МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных

технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Аукционная площадка»

Выполнил студент Ермакович Виталий Сергеевич

(Ф.И.О.)

Руководитель проекта пр.-стаж. Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: пр.-стаж. Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: пр.-стаж. Нистюк О.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

Оглавление

[Введение 3](#_Toc58059887)

[1 Постановка задачи 4](#_Toc58059888)

[**1.1 Обзор аналогов** 4](#_Toc58059889)

[**1.2 Анализ прототипов** 6](#_Toc58059890)

[**1.3 Описание используемых технологий** 6](#_Toc58059891)

[2 Проектирование модели базы данных 7](#_Toc58059892)

[3 Разработка необходимых объектов 8](#_Toc58059893)

[**3.1 Таблицы** 8](#_Toc58059894)

[**3.2 Процедуры** 10](#_Toc58059895)

[4 Описание процедур импорта и экспорта данных 12](#_Toc58059896)

[**4.1 Процедура импорта данных из XML-файла** 12](#_Toc58059897)

[**4.2 Процедура экспорта данных в формате XML** 12](#_Toc58059898)

[5 Описание технологии 13](#_Toc58059899)

[6 Тестирование 17](#_Toc58059900)

[**6.1 Тестирование производительности базы данных** 17](#_Toc58059901)

[Заключение 19](#_Toc58059902)

[**Список литературных источников** 20](#_Toc58059903)

[Приложение A 21](#_Toc58059904)

[Приложение Б 22](#_Toc58059905)

# Введение

На сегодняшний день огромное количество информации хранится в цифровом виде и для хранения такой информации отлично подходят различные базы данных. Использование базы данных упрощает управление и хранение данных. Одним из примеров успешного внедрения баз данных является использование их различными приложениями.

Сегодня существует очень много площадок для аукциона, которые упрощают борьбу за лот. Не нужно ездить в другие города или даже страны для участия.

Цель моего курсового проекта является создание базы данные аукционной площадки.

Основными задачами курсовой работы являются:

* провести аналитический обзор литературы;
* спроектировать базу данных;
* реализовать функциональность базы данных;
* провести тестирование используемой технологии в базе данных;

В соответствии с заданием курсового проекта для проектирования базы данных аукционной площадки используется система управления базами данных Microsoft SQL Server.

# Постановка задачи

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере показать возможности базы данных. Для того чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы из той же области.

## **Обзор аналогов**

На сегодняшний день существует множество площадок, которые дают возможность участвовать в аукционах. Рассмотрим примеры web-приложений подобных сайтов.

e-auction.by

Одним из сайтов является e-auction.by. На этом сайте можно не только выбрать интересующий лот и принять участие в аукционе, но и выставить свой лот на аукцион.

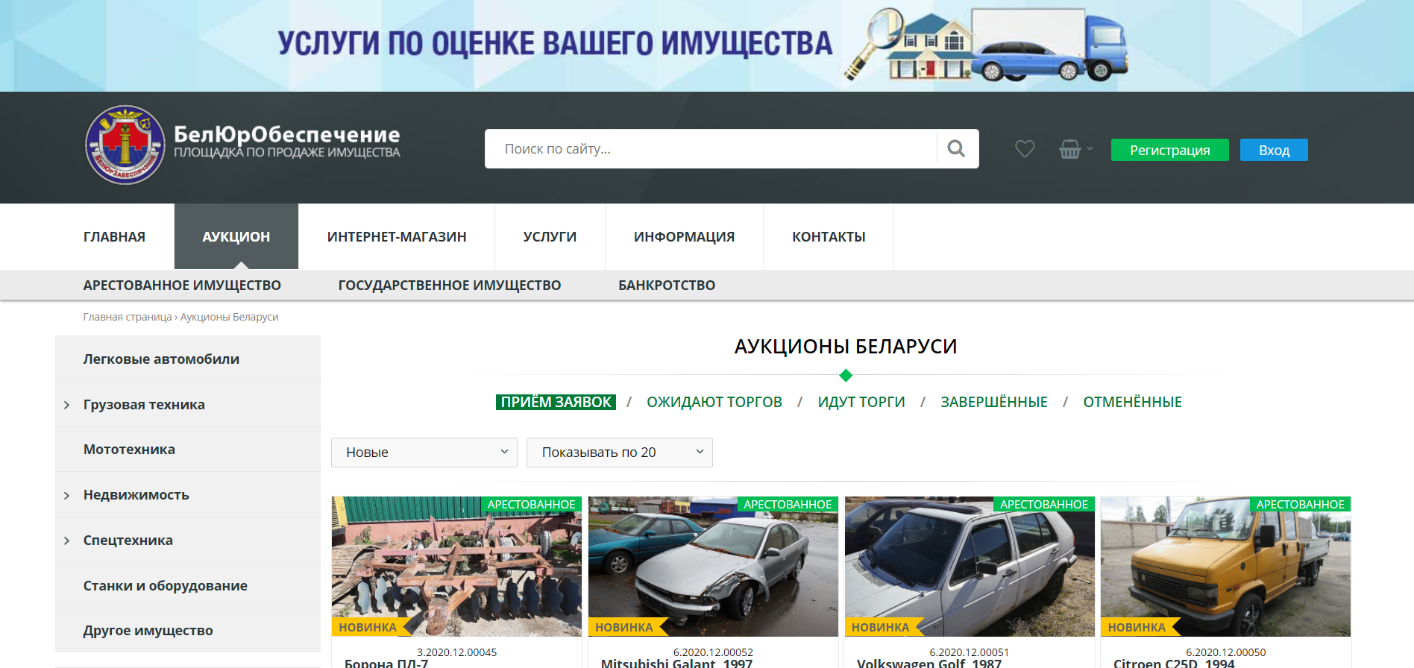


Рисунок 1.1 – Интерфейс сайта e-auction.by.

Auction.ru

Данный сайт имеет простой, но при этом очень удобный и понятный интерфейс. Сайт имеет множество категорий лотов. Особенностью сайта является наличие аукционов начиная от 1 рубля.

