Вопросы к экзамену

ПББДМС, 3курс ПОИБМС

1. Модели мобильных приложений, работающих с персистентными данными.

**Управление данными**: набор компьютерных технологий, связанных с надежным хранением, извлечением и обработкой данных.

**Персистентные данные** – это данные, которые можно сохранять.

1. **Модель: приложение – локальная файловая система**



Используются такие форматы данных, как:

* **плоские файлы**. Плоские файлы – самая простая разновидность структурированных данных. Плоские файлы называются так потому, что имеют минимальную структуру. Каждое поле имеет фиксированную длину (например, длина поля имени всегда равна 15 символам), и в этой структуре поля не отделены друг от друга. Тот, кто создал базу данных, для каждого из полей назначил позицию и длину. Любая программа, которая использует этот файл, должна "знать", какие характеристики назначены каждому полю, потому что этой информации в самой базе данных нет.
* **XML-файлы**. Файл XML является расширяемым видом языка разметки Markup Language. Такие файлы формата, представляют из себя документы, использующие теги в целях определения объектов, а также их атрибутов.
* **JSON-файлы**. JSON— текстовый формат обмена данными, основанный на JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования.
* **сериализация** и **десериализация** объектов в файлы. Сериализация — процесс перевода какой-либо структуры данных в последовательность битов. Обратной к операции сериализации является операция десериализации (структуризации) — восстановление начального состояния структуры данных из битовой последовательности.

1. **Модель: приложение – локальная база данных**

**База данных** (БД) - это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

**Система управления базами данных** (СУБД) - это программа, позволяющая создавать базы данных, а также обеспечивающая обработку (сортировку) и поиск данных.

**Сервер базы данных** - сервер БД обслуживает базу данных и отвечает за целостность и сохранность данных, а также обеспечивает операции ввода вывода при доступе клиента к информации.

**Реляционная база данных** - это базы, где вся информация хранится в таблицах, связанных друг с другом специальными отношениями.



* + SQL-базы данных
  + NoSQL-базы данных. NoSQL убирает все ограничения реляционной модели (недостаточная производительность, трудоёмкое горизонтальное масштабирование, недостаточная производительность в кластере) и облегчает средства хранения и доступа к данным. Такие БД используют неструктурированный подход (создание структуры на лету), тем самым снимая ограничения жестких связей и предлагая различные типы доступа к специфическим данным. NoSQL базы данных не работают с реляционными моделями. Существует много различных решений, каждое из которых работает немного по-своему и служит специфической цели. Эти безсхемные решения снимают ограничения с формирования сущностей и допускают хранения данных в виде ключ-значение.
* **Структуры данных и их типы** - реляционные БД используют строгие схемы данных, NoSQL БД допускают любой тип данных
* **Запросы** - вне зависимости от типа лицензии, реляционные базы данных в той или иной мере соответствуют стандартам SQL, поэтому данные из них можно получать при помощи языка SQL. NoSQL БД используют специфические способы запросов к данным.
* **Масштабируемость** - оба эти типа СУБД довольно легко поддаются вертикальному масштабированию (т.е. увеличение системных ресурсов). Тем не менее, так как NoSQL это более современный продукт, именно такие СУБД предлагают более простые способы горизонтального масштабирования (т.е. создание кластера из нескольких машин).
* **Надежность** - когда дело доходит до сохранности данных и гарантии выполнения транзакций SQL БД по-прежнему занимают лидирующие позиции.
* **Поддержка** - Реляционные СУБД имеют не малую историю за плечами. Они очень популярны и предлагают, как платные, так и бесплатные решения. При возникновении проблем, все же гораздо проще найти ответ, если дело касается реляционных систем, чем NoSQL, особенно если решение довольно сложное по своей природе (например, MongoDB).

Хранение и доступ к сложным структурам данных - изначально реляционные системы предполагали работу со сложными структурами, именно поэтому они превосходят остальные решения по производительности.

