УДК 004.65:004.652

ОСОБЕННОСТИ ХРАНЕНИЯ ИЗОБРАЖЕНИЙ В РЕЛЯЦИОННЫХ СУБД

Одиноков Д.А., Артюшина Е.А.
Пензенский государственный технологический университет, Пенза, e-mail: Alexey314@yandex.ru

В настоящий момент большое количество существующих программных приложений в процессе своего функционирования используют базы данных для организованного хранения информации, а также её чтения, записи или обработки. Одна из популярных задач в управлении данными - это операции с большими двоичными массивами (BLOB), которые обычно представляют собой растровые изображения, аудио или другие мультимедийные объекты. Поддержка СУБД для BLOB-объектов не всегда является универсальной. Цель научно-исследовательской работы студента - освоение технологий хранения изображений в реляционных системах управления базами данных. Список задач, решаемых в ходе исследования, включает в себя: изучение особенностей хранения изображений в традиционных OLTP-системах с различными архитектурами; выбор инструментов для разработки баз данных; проектирование логической модели данных в выбранном формате, а также создание тестового программного приложения для хранения изображений в интегрированной среде MS Visual Studio 2017 на объектно-ориентированном языке C#. Традиционные реляционные модели сегодня активно изменяются, чтобы соответствовать современному состоянию рынка программного обеспечения. В данной статье рассматривается подходящий инструментарий СУБД и критерии для выбора оптимального решения, наилучшим образом удовлетворяющего конкретным нуждам заказчика.

Ключевые слова: архитектуры для обработки транзакций, хранение изображений, реляционные СУБД

FEATURES OF IMAGE STORAGE IN RELATIONAL DBMS

Odinokov D.A., Artyushina E.A.
Penza State Technological University, Penza, e-mail: Alexey314@yandex.ru

At the moment, a large number of existing software applications in the course of their operation use databases for organized storage of information, as well as its reading, recording or processing. One of the most popular tasks in the management of data – operations with binary large objects (BLOB), which are usually bitmaps, audio or other multimedia object. Database support for blobs is not always universal. The purpose of the research work of the student is the development of image storage technologies in relational DBMS. The tasks solved in the course of the study include: the study of the features of image storage in traditional OLTP-systems with different architectures; the choice of tools for database development; the design of a logical data model in the selected format; the creation of a test application for storing images in the integrated environment MS Visual Studio 2017 (C#). Established relational models are changing to keep pace. In this article are considered practical tools for navigating this shifting product landscape and finding candidate systems that best fit a data managers applications needs.

Keywords: architecture for transaction processing, image storage, relational database management system

Целью научно-исследовательской работы является освоение технологий хранения изображений в реляционных СУБД. К задачам, решенным в ходе исследования, относятся: 1) изучение особенностей хранения изображений в традиционных СУБД с различной архитектурой; 2) выбор инструментальных средств для разработки базы данных; 3) проектирование логической модели данных в выбранном формате [1]; 4) создание тестового приложения для хранения изображений в IDE MS Visual Studio 2017 (язык C#).

Логическая структура БД туристического агентства представлена на рис. 1.

Реляционные СУБД разнообразной архитектуры предусматривают множество вариантов хранения графической информации. Так, например, в таблицах файл-серверной СУБД Microsoft Access нетекстовая информация (фото, графики и диаграммы, формулы, рисунки и пр.) содержится в полях-контейнерах OLE.

Поле объекта OLE (Object Linking and Embedding) является средством, позволяющим установить связь с объектами другого приложения или внедрить нетекстовый объект в базу данных. В соответствии с этой технологией внедренным или связанным OLE-объектом является документ или его фрагмент, созданный с помощью другого приложения, поддерживающий формат документа-источника, а также информацию об OLE-сервере. Внедренный OLE- объект может сохраняться непосредственно в файле базы данных. Для обработки подобных полей в SQL-клиенте, который спроектирован, например, с помощью IDE Embarcadero Delphi, можно использовать компонент OleContainer. Он принимает объект OLE, который загружается из графического файла стандартным методом CreateObjectFromFile. Используя этот контейнер, можно отобразить в клиентском приложении любую фотографию, которая хранится в базе данных в поле OLE.