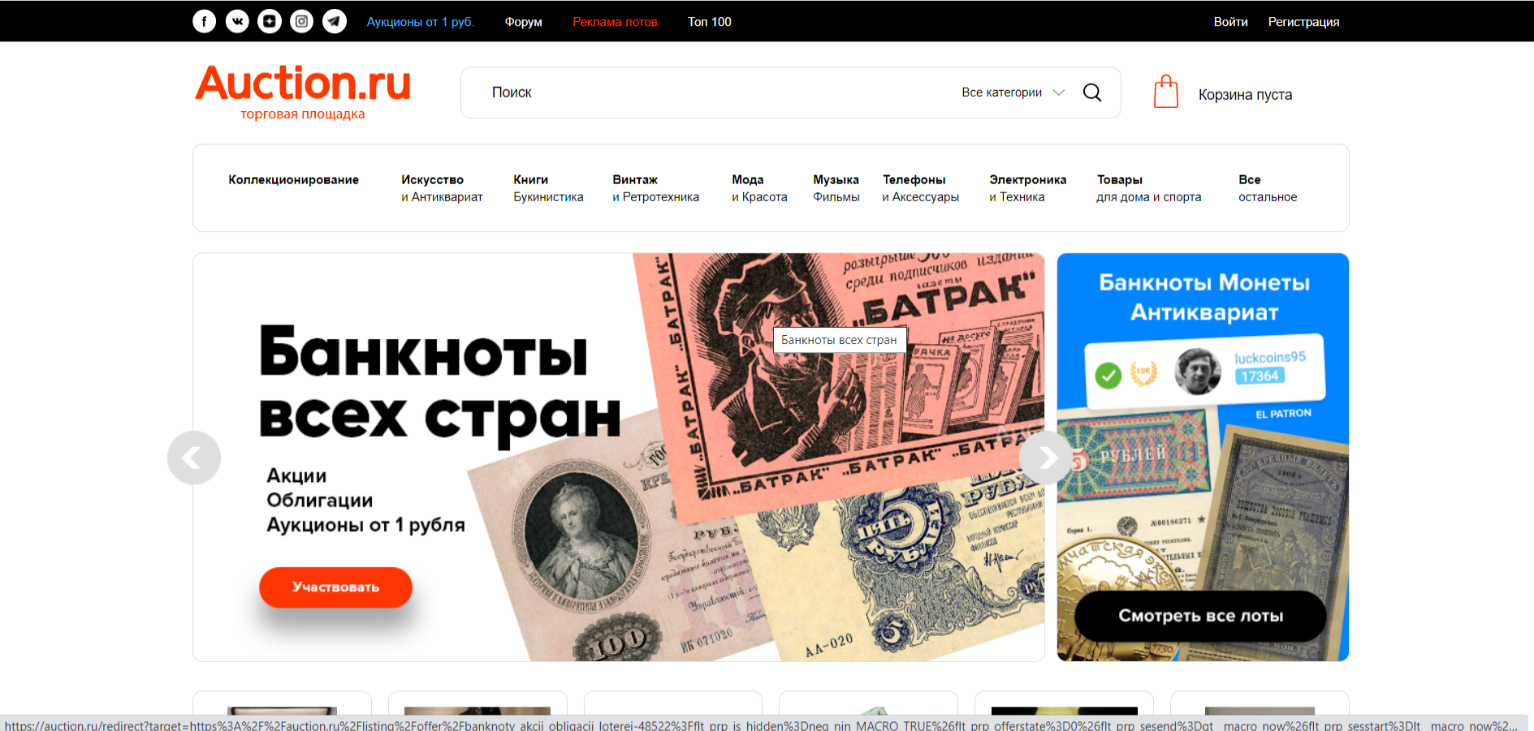


Рисунок 1.2 – Интерфейс сайта Auction.ru.

eBid.net

EBid.net это всемирная аукционная площадка. На сайте можно участвовать в аукционах из любой точки земного шара и это является огромным плюсом данного приложения. Здесь можно найти товары на любой вкус.

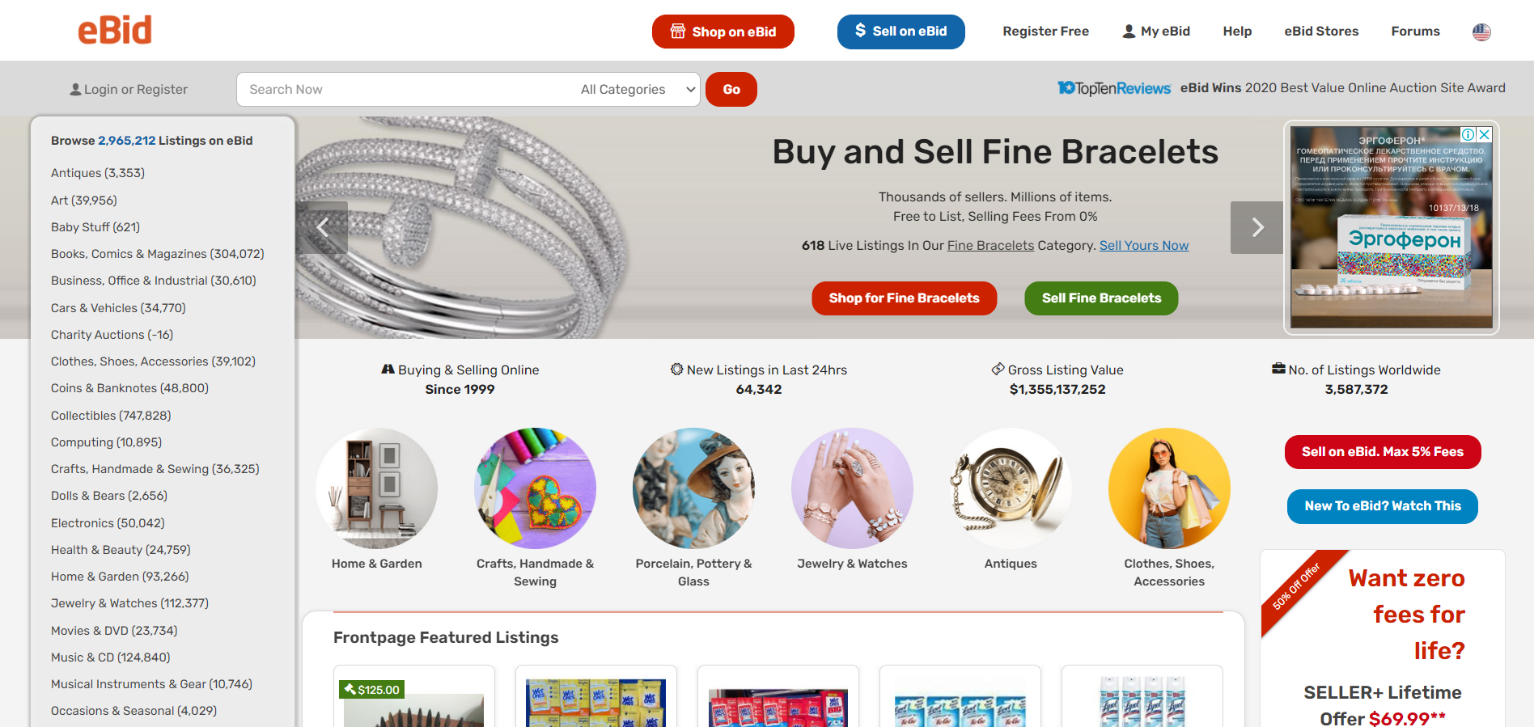


Рисунок 1.2 – Интерфейс сайта eBid.net.

* 1. **Анализ прототипов**

Проанализировав прототипы сайтов, можно проследить схожесть в их функциональности и интерфейсе.

Исходя из полученных результатов, можно сформулировать основные функциональные требования курсовой работы:

* Получение списка всех лотов.
* Получение списка лотов по категории.
* Прикрепление фото к лоту.
* Добавление нового лота.
* Возможность делать ставки.
* Полнотекстовой поиск.
* Просмотр ставок пользователя.
* Поиск лота по локации.

Данные требования будут реализованы в базе данных в виде хранимых процедур, которые описаны в главе 3 данной пояснительной записки.

* 1. **Описание используемых технологий**

СУБД существует огромное множество: Oracle, MS SQL Server, NoSQL, MySql и так далее. Для организации работы с базой данных в данной работе было решено использовать одну из наиболее популярных СУБД «Microsoft SQL Server», так как она предоставляет необходимые возможности оперирования объектами базы данных.

Также в процессе выбора технологии была рассмотрена технология Full-text search, которая позволяет сильно расширить возможности.

По сведениям из документации в полнотекстовый индекс включается один или несколько символьных столбцов таблицы. Эти столбцы могут содержать любой из следующих типов данных: char, varchar, nchar, nvarchar, text, ntext, image, xml или varbinary(max) и FILESTREAM. Каждый полнотекстовый индекс индексирует один или несколько столбцов таблицы.

# Проектирование модели базы данных

Для реализации поставленной задачи была создана база данных Auction. Диаграмма структуры полученной базы данных, разработанной в СУБД «Microsoft SQL Server», представлена в приложении А.

Для базы данных было разработано 6 таблиц, которые связаны друг с другом внешними ключами.

Таблица USERS хранит данные о пользователях.

Таблица BIDS содержит информацию о ставках пользователей.

Таблица CATEGORY хранит категории для более удобного поиска подходящего лота.

Таблица LOCATIONS хранит города, страны, области для более удобного поиска подходящего лота.

Таблица LOTS хранит информацию о лотах.

Таблица PHOTO хранит фотографии к лотам.

Создание перечисленных таблиц с установлением внешних связей описано в главе 3.

# Разработка необходимых объектов

## **Таблицы**

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2, а код их создания можно увидеть в Приложении Б. Ниже мы рассмотрим каждую таблицу подробнее.

Таблица USER состоит из семи столбцов:

* ID– хранит идентификатор пользователя;
* NAME – имя пользователя;
* LASTNAME – фамилия пользователя;
* EMAIL – электронная почта;
* PHONE – телефон пользователя;
* LOCATION\_ID – идентификатор локации пользователя, внешний ключ к таблице LOCATIONS;
* USER\_PASSWORD – пароль пользователя;

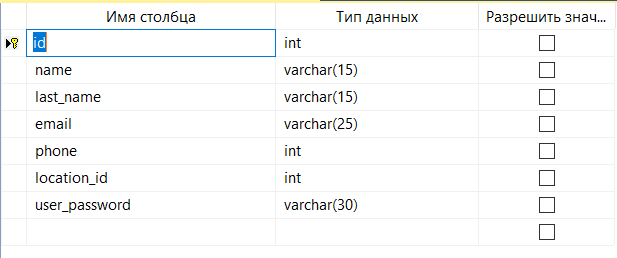


Рисунок 3.1 – Описание таблицы USER

Таблица BIDS состоит из шести столбцов:

* ID– хранит идентификатор ставки;
* USER\_ID – идентификатор пользователя, внешний ключ к таблице USERS;
* LOTS\_ID – идентификатор лота, внешний ключ к таблице LOTS;
* BID – размер ставки;
* CREATE\_DT – дата первой ставки;
* UPDATE\_DT – дата последней ставки;

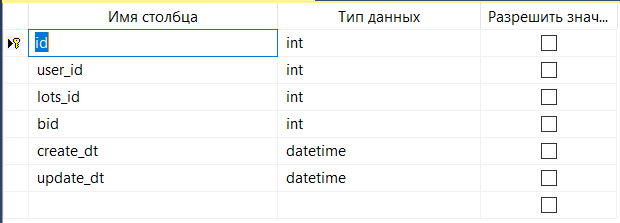


Рисунок 3.2 – Описание таблицы BIDS

Таблица CATEGORY состоит из двух столбцов:

* ID – хранит идентификатор категории;
* NAME – хранит название категории;

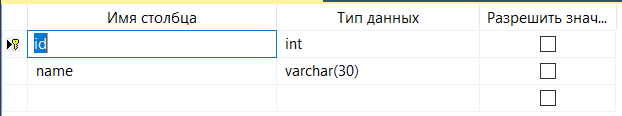


Рисунок 3.3 – Описание таблицы BIDS

Таблица LOCATIONS состоит из четырех столбцов:

* ID– хранит идентификатор локации;
* CITY – хранит название города;
* COUNTRY– хранит название страны;
* REGION – хранит название области;

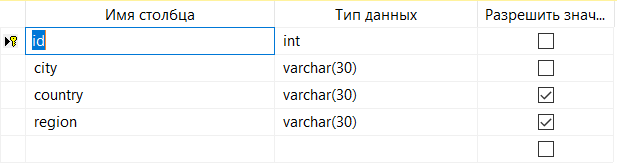


Рисунок 3.4 – Описание таблицы LOCATIONS

Таблица LOTS состоит из десяти столбцов:

* ID – хранит идентификатор лота;
* USER\_ID – хранит идентификатор пользователя, внешний ключ к таблице USERS;
* TITLE – хранит название лота;
* DESCRIPTION – хранит описание к лоту;
* START\_PRICE – хранит начальную цену лота;
* START\_TIME – хранит время начала аукциона;
* END\_TIME – хранит время конца аукциона;
* CATEGORY\_ID – хранит идентификатор категории, внешний ключ к таблице CATEGORY;
* NOW\_PRICE – хранит текущую ставку у лота;
* LOCATION\_ID – хранит идентификатор локации, внешний ключ к таблице LOCATIONS;

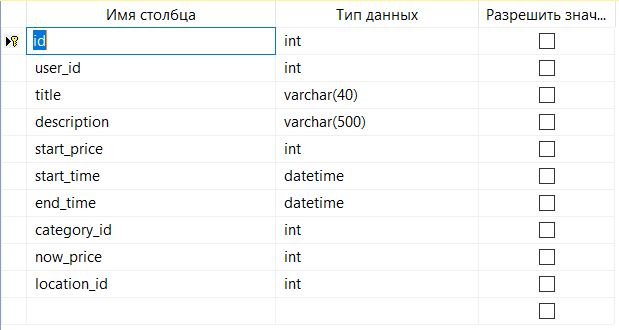


Рисунок 3.5 – Описание таблицы LOTS

Таблица PHOTO состоит из трех столбцов:

* ID – хранит идентификатор фото;
* LOT\_ID – хранит идентификатор лота, внешний ключ к таблице LOTS;
* PHOTO – хранит путь до фото;

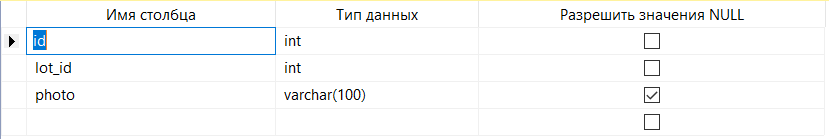


Рисунок 3.6 – Описание таблицы PHOTO

## **Процедуры**

Всего было разработано 27 процедур для работы с данными. Краткое описание процедур описано далее.

Пример создания хранимой процедуры Add\_lot, которая добавляет лоты представлена в приложении Б.

Данная процедура принимает входной параметр id\_lot (идентификатор лота), id\_user (идентификатор пользователя), title (название лота) и description (описание к лоту), start\_price (стартовая цена), start\_time (начало аукциона), end\_time (конец аукциона), category\_id (идентификатор категории), now\_price(текущая цена), location\_id (идентификатор локации).

Add\_location – процедура для добавления локации.

Add\_user – процедура для добавления пользователя;

Add\_category – процедура для добавления категории;

Delete\_location\_byCity – процедура для удаления локации по городу;

Delete\_category\_by\_name – процедура для удаления категории по названию;

Delete\_lot – процедура для удаления лота;

Delete\_photo – процедура для удаления фотографии;

ExportXML – процедура для экспорта данных в XML;

Generate\_rows – процедура для генерации 100000 строк;

ImportFromXML – процедура для импорта данных из XML ;

Insert\_photo\_from\_lot – процедура добавления фотографии к лоту;

Place\_a\_bet – процедура для того чтобы делать ставки;

Search\_lot\_from\_location – процедура поиска лота по локации;

Show\_all\_location – процедура выборки всех локаций;

Show\_all\_lots – процедура выборки всех лотов;

Show\_all\_photo – процедура выборки всех фото;

Show\_all\_users – процедура выборки всех пользователей;

Show\_allBets\_user – процедура выборки всех ставок заданного пользователя;

Show\_location\_byCity – процедура выборки локации по городу;

Show\_lots\_from\_category – процедура выборки лотов по категории;

Show\_photo\_by\_id – процедура выборки фото по id;

Update\_lot – процедура обновления информации о лоте;

Update\_photo – процедура обновления фотографии;

Update\_user – процедура обновления информации о пользователе;

Update\_category\_byId – процедура обновления информации о категории по Id;

Full\_text\_search – процедура для полнотекcтового поиска;

Примеры создания процедур представлены в приложении Б.

# Описание процедур импорта и экспорта данных

При всей своей отлаженности и очевидности, классический способ хранения и представления объектов развитой структуры имеет и вполне определенные недостатки и может вызывать проблемы, с которыми сталкивался любой разработчик, пытавшийся реализовать таким способом достаточно сложную систему. В некоторых ситуациях, решить эти проблемы позволяет хранение объекта в виде XML.

## **Процедура импорта данных из XML-файла**

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. С помощью процедуры ImportProcfromXml администратор может добавить новые локации в базу данных из XML-файла. Листинг данной процедуры можно увидеть в приложении Б

## **Процедура экспорта данных в формате XML**

Процедура ExportXML по выходному параметру возвращает данные о всех локациях в формате XML, которые потом отображаются в приложении. Её реализацию можно увидеть в приложении Б.

# Описание технологии

Полнотекстовый поиск в SQL Server позволяет пользователям и приложениям выполнять полнотекстовые запросы к символьным данным в таблицах SQL Server .

В полнотекстовый индекс включается один или несколько символьных столбцов таблицы. Каждый полнотекстовый индекс индексирует один или несколько столбцов таблицы, а каждому столбцу может соответствовать определенный язык.

Полнотекстовые запросы выполняют лингвистический поиск в текстовых данных в полнотекстовых индексах путем обработки слов и фраз в соответствии с правилами конкретного языка, например английского или русского. Полнотекстовые запросы могут включать простые слова и фразы или целые предложения. Полнотекстовый запрос возвращает все документы, которые содержат как минимум одно совпадение (известное также как попадание). Совпадение возникает в том случае, когда целевой документ содержит все термины, указанные в полнотекстовом запросе, и соответствует всем остальным условиям поиска, например расстояние между совпадающими терминами.

После добавления столбцов в полнотекстовый индекс, приложения и пользователи смогут выполнять полнотекстовые запросы к тексту из столбцов. Эти запросы могут вести поиск любых приведенных ниже элементов.

Одно или несколько конкретных слов или фраз (простое выражение);

Слова, начинающиеся заданным текстом, или фразы с такими словами (префиксные выражения);

Словоформы конкретного слова (производное выражение);

Слова или фразы, находящиеся рядом с другими словами или фразами (выражения с учетом расположения);

Синонимические формы конкретного слова (тезаурус);

Слова или фразы со взвешенными значениями (взвешенное выражение).

В полнотекстовых запросах не учитывается регистр букв. Например, поиск значения «Процессор» или «процессор» вернет одинаковые результаты.

В процедуре Full\_text\_search поиск в таблице LOTS осуществляется с помощью предиката CONTAINS. Листинг процедур приведён в приложении Б.

**Сравнение предиката LIKE и запросов полнотекстового поиска**

В отличие от полнотекстового поиска предикат LIKE Transact-SQL работает только с комбинациями символов. Кроме того, предикат LIKE нельзя использовать в запросах к форматированным двоичным данным. Более того, запрос с предикатом LIKE к большому количеству неструктурированных текстовых данных выполняется гораздо медленнее, чем эквивалентный полнотекстовый запрос к тем же данным. Выполнение запроса LIKE к нескольким миллионам строк текстовых данных может занять несколько минут, в то время как полнотекстовый запрос к тем же данным занимает всего несколько секунд или даже меньше, в зависимости от количества возвращаемых строк.

В полнотекстовом поиске используются следующие компоненты процесса SQL Server:

* **Пользовательские таблицы.** В этих таблицах содержатся данные, по которым осуществляется полнотекстовое индексирование.
* **Средство сбора полнотекстовых данных.** Средство сбора полнотекстовых данных работает с потоками полнотекстового сканирования. Оно отвечает за планирование заполнения полнотекстовых индексов и управление им, а также за наблюдение за полнотекстовыми каталогами.
* **Файлы тезауруса.** Эти файлы содержат синонимы искомых терминов.
* **Объекты списка стоп-слов.** Объекты списка стоп-слов содержат список часто встречающихся слов, бесполезных при поиске.
* **SQL Server обработчик запросов.** Обработчик запросов компилирует и выполняет SQL-запросы. Если SQL-запрос включает запрос полнотекстового поиска, то запрос направляется в средство полнотекстового поиска как в процессе компиляции, так и при выполнении. Результат запроса сопоставляется с полнотекстовым индексом.
* **Средство полнотекстового поиска.** Средство полнотекстового поиска в SQL Server полностью интегрировано с обработчиком запросов. Средство полнотекстового поиска компилирует и выполняет полнотекстовые запросы. Как часть выполнения запроса средство полнотекстового поиска может получать входные данные из тезауруса и списка стоп-слов.
* **Модуль записи индекса (индексатор).** Модуль записи индекса строит структуру, используемую для хранения индексированных токенов.
* **Диспетчер управляющей программы фильтрации.** Диспетчер управляющей программы фильтрации отвечает за наблюдение за состоянием узла управляющей программы фильтрации для полнотекстового поиска.

Обработка полнотекстового поиска:

Полнотекстовый поиск работает на базе средства полнотекстового поиска. Средство полнотекстового поиска имеет две роли: **поддержка индексирования и поддержка запросов**.

**Процесс полнотекстового индексирования:**

При инициации заполнения полнотекстового индекса (который также называют «сканированием») средство полнотекстового поиска помещает большие пакеты данных в память и оповещает управляющую программу полнотекстовой фильтрации. Управляющая программа выполняет фильтрацию и разбиение по словам, а также преобразует конвертированные данные в инвертированный список слов. Затем средство полнотекстового поиска запрашивает конвертированные данные из списка слов, удаляет стоп-слова и сохраняет списки слов в виде пакета в один или несколько инвертированных индексов.

При индексировании данных, хранящихся в столбце типа **varbinary(max)** или **image** , фильтр, реализующий интерфейс **IFilter** , извлекает текст в соответствии с заданным для этих данных форматом файлов, например Microsoft Word. В некоторых случаях для работы компонентов-фильтров необходимо, чтобы данные типов **varbinary(max)** или **image** были записаны в папку filterdata, а не принудительно отправлены в память.

Одним из этапов обработки собранных текстовых данных является их анализ средством разбиения по словам, которое разделяет текст на отдельные токены, или ключевые слова. Язык, используемый при разметке, задается на уровне столбца или может быть определен компонентом-фильтром по данным типа **varbinary(max)** , **image**или **xml** .

Для удаления стоп-слов и нормализации токенов перед их сохранением в полнотекстовом индексе или фрагменте индекса может быть проведена дополнительная обработка.

После завершения операции заполнения инициируется заключительный процесс слияния фрагментов индекса в один главный полнотекстовый индекс. Это повышает производительность запросов за счет использования одного главного индекса вместо нескольких фрагментов индексов и позволяет использовать более точные количественные оценки для ранжирования данных по релевантности.

**Обработка полнотекстовых запросов**

Обработчик запросов передает для обработки полнотекстовые части запроса средству полнотекстового поиска. Средство полнотекстового поиска выполняет разбиение по словам и при необходимости расширения тезауруса, морфологический поиск и обработку стоп-слов (пропускаемых слов). Затем полнотекстовые части запроса представляются в форме операторов SQL, в основном как потоковые функции, возвращающие табличные значения. Во время выполнения запроса эти потоковые функции для получения правильных результатов обращаются к инвертированному индексу. Результаты возвращаются клиенту в этой точке, либо перед возвращением клиенту они подвергаются дополнительной обработке.

**Архитектура полнотекстового индекса:**

Данные полнотекстовых индексов используются средством полнотекстового поиска для компиляции полнотекстовых запросов, способных быстро находить таблицу с теми или иными словами или словосочетаниями. В полнотекстовом индексе хранятся данные о значимых для поиска словах и их расположении в одном или нескольких столбцах таблицы базы данных. Полнотекстовый индекс — это специальный тип функционального индекса на основе токенов, создаваемый и используемый средством полнотекстового поиска для SQL Server. Процесс создания полнотекстового индекса отличается от создания индексов других типов. Вместо создания сбалансированного дерева на основе значения, хранящегося в конкретной строке, служба полнотекстового поиска создает инвертированную стековую сжатую структуру индекса на основе отдельных токенов индексируемого текста. Размер полнотекстового индекса ограничен только доступными ресурсами памяти компьютера, на котором запущен экземпляр SQL Server .

Начиная с SQL Server 2008полнотекстовые индексы встроены в ядро СУБД, а не размещены в файловой системе, как в предыдущих версиях SQL Server. В новой базе данных полнотекстовый каталог является виртуальным объектом, не принадлежащим ни к одной файловой группе. Он является лишь логическим понятием, связанным с группой полнотекстовых индексов. Однако обратите внимание, что при обновлении базы данных SQL Server 2005 (9.x) (любого полнотекстового каталога с файлами данных) создается новая файловая группа.

На одну таблицу может приходиться только один полнотекстовый индекс. Чтобы в таблице можно было создать полнотекстовый индекс, она должна содержать один уникальный столбец, значением которого не является NULL. Можно создать полнотекстовый индекс на столбцах типа **char**, **varchar**, **nchar**, **nvarchar**, **text**, **ntext**, **image**, **xml**и **varbinary**, а также для полнотекстового поиска может индексироваться **varbinary(max)** . Для создания полнотекстового индекса на столбце, тип данных которого **varbinary**, **varbinary(max)** , **image**или **xml** , необходимо указать столбец типов. **Столбец типов** — это столбец таблицы, в каждой строке которого хранятся расширения файлов (DOC, PDF, XLS и т. д.) документа.

# Тестирование

## **Тестирование производительности базы данных**

Для тестирования производительности была взята за основу таблица LOCATIONS, которую мы заполнили 100000 строками. После этого был применён SELECT-запрос к данной таблице на столбец ID и при помощи стандартных средств IDE MS SQL Server Management Studio оценена цена выборки к таблице. Результат данной оценки запроса приведён на рисунке 6.1.

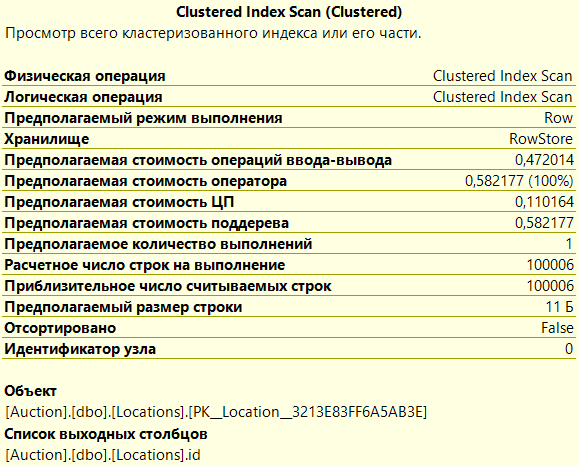


Рисунок 6.1 – Оценка запроса к таблице LOCATIONS с кластеризованным индексом

После проведения первоначальной оценки был построен некластеризованный индекс к таблице LOCATIONS, создание которого можно увидеть в приложении Б, и проведена оценка такого же SELECT-запроса к таблице LOCATIONS. Результаты, полученные во время оценки, представлены на рисунке 6.2.

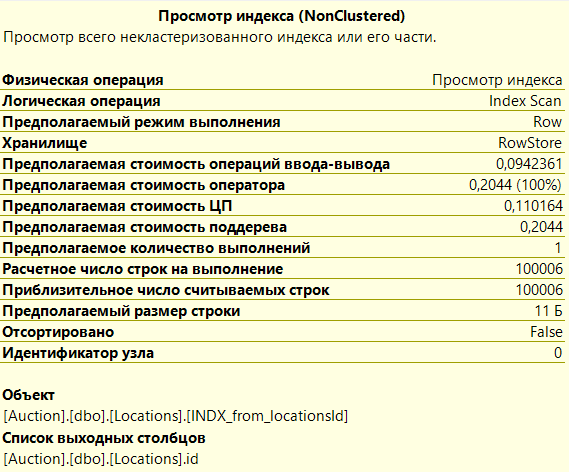


Рисунок 6.2 – Оценка запроса к таблице с построенным некластеризованным индексом

По результатам проведённых оценок, можно сделать вывод, что после создания некластеризованного индекса:

– стоимость по параметру “Предполагаемая стоимость операций ввода-вывода” стала в 5 раза меньше;

– стоимость по параметру “Предполагаемая стоимость оператора” снизилась почти в 3 раза;

– стоимость по параметру “Предполагаемая стоимость поддерева” уменьшилась почти в 3 раза.

Таким образом, постройка индекса к таблице была более чем оправдана, так как мы получили прирост производительности в 3 и более раза.

# Заключение

Перед началом разработки базы данных был произведен аналитический обзор прототипов приложений подобной тематики и определение функциональных возможностей моей базы данных.

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных «Аукционная плащадка».

В процессе выполнения курсовой работы была спроектирована база данных для аукционной площадки. База данных была разработана с помощью системы управления базами данных «Microsoft SQL Server 2012». Помимо этого, была изучена и применена при разработке приложения технология полнотекстового поиска.

При разработке базы данных было созданы следующие объекты:

– 6 таблиц с внешними связями;

– 27 хранимых процедур;

– 1 индекс;

При разработке выполнены следующие пункты:

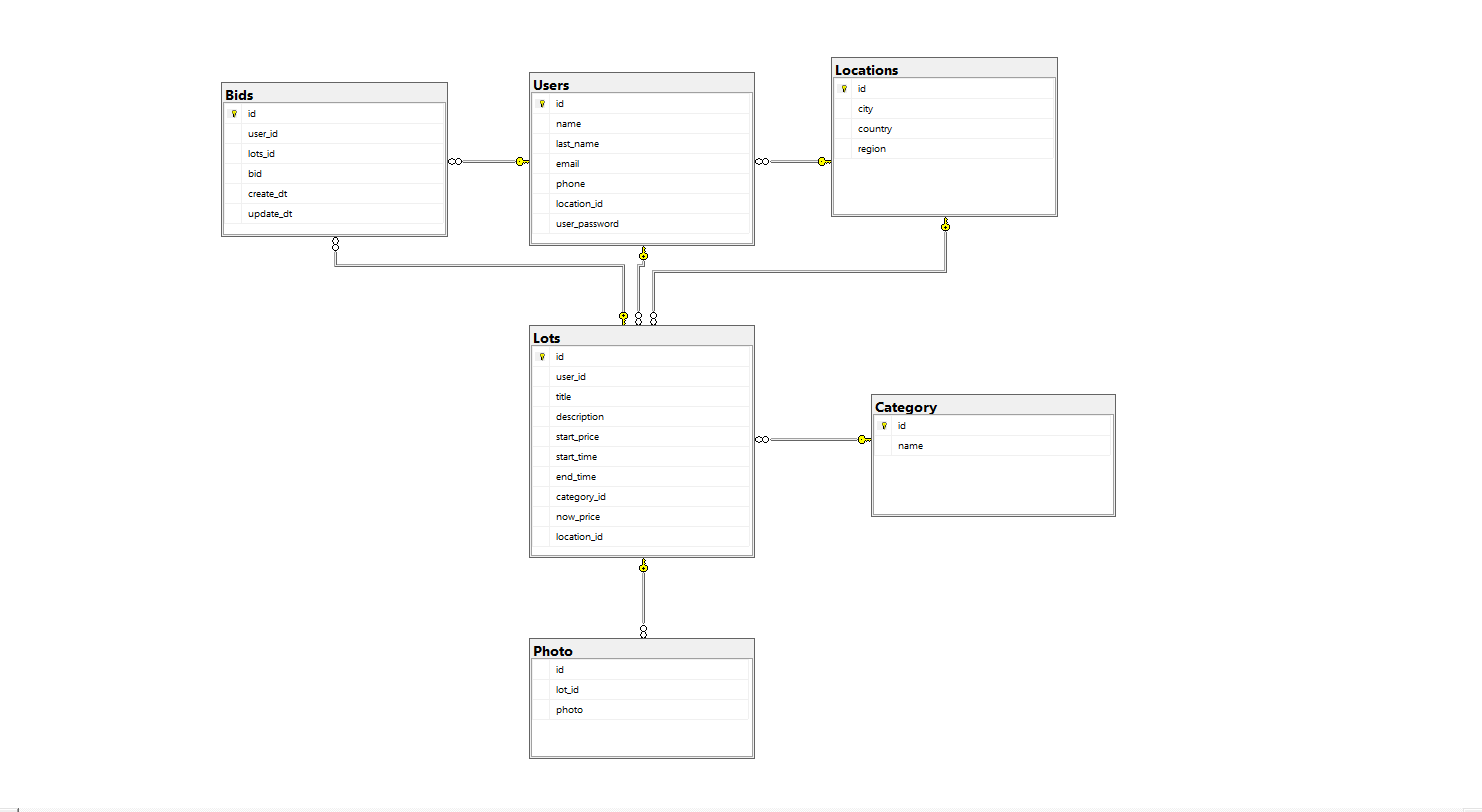
* Получение списка всех лотов.
* Получение списка лотов по категории.
* Прикрепление фото к лоту.
* Добавление нового лота.
* Возможность делать ставки.
* Полнотекстовой поиск.
* Просмотр ставок пользователя.
* Поиск лота по локации.
* Импорт и экспорт XML.

1. В соответствии с полученным результатом, можно сказать, что разработанная база данных функционирует верно, требования технического задания реализованы в полном объеме, поэтому цель курсового проекта можно считать достигнутой

**Список литературных источников**

1. Официальный сайт «Auction» [Электронный ресурс] – Режим доступа: <https://auction.ru/>
2. Официальный сайт «e-auction» [Электронный ресурс] – Режим доступа: <https://e-auction.by/>
3. Официальный сайт «eBid» [Электронный ресурс] – Режим доступа: <https://eBid.net/eu/>
4. Документация СУБД «Microsoft SQL Server» [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15>
5. Компонент Full-text search – SQL Server [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/sql/relational-databases/search/full-text-search?view=sql-server-ver15>

# Приложение A



# Приложение Б

Листинг создания таблиц

create table Locations(

id int identity(1,1) primary key,

city varchar(30) not null,

country varchar(30),

region varchar(30),

)

create table Category(

id int identity(1,1) primary key,

name varchar(30) not null

)

create table Users (

id int identity(1,1) primary key,

name varchar(15) not null,

last\_name varchar(15) not null,

email varchar(25) not null,

phone int not null,

location\_id int constraint User\_location foreign key references

Locations(id) not null

)

create table Lots(

id int identity(1,1) primary key,

user\_id int foreign key references Users(id) not null,

title varchar(40) not null,

description varchar(500) not null,

start\_price int not null,

start\_time DATETIME not null,

end\_time DATETIME not null,

category\_id int constraint Lots\_category foreign key references

Category(id) not null,

now\_price int not null,

location\_id int constraint Lots\_location foreign key references

Locations(id) not null

)

create table Bids(

id int identity(1,1) primary key,

user\_id int constraint Bids\_users foreign key references Users(id) not null,

lots\_id int constraint Bids\_lots foreign key references Lots(id) not null,

bid int not null,

create\_dt DATETIME not null,

update\_dt DATETIME not null

)

create table Photo(

id int identity(1,1),

lot\_id int constraint Photo\_lots foreign key references

Lots(id) not null,

photo varchar(100)

)

Листинг процедуры Add\_lot

go

create procedure Add\_lot

@id\_user int,

@title varchar(40),

@description varchar(500),

@start\_price int,

@start\_time DATETIME,

@end\_time DATETIME,

@category\_id int,

@location\_id int

as

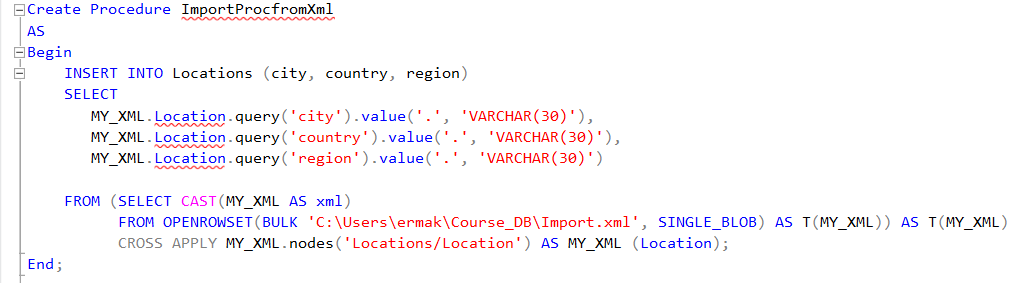
begin

insert into Lots values(@id\_user,@title,@description,@start\_price,

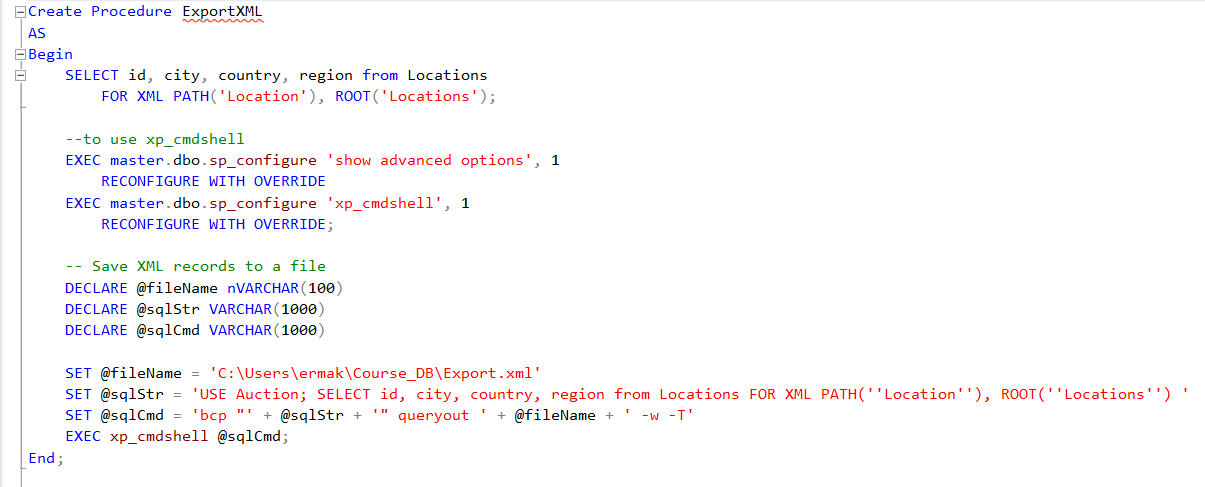
@start\_time,@end\_time,@category\_id,@start\_price,@location\_id);

End

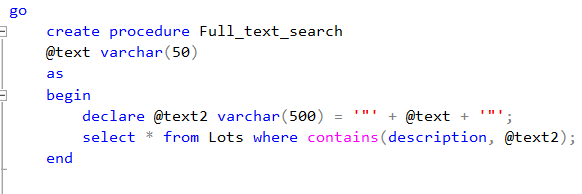
Процедура импорта данных из XML-файла

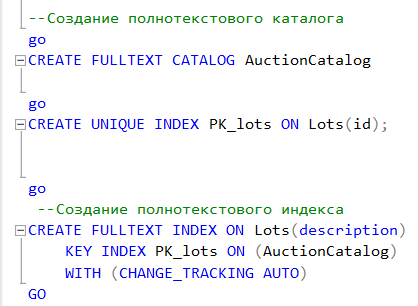


Процедура экспорта данных в формат XML

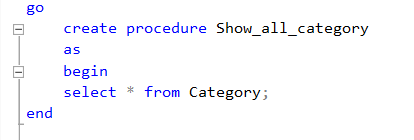


Процедура Full\_text\_search, процедура применяющая технологию полнотекстого поиска:

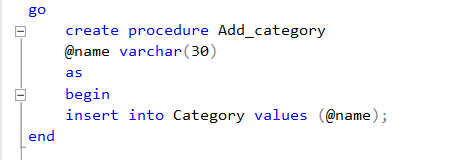
 Реализация полнотекстового поиска:



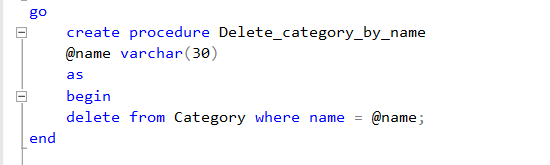
Процедура просмотра всех категорий:



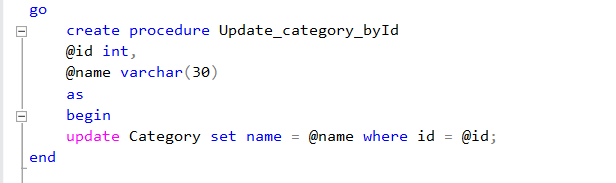
Процедура добавления категории:



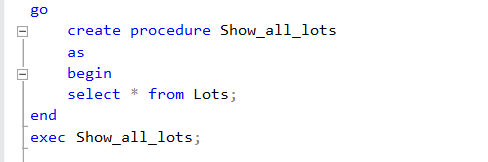
Процедура удаления категории по названию:



Изменение категории по Id:



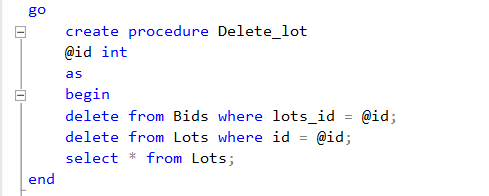
Процедура просмотра всех лотов:



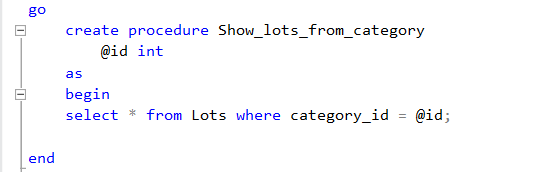
Процедура обновления лота:



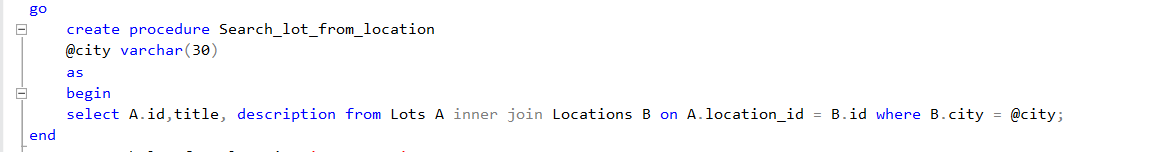
Процедура удаления лота:



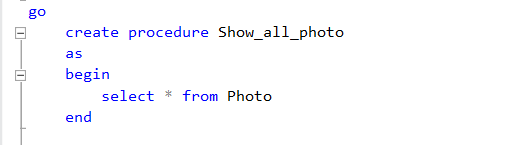
Процедура просмотра лотов по категории:



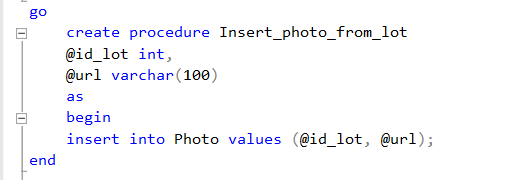
Процедура поиска лота по локации:



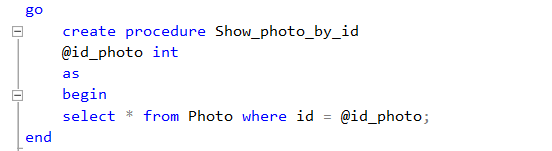
Процедура просмотра всех фотографий:



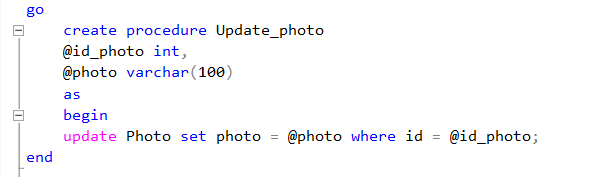
Процедура добавления фотографии к лоту:



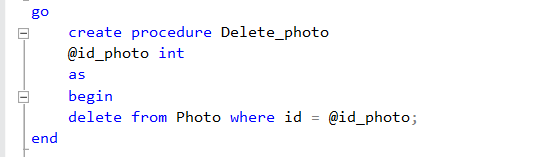
Процедура просмотра фото по Id:



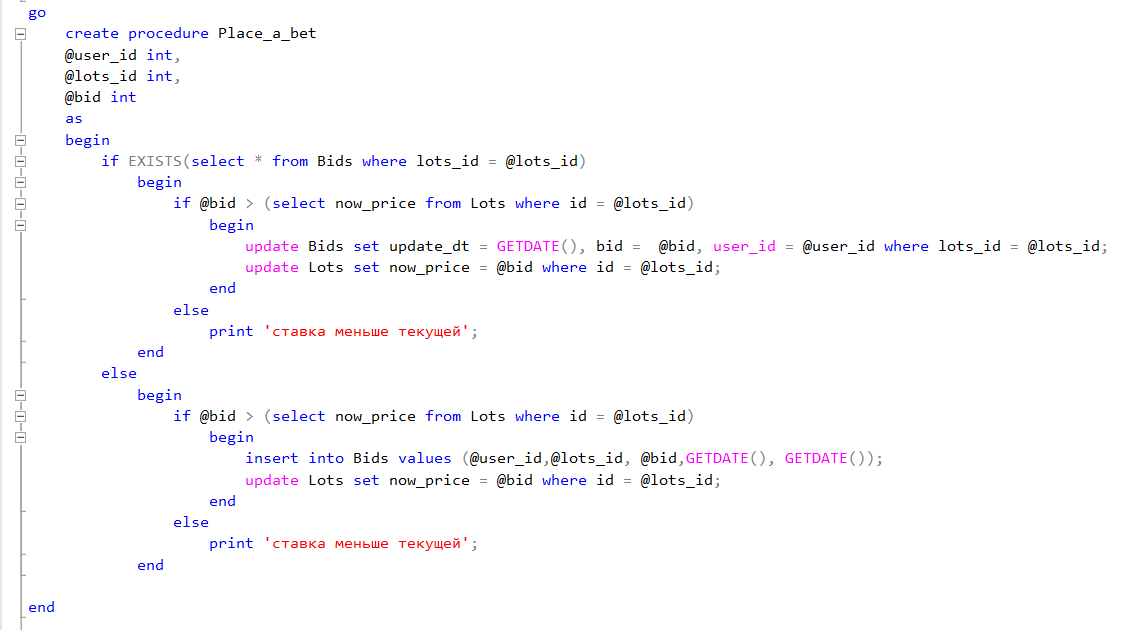
Процедура изменения фото:



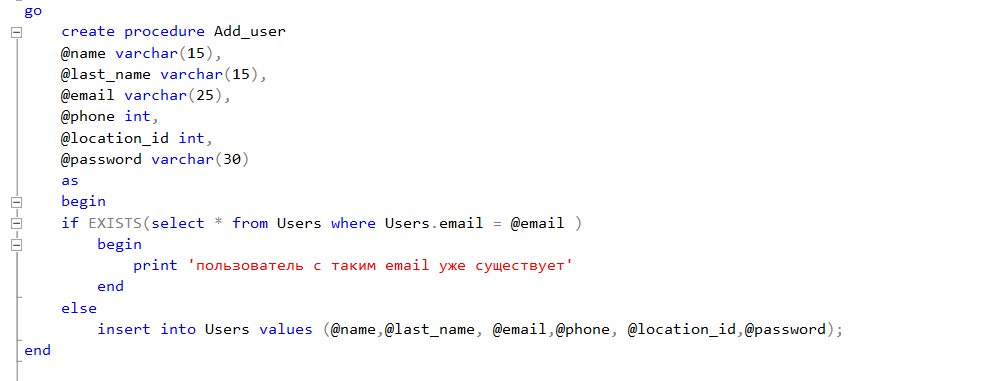
Процедура удаления фото:



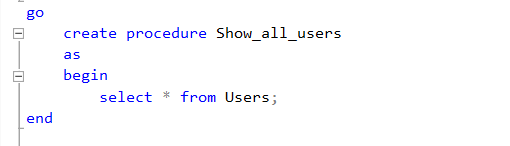
Процедура для делания ставки:



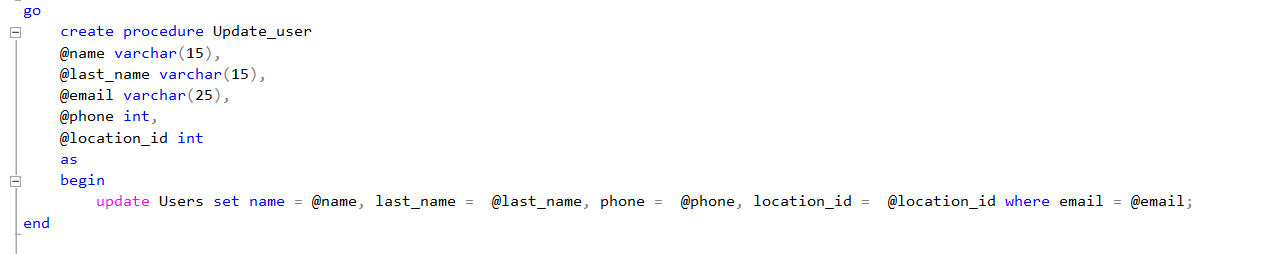
Процедура добавления пользователя:



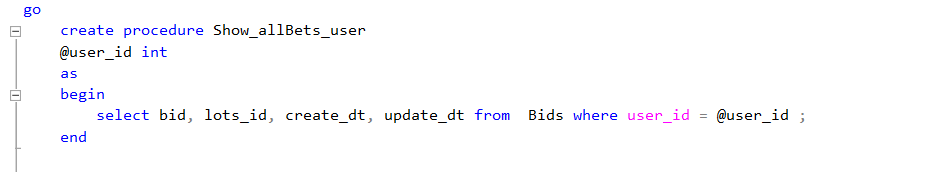
Процедура просмотра всех пользователей:



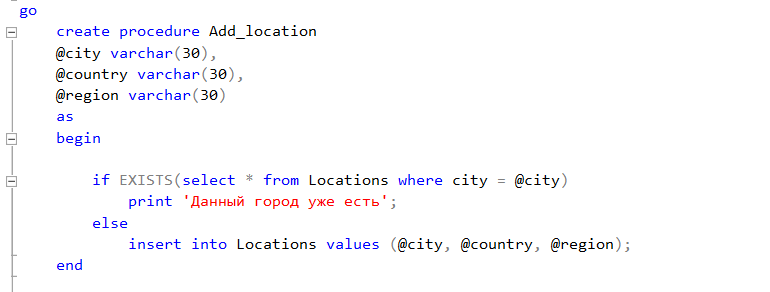
Процедура изменения пользователя:



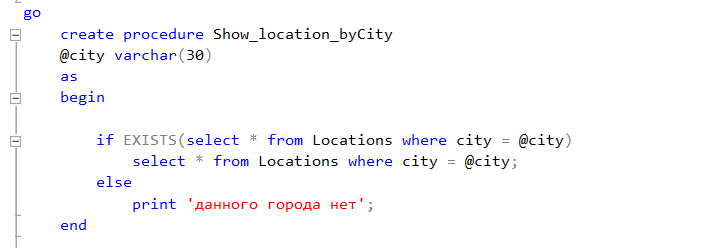
Процедура просмотра ставок конкретного пользователя:



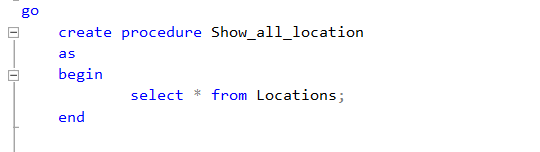
Процедура добавления локации:



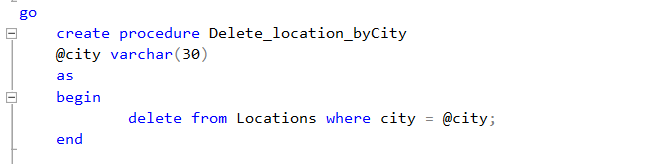
Процедура просмотра локации по городу:



Процедура просмотра всех локаций:



Процедура удаления локации:



Процедура генерации 100000 строк:

