ball from sr_ball;
 select fio, avg(_score) as max_sr from student, exam
where exam.id_stud=student.id_stud group by exam.id_stud
having max_sr >=max_ball;
 select fio, avg(_score) as min_sr from student, exam
where exam.id_stud=student.id_stud group by exam.id_stud
having min_sr <=min_ball;

end; ^
```

Creating a complex of procedures that implement the basic business processes ensures reliable operation of the information system.

## 4 Using stored procedures in the client part of the application

The client-server MySQL DBMS, like other SQL servers, does not have a graphical user interface. Consider the technology of using stored procedures in the client part of the application for MySQL in the C++ programming language in the Embarcadero

RAD Studio integrated software development environment - the Rapid Application Development (RAD) environment. Combining powerful tools of the C ++ programming language with the library of visual components provides the programmer with the tools for the rapid development of a graphical interface [11].

Embarcadero RAD Studio provides a programmer with several technologies for developing database applications. Consider one of them - BDE (Borland Database Engine). The main components of the BDE technology that provide the data set are such components as TTable (table), TQuery (query) and TStoredProc (stored procedure). There is a large set of visualization and data management components, such as DBGrid (grid), DBNavigator (navigator), DBText (label), DBEdit (input field), etc. The DataSource component is used to exchange information between data sets and visual components. - data source. The connection of components providing access to the database is shown in Fig. 2.

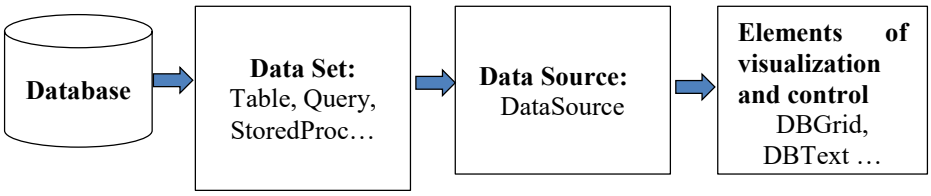


**Fig. 2.** The scheme of interaction of BDE components with the database

Consider a specific example of using a stored procedure in the client part of an application.

As an example, take a procedure that displays information about student results of the examination session. As shown above, the procedure is stored on the server under the name stud_fio_scores. The input parameter of the procedure is the last name of the student.

The data access technology through the StoredProc component requires setting the following properties: DataBaseName — selected from the list of available databases; in the StoredProcName property, the required stored procedure is selected from the set of stored procedures of this database. Since stored procedures often have input parameters, it is necessary to select the parameter binding mode, for which the ParamBindMode property is responsible, in which the choice by name or index is available. The data type and other characteristics of the parameters of the stored procedure are available by the Params property, in which you can specify a specific value of the parameter already at the application development stage. In most cases, the value of the input parameter is set or calculated during program execution (runtime).

In this example, in addition to the StoredProc component, you will need to use the TTable (TStudent) component to get a list of student names. This information will be displayed in such a visual component as the ComboBox drop-down list (CBFio_stud). This is implemented by the following code snippet.:

```
void __fastcall TForm1::FormActivate(TObject *Sender)
{
```

```
TStudent->First();
TStudent->Active=true;
CBFio_stud->Clear();
while(!TStudent->Eof)
{
  CBFio_stud->Items->Add(TStudentfio->AsString) ;
  TStudent->Next();
}
CBFio_stud->ItemIndex=0;
TStudent->First();
}
```

The selection of the input parameter, the call of the stored procedure and the display of the results of its work is implemented by the following fragment:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   StoredProc1->Active=false;
   StoredProc1->ParamByName("in_fio")->AsString =
CBFio_stud->Text;
   StoredProc1->Prepare();
   StoredProc1->Active=true;
}
```

In Fig. 3 shows an example of a call to a stored procedure that displays information about a particular student's grades with an indication of the name of the discipline.

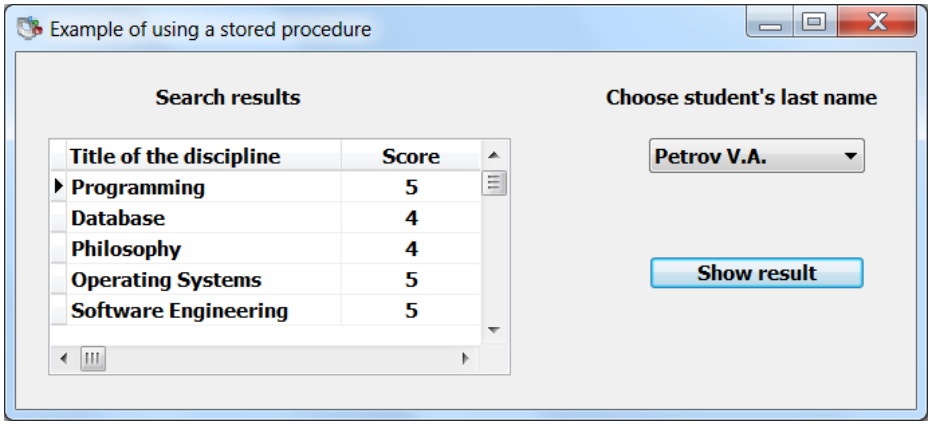


**Fig. 3.** An example of using a stored procedure

The presented information, considered examples clearly demonstrate the advantage of using stored procedures as compared to the usual queries, since it allows you to receive data with any specific input parameter.

If using dynamic SQL, reusing a query with a different input parameter would be difficult. In addition to improving performance, since stored procedures are usually faster than regular SQL statements, they make it easier to perform repetitive tasks, allowing you to output complex operations into a single object. The stored procedure code is compiled once and then stored in compiled form.

## Conclusion

Based on the considered examples of creating stored procedures on a server and their use in client applications, it can be concluded that the use of a database object such as a stored procedure in information systems simplifies program maintenance and changes to it. Being independent of tables or any other database objects, stored procedures accumulate in themselves all integrity constraints in the form of rules and data processing algorithms.

They are implemented on the server and the end user is provided only with the data processing interface in the form of a call to certain stored procedures. The mechanism for passing parameters to stored procedures provides the multifunctionality of the query to obtain the necessary data.

## References


1. Andreeva, Natalia Viktorovna. Database programming: SQL. Data selection [Electronic resource]: study guide / N. V. Andreeva, O. Yu. Sabinin; - St. Petersburg Polytechnic University of Peter the Great. St. Petersburg, 2018. Available at: http://elib.spbstu.ru/dl/2/s18-64.pdf. (In Russian) DOI: 10.18720/SPBPU/2/s18-64
2. Babanov Alexey M., Petrov Alexander V. Implementation of the ERM-model repository in CASE-system Oracle Designer. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika* [Tomsk State University Journal of Control and Computer Science], 2017. - No.41, pp. 47-54. (In Russian) DOI: 10.17223/19988605/41/6
3. Babanov, A. M. Database design prospects opening with application of modern semantic data models. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naya tekhnika i informatika* [Tomsk State University Journal of Control and Computer Science], 2015. - No. 2. pp. 73–80. (In Russian). DOI: 10.17223/19988605/31/8
4. Borovskoy I.G., Shelmina E.A. Comparative analysis of desktop and client-server databases. Reports of Tomsk State University of Control Systems and Radioelectronics. 2017 - pp. 92-94. (In Russian). DOI: 10.21293/1818-0442-2017-20-4-92-94
5. Chen Peter. The Entity-Relationship Model – Toward a Unified View of Data // ACM Transactions on Database Systems, 1976, 1(1): 9–36. DOI: 10.1145/320434.32044
6. Egea M., Dania C., Clavel M. MySQL4OCL: A Stored Procedure-Based MySQL Code Generator for OCL // Electronic Communications of the EASST. 2010. - vol.36, 16p. DOI:10.14279/TUJ.ECEASST.36.445
7. Kharitonov D.I., Tarasov G.V., Leontiev D.V., Parakhin R.V. Modeling the subject area for the formation of electronic collections. Territory of new opportunities. Bulletin of the Vladivostok State University of Economics and Service. 2018. - vol. 10, No.2. pp. 125–136. (In Russian). DOI: 10.24866/VVSU/2073-3984/2018-2/125-136

8. Koroleva Yu.A., Maslova V.O., Kozlov V.K. Development of the concept of data migration between relational and non-relational database systems // International Scientific and Practical Journal Software & Systems. 2019. - vol. 32, No. 1, pp. 063–067. (In Russian). DOI: 10.15827/0236-235X.125.063-067
9. Poltavtsev A.A. Dynamic structures in relational databases // International Scientific and Practical Journal Software & Systems. 2015. - No. 2, pp.95-97. DOI:10.15827/0236-235X.110.095-097
10. Posevkin R.V. Database semantic model application in natural language user interface development process. Scientific and Technical Journal of Information Technologies, Mechanics and Optics. 2018. - vol. 18, No. 2, pp. 262–267 (In Russian). DOI: 10.17586/2226-1494-2018-18-2-262-267
11. Sazonov, A.D. Application of the MYSQL DBMS to create information systems in the client-server architecture: final qualification work of the bachelor: 09.03.02 - Information Systems and Technologies, St. Petersburg Polytechnic University of Peter the Great. 2018. (In Russian). DOI: 10.18720 / SPBPU / 2 / V18-6487