1. **Модель: приложение – удаленный сервер базы данных**



* + SQL-базы данных
  + NoSQL-базы данных

1. **Модель: приложение – удаленный SOAP-сервис – удаленный сервер базы данных**



**SOAP** (от англ. Simple Object Access Protocol — простой протокол доступа к объектам; вплоть до спецификации 1.2) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур. Использование SOAP для передачи сообщений увеличивает их объём и снижает скорость обработки.

1. **Модель: приложение – удаленный REST-сервис – удаленный сервер базы данных**



**REST** (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Технически REST-сервисы можно реализовать при помощи статических HTML-страниц.

1. **Модель: приложение – удаленный TCP/UDP-сервис – удаленный сервер базы данных**



1. Модель памяти в ОС Android.

ROM – постоянная память, ПЗУ

* Read Only Memory
* Энергонезависимая
* Предназначен только для чтения
* Записывается однократно при изготовлении
* Включает несколько разделов
* Содержит образ ОС
* Подключение с правами ROOT (суперпользователь) позволяет заменить образ ОС

RAM – оперативная память, ОЗУ

* Random Access Memory
* Энергозависимая память
* Оперативная память для выполнения приложений

Internal – внутренняя память

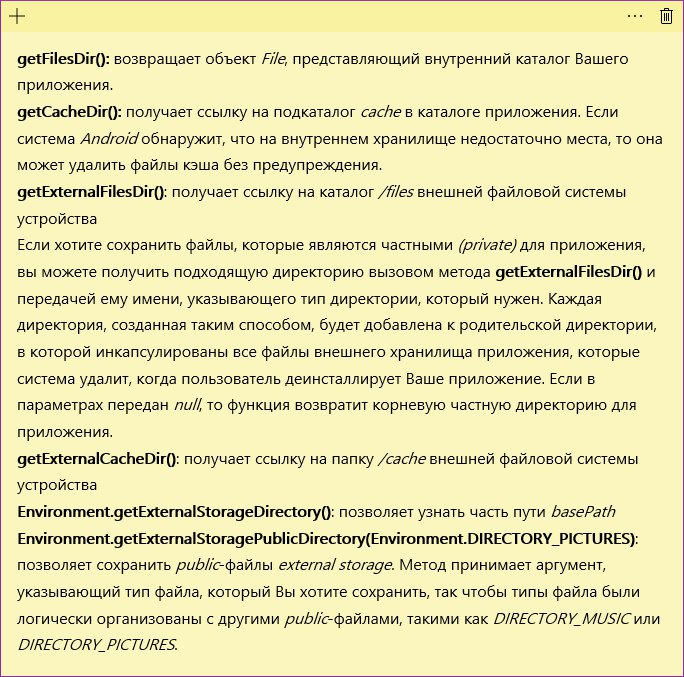
* энергонезависимая память
* для хранения пользовательских файлов:
  + apk-файлов (приложений)
  + Данных приложений (в т.ч. БД)
  + Медиа-файлов, документов и пр.
* Приложение всегда имеет разрешение на чтение и запись файлов в свой внутренний каталог на internal storage
* Существует специальный внутренний каталог временных файлов кэша приложения. Как только файл больше не нужен, его следует удалить. Если Android обнаружит, что на внутреннем хранилище недостаточно места, то файлы кэша могут быть удалены без предупреждения.
* Каталог internal storage приложения указывается на основе имени пакета приложения
* Другое приложение может прочитать внутренние файлы, если установить файловый режим с разрешенным чтением
* Файлы не будут доступны для других приложений пока используется MODE\_PRIVATE

External – внешняя память

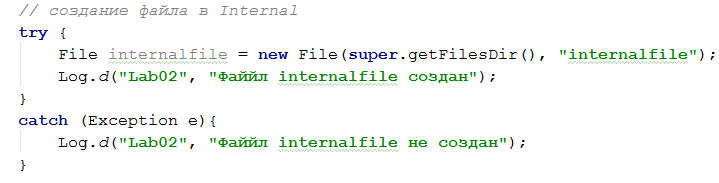
* Энергонезависимая память
* Может располагаться:
  + в собственной памяти устройства (эмуляция)
  + на внешнем носителе (SD-карте)
* память SD-карты не всегда доступна:
  + монтируется (подключается)
  + размонтируется (отключается)
* для доступа к External-памяти требуется разрешение, устанавливаемое в файле манифеста приложения
* при деинсталляции приложения автоматически удалятся файлы только из специальной директории для частных внешних файлов
* Перед доступом нужно проверить, что есть в наличии
* External Storage = MEDIA\_MOUNTED, то можно читать и записывать файлы
* **Файлом** (от англ. file — цепочка) называется последовательный набор данных, хранящийся на каком-либо физическом носителе и имеющий собственные имя и расширение. Расширение файла предназначено для однозначной и исчерпывающей идентификации типа файлового объекта; оно записывается справа от имени файла и отделяется от него точкой. В данном случае тип файлового объекта — это функциональная характеристика файла, с помощью которой операционная система определяет набор программ, способных обрабатывать или использовать данный файл.

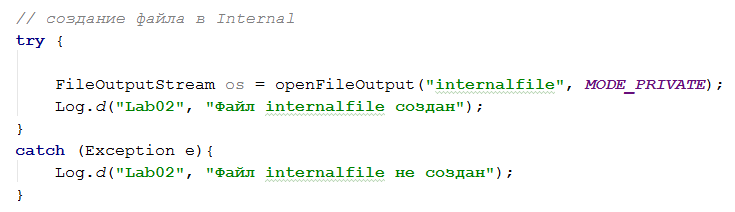
1. Работа с локальной файловой системой. Внутренняя и внешняя память в ОС Android.

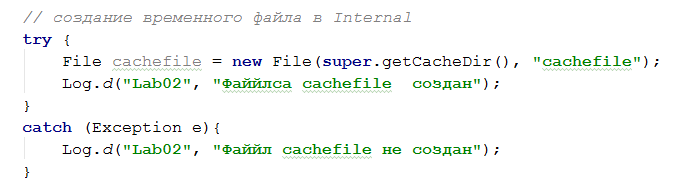


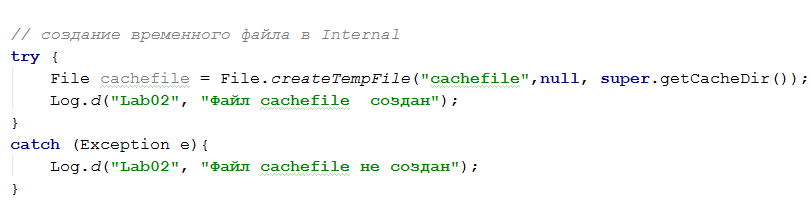


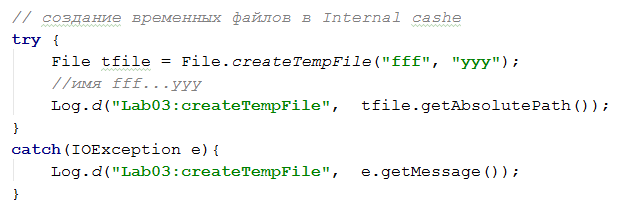
1. Работа файловой системой, расположенной во внутренней памяти ОС Android.











1. Работа файловой системой, расположенной во внешней памяти ОС Android.

Публичные файлы – Public files

* Публичные файлы (Public files) - это файлы, которые должны быть свободно доступны для других приложений и пользователя.
* Когда пользователь деинсталлирует Ваше приложение, эти файлы должны остаться доступными для пользователя.
* Пример – фотографии
* Можно указать тип файлов, который будет храниться DIRECTORY\_MUSIC или DIRECTORY\_PICTURES

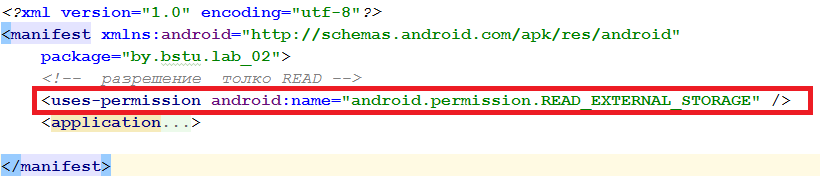
Частные файлы – Private files

* Частные файлы (Private files) - это файлы, полные права на которые принадлежат приложению, и которые должны быть удалены при деинсталляции приложения пользователем.
* Когда пользователь деинсталлирует Ваше приложение, система удалит все файлы в Вашем частном каталоге на внешнем хранилище.

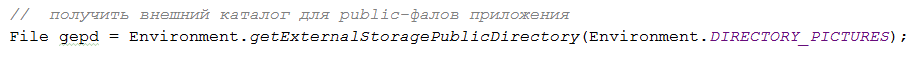
Пример – временные файлы

* Установка разрешений READ/WRITE:

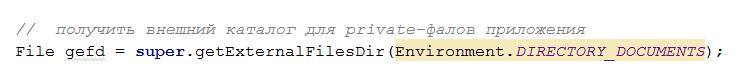




**External-память**: Public files – доступны для других приложений, при деинсталляции приложения файлы сохраняются.



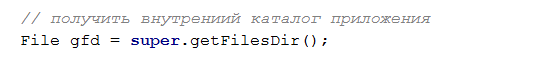
**External-память**: Private files – доступны для других приложений, при деинсталляции приложения файлы автоматически удаляются. Если указать параметр null вернет директорий External-корня для приложения.

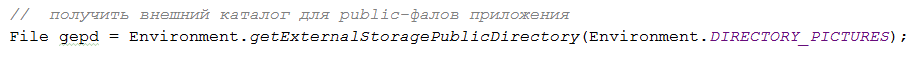


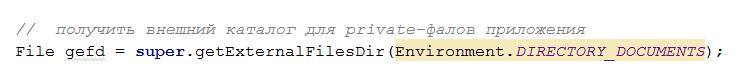
1. Работа с каталогами файловой системы ОС Android: создание, удаление каталогов, чтение свойств каталогов (имя, полный путь, дата последнего изменения, доступность для чтения записи, содержимое каталога, родительский каталог, …).

* android.content.Context
* fileList() – получает все файлы, которые содержатся в подкаталоге */files* в каталоге приложения
* getCacheDir() – получает ссылку на подкаталог *cache* в каталоге приложения
* getDir(String dirName, int mode) – получает ссылку на подкаталог в каталоге приложения, если такого подкаталога нет, то он создается
* getExternalCacheDir() – получает ссылку на папку */cache* внешней файловой системы устройства
* getExternalFilesDir() – получает ссылку на каталог */files* внешней файловой системы устройства
* getFileStreamPath(String filename) – возвращает абсолютный путь к файлу в файловой системе
* openFileInput(String filename) – открывает файл для чтения
* openFileOutput (String name, int mode) – открывает файл для записи
* deleteFile(String name) – удаляет файл

Получить внутренний каталог приложения, применяемый для хранения файлов приложения:





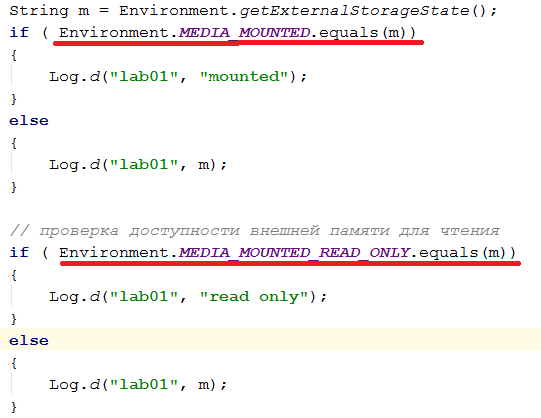


Получитьвнутренний каталог для временных кэш-файлов:

* Если недостаточно internal-памяти, то кэш-файлы могут быть удалены без предупреждения



* Проверить состояние внешней памяти:



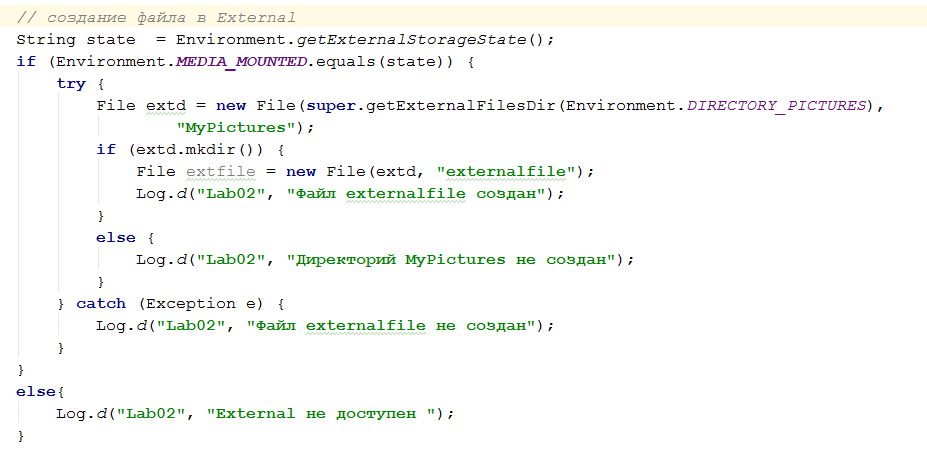
Класс File имеет ряд методов, которые позволяют управлять файлами и каталогами. Рассмотрим некоторые из них:

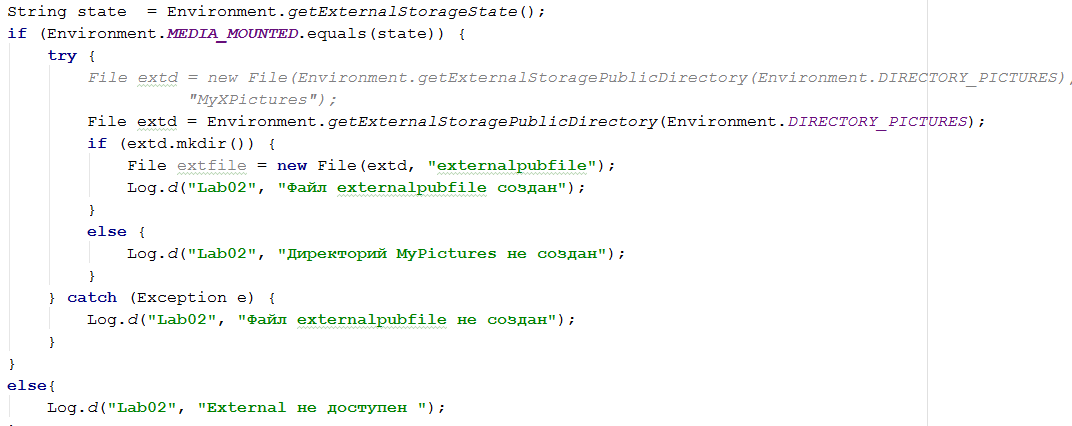
* **boolean createNewFile():** создает новый файл по пути, который передан в конструктор. В случае удачного создания возвращает true, иначе false
* **boolean delete():** удаляет каталог или файл по пути, который передан в конструктор. При удачном удалении возвращает true.
* **boolean exists():** проверяет, существует ли по указанному в конструкторе пути файл или каталог. И если файл или каталог существует, то возвращает true, иначе возвращает false
* **String getAbsolutePath():** возвращает абсолютный путь для пути, переданного в конструктор объекта
* **String getName():** возвращает краткое имя файла или каталога
* **String getParent():** возвращает имя родительского каталога
* **boolean isDirectory():** возвращает значение true, если по указанному пути располагается каталог
* **boolean isFile():** возвращает значение true, если по указанному пути находится файл
* **boolean isHidden():** возвращает значение true, если каталог или файл являются скрытыми
* **long length():** возвращает размер файла в байтах
* **long lastModified():** возвращает время последнего изменения файла или каталога. Значение представляет количество миллисекунд, прошедших с начала эпохи Unix
* **String[] list():** возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге
* **File[] listFiles():** возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге
* **boolean mkdir():** создает новый каталог и при удачном создании возвращает значение true
* **boolean renameTo(File dest):** переименовывает файл или каталог

1. Работа файлами ОС Android: создание, удаление файлов , чтение свойств файлов (имя, полный путь, длина, дата последнего изменения, доступность для чтения записи…).

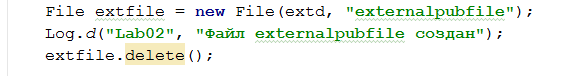
Класс File может использоваться для создания файлов или каталогов. Также можно узнать свойства файлов (размер, дату последнего изменения, режим чтения/записи), определить к какому типу (файл или каталог) относится объект File, удалить файл. У класса очень много методов, перечислим некоторые.

* **getAbsolutePath()** - абсолютный путь файла, начиная с корня системы. В Android корневым элементом является символ слеша (/)
* **canRead()** – доступно для чтения
* **canWrite()** – доступно для записи
* **exists()** – файл существует или нет
* **getName()** – возвращает имя файла
* **getParent()** – возвращает имя родительского каталога
* **getPath()** – путь
* **lastModified()** – дата последнего изменения
* **isFile()** – объект является файлом, а не каталогом
* **isDirectory()** – объект является каталогом
* **isAbsolute()** – возвращает true, если файл имеет абсолютный путь
* **renameTo(File newPath)** – переименовывает файл. В параметре указывается имя нового имени файла. Если переименование прошло неудачно, то возвращается false
* **delete()** – удаляет файл. Также можно удалить пустой каталог

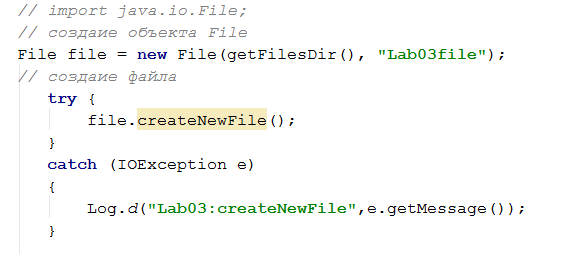


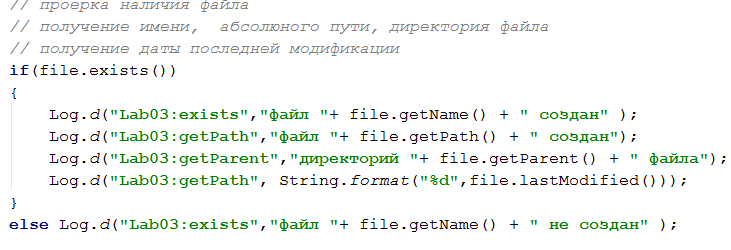


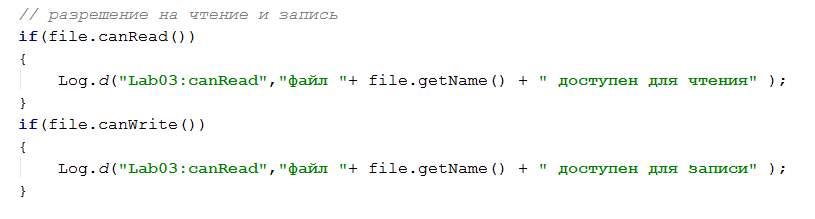
* Delete File:



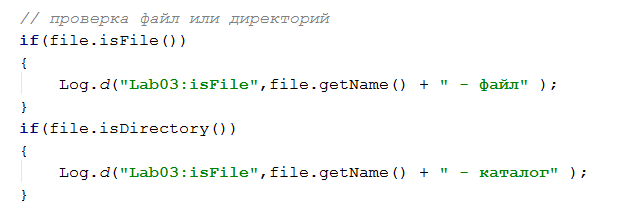
* Создание и свойства:

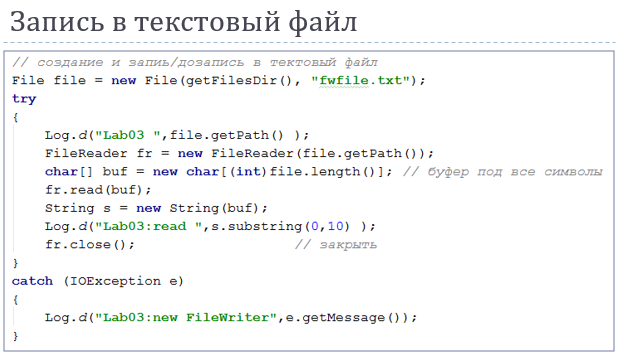


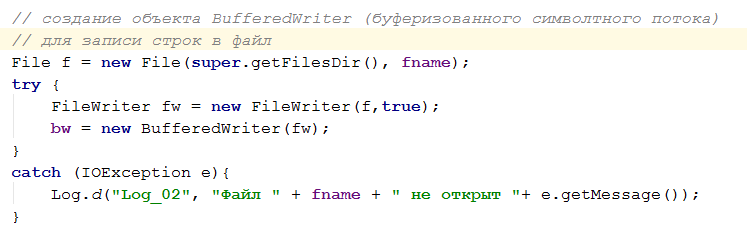


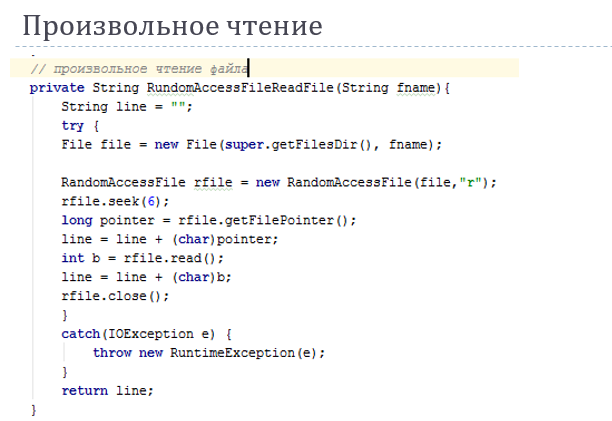


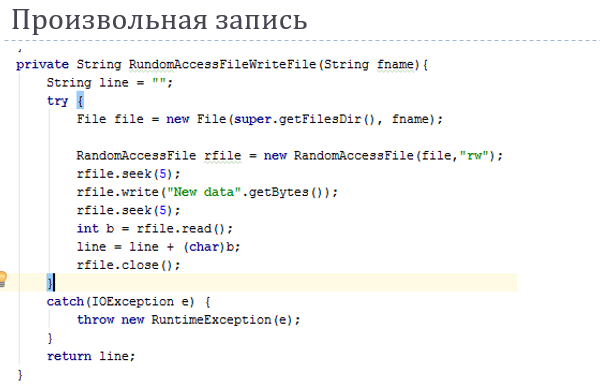
* Файл или каталог?











1. Концепция потоковых данных. Работа с потоковыми данными в ОС Android.

Поток (Stream) – это абстракция источника или приемника данных

Потоки бывают байтовые и символьные

java.io - предназначен для чтения и записи данных в ресурс

* **Базовые**
  + InputStream / OutputStream
  + Reader / Writer
  + InputStreamReader / OutputStreamWriter
* **Буферизация**
  + BufferedInputStream / BufferedOutputStream
  + BufferedReader / BufferedWriter
* **Файлы**
  + FileInputStream / FileOutputStream
  + RandomAccessFile
  + FileReader / FileWriter
* **Исключения ввода-вывода**: базовый IOException

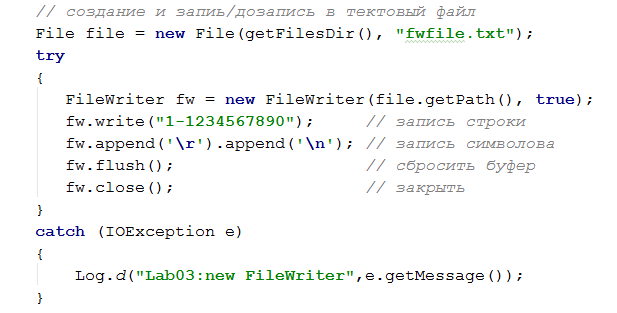
java.io.DataOutputStream – для записи примитивных типов (int, double, float, boolean, UTF-строки и пр.)

java.io.DataInputStream – для чтения примитивных типов

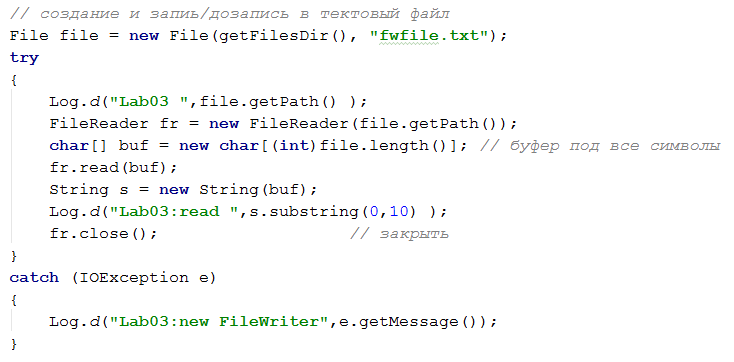
**Поток** (Stream) – это абстракция источника или приемника данных. Поток необходим для того, чтобы программист не задумывался о физическом размещении и формате данных. Потоки бывают байтовые и символьные. Базовые классы для потоков InputStream и Reader, read(), OutputStream и Writer, write().

Исключения ввода вывода: базовый **IOException**.

**Класс java.io.FileWriter:** для записи текстовых данных в файл, производный от абстрактного класса Writer, несколько конструкторов, метод write имеет несколько вариантов (строка, массив символов, int, …)



**Класс java.io.FileReader:** для чтения текстовых данных из файла, производный от абстрактного класса Reader, несколько конструкторов, метод write имеет несколько вариантов (строка, массив символов, int, …)



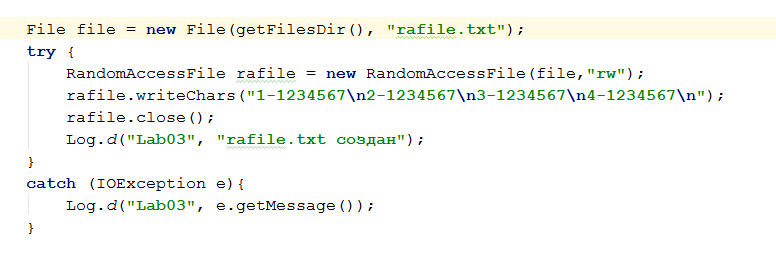
1. Концепция буферизированного ввода-вывода. Буферизированный ввод-вывод данных в ОС Android.

* Операции ввода/вывода по сравнению с операциями в оперативной памяти выполняются медленно
* В оперативной памяти выделяется промежуточная область - буфер, в которой постепенно накапливается информация
* Когда буфер заполнен, его содержимое однократно переносится на диск и