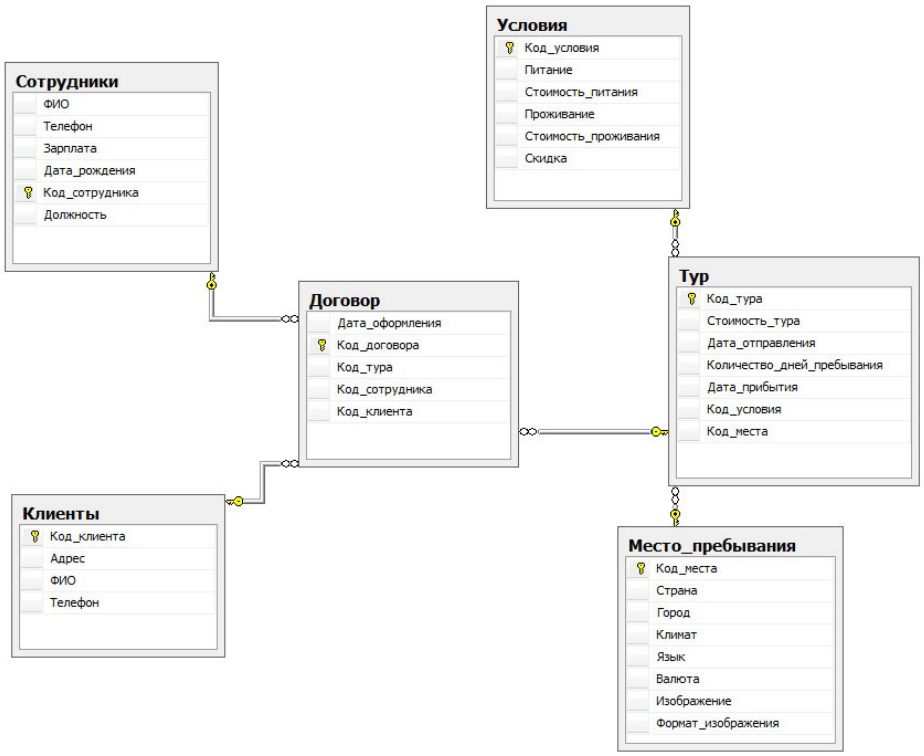

Рис. 1. Диаграмма БД «Туристическое агентство»

Популярная ныне СУБД SQLite со встраиваемой архитектурой не рассчитана на коллективное использование в сети и может поставляться как составная часть IDE, например, Android Studio. Наиболее часто СУБД SQLite используется в мобильных приложениях для локального хранения данных клиента, в том числе – изображений различных форматов. Для этого служат поля специфического типа BLOB (Binary Large Object).

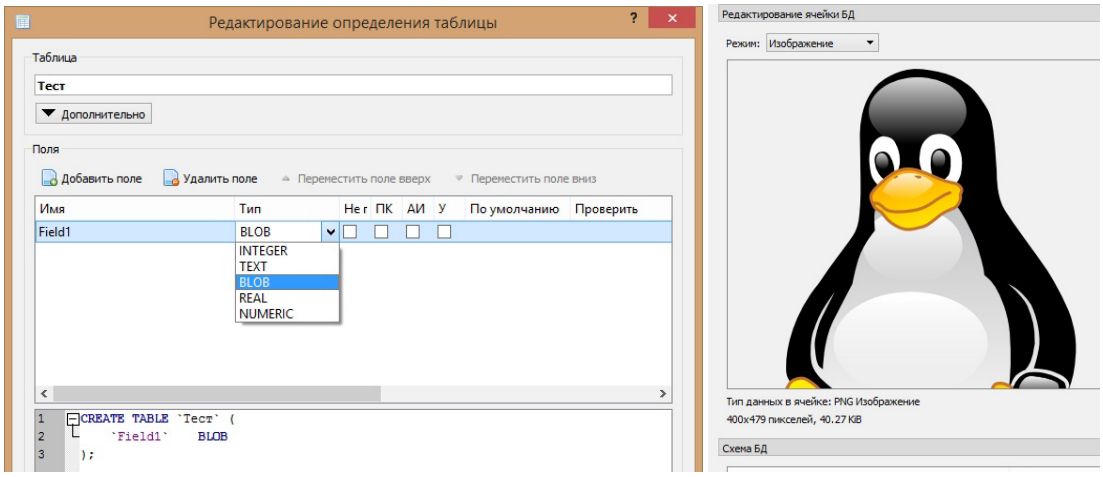

Рис. 2. Конструктор таблицы Test в администраторе DB Browser for SQLite

Как видно из рис. 2, среда администрирования DB browser for SQLite имеет универсальные встроенные средства для просмотра двоичных объектов BLOB. Однако при проектировании любого, даже элементарного Android-приложения для работы с БД программисту придется самостоятельно обеспечивать доступ к изображениям, хранящимся в локальной базе. Это возможно сделать с помощью метода execSQL, который поддерживает выполнение обычных SQL-запросов на манипулирование данными, либо используя стандартные библиотеки, к примеру – метод getBlob класса Cursor.

В клиент-серверной реляционной СУБД Microsoft SQL Server также имеются различные возможности для сохранения изображений. Можно хранить фотографии непосредственно в базе, используя для этого подходящий системный тип (например, image). Или же можно хранить в БД лишь ссылки на изображения (filestream), которые, в свою очередь, будут располагаться в заданных каталогах файловой системы компьютера-сервера.

В настоящем исследовании было принято решение выбрать для разработки первый вариант, так как использовать тип filestream рекомендуется при среднем объеме файлов свыше 1 Мб, а в нашем случае изображения приемлемого качества имеют средний объём ≈100 Кб [2]. Изображения будут храниться в бинарном коде в ячейках таблицы с типом image. Для апробации предлагаемого способа обработки изображений в СУБД MS SQL Server 2008 R2 Express Edition была создана тестовая таблица report [3, 4].

Конструктор таблицы представлен ниже на рис. 3.

Чтобы обеспечить загрузку в базу данных изображений, например, с локального диска компьютера и, наоборот, получить это изображение из базы, использована интегрированная среда разработки программ Microsoft Visual Studio 2017. Для этого на объектно-ориентированном языке C# [5] было реализовано два метода, для загрузки и получения изображений из БД, соответственно.

Так, в методе PutImageBinaryInDb, который используется для записи изображения в БД, содержатся операторы для преобразова-

ДМИТРИЙ-ПК\SQL...o1 - dbo.report

Column Name	Data Type	Allow Nulls
id	bigint	☐
screen	image	☑
screen_format	varchar(5)	☑
		☐

Рис. 3. Конструктор тестовой таблицы report

```
private static void PutImageBinaryInDb(string iFile)
{
    byte[] imageData = null;
    FileInfo fInfo = new FileInfo(iFile);
    long numBytes = fInfo.Length;
    FileStream fStream = new FileStream(iFile, FileMode.Open, FileAccess.Read);
    BinaryReader br = new BinaryReader(fStream);
    imageData = br.ReadBytes((int)numBytes);
    string iImageExtension = (Path.GetExtension(iFile)).Replace(".", "").ToLower();
```

Рис. 4. Конвертация файла изображения в бинарный код

ния исходного изображения в бинарный код и получения расширения файла. Расширение необходимо, чтобы в дальнейшем можно было бы работать с изображениями разных форматов. Код метода представлен ниже на рис. 4.

Далее в этом же методе связываемся с локальной базой данных и записываем изображение в текущую строку таблицы report, соответствующий фрагмент кода представлен далее на рис. 5.

Помещаем вызов метода PutImageBinaryInDb в тело главного метода Main, запускаем тестовое приложение на C# и в результате его выполнения получаем новую запись в таблице report, представленную на рис. 6.

Для обратного процесса получения изображения из базы данных будем использовать метод GetImageBinaryFromDb (рис. 7).

```
using (SqlConnection sqlConnection = new SqlConnection("user id=Дмитрий-ПК\\Дмитрий;" +
                            "server=ДМИТРИЙ-ПК\\SQLEXPRESS;" +
                            "Trusted_Connection=yes;" +
                            "database=Туристическое агентство1; " +
                            "connection timeout=15")) // строка подключения к БД
{
    string commandText = "INSERT INTO report (screen, screen_format) VALUES(@screen, @screen_format)";
    SqlCommand command = new SqlCommand(commandText, sqlConnection);
    command.Parameters.AddWithValue("@screen", (object)imageData); // записываем само изображение
    command.Parameters.AddWithValue("@screen_format", iImageExtension); // записываем расширение изобр
    sqlConnection.Open();
    command.ExecuteNonQuery();
    sqlConnection.Close();
}
```

Рис. 5. Загрузка изображения в БД MS SQL Server

ДМИТРИЙ-ПК\SQL...o1 - dbo.report

	id	screen	screen_format
	1	<Binary data>	jpg
▸*	NULL	NULL	NULL

Рис. 6. Результат загрузки бинарного кода изображения в таблицу report

```
private static void GetImageBinaryFromDb()
{
    List<byte[]> iScreen = new List<byte[]>();
    List<string> iScreen_format = new List<string>();
    using (SqlConnection sqlConnection = new SqlConnection("user id=Дмитрий-ПК\\Дмитрий;" +
                            "server=ДМИТРИЙ-ПК\\SQLEXPRESS;" +
                            "Trusted_Connection=yes;" +
                            "database=Туристическое агентство1; " +
                            "connection timeout=15"))
    {
        sqlConnection.Open();
        SqlCommand sqlCommand = new SqlCommand();
        sqlCommand.Connection = sqlConnection;
        sqlCommand.CommandText = @"SELECT [screen], [screen_format] FROM [report] WHERE [id] = 1";
        SqlDataReader sqlReader = sqlCommand.ExecuteReader();
        byte[] iTrimByte = null;
        string iTrimText = null;
        while (sqlReader.Read())
        {
            iTrimByte = (byte[])sqlReader["screen"]; // читаем строки с изображениями
            iScreen.Add(iTrimByte);
            iTrimText = sqlReader["screen_format"].ToString().TrimStart().TrimEnd(); // читаем строки
            iScreen_format.Add(iTrimText);
        }
        sqlConnection.Close();
    }
```

Рис. 7. Получение из БД строки с изображением и его форматом

В приведенном выше фрагменте кода подключаемся к базе данных и создаем запрос на выборку SELECT, возвращающий из таблицы report запись со значением уникального поля id = 1. Полученный из БД двоичный код помещаем в целочисленный массив для последующей обработки.

Далее конвертируем бинарный код обратно в изображение и сохраняем файл на диск, программный код приведен на рис. 8.

```
// конвертируем бинарные данные в изображение
byte[] imageData = iScreen[0];
MemoryStream ms = new MemoryStream(imageData);
Image newImage = Image.FromStream(ms);
// сохраняем изоражение на диск
string iImageExtension = iScreen_format[0]; // получаем расширение текущего изображения хранящееся в БД
string iImageName = "D:\\images\\result_new" + "." + iImageExtension; // задаём путь сохранения и имя нового изображения
if (iImageExtension == "png") { newImage.Save(iImageName, System.Drawing.Imaging.ImageFormat.Png); }
else if (iImageExtension == "jpg" || iImageExtension == "jpeg") { newImage.Save(iImageName, System.Drawing.Imaging.ImageFormat.Jpeg); }
else if (iImageExtension == "gif") { newImage.Save(iImageName, System.Drawing.Imaging.ImageFormat.Gif); }
```

Рис. 8. Конвертация и сохранение изображения на диск

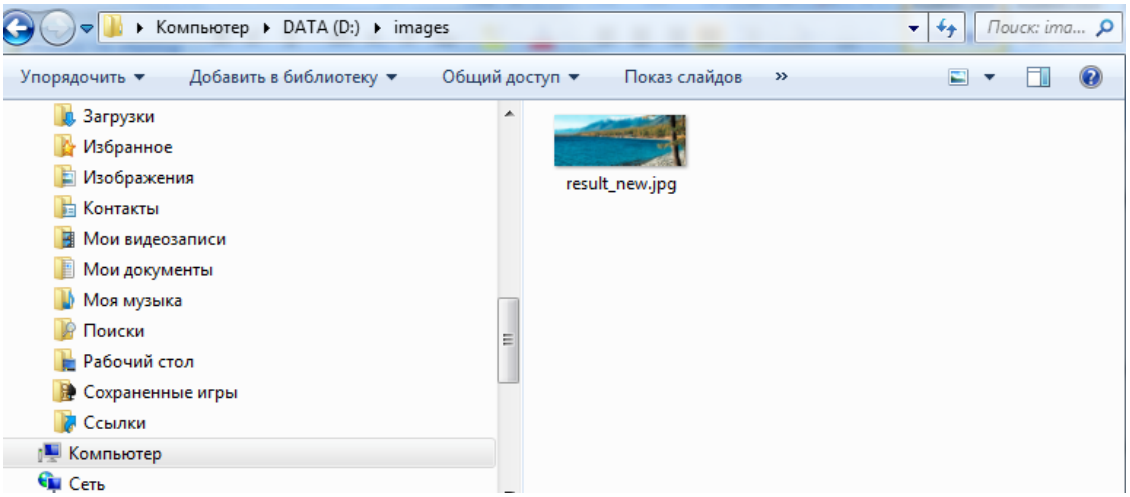

Рис. 9. Файл с изображением, полученным из БД

При вызове метода GetImageBinaryFromDb из точки входа Main тестовой программы, изображение успешно сохраняется в заданном каталоге images на локальном диске D:\ в файле result_new.jpg, как это показано ниже на рис. 9.

Таким образом, на основе вышеизложенного материала можно сделать вывод, что для хранения графической информации в данной работе вполне может быть использован тип image, входящий в набор системных типов реляционной клиент-серверной СУБД Microsoft SQL Server 2008 R2 Express Edition при условии соблюдения ограничений на средний объем изображений до 1 Мб.

Список литературы

1. Дейт К.Дж. Введение в системы баз данных. 6–е изд. – К.; М., СПб.: «Вильямс», 2015. – 848 с.

2. Оти М. Типы данных для больших объектов в SQL Server. [Электронный ресурс]: URL: https://www.osp.ru/winitpro/2011/01/13008224 (дата обращения: 05.02.2019).

3. Артюшина Е.А. Использование облачных технологий при изучении СУБД MS SQL Server/ Е.И. Маркин, К.М. Рябова, Е.А. Артюшина // Современные методы и средства обработки пространственно-временных сигналов: Сборник статей XIV Всероссийской научно-технической конференции. – Пенза: ПДЗ, 2016. – С. 89–94.

4. Артюшина Е.А. Проблемы использования облачных технологий при изучении MS SQL Server / С.А. Жулев, А.Е. Ведюшкина, Е.А. Артюшина // Международный студенческий научный вестник: Электронный научный журнал. – 2015. – №3(2) – С. 271.

5. Гринштейн С. История языков программирования: C# – впереди планеты всей. [Электронный ресурс]. – URL: https://habr.com/ru/post/313694/(дата обращения: 05.02.2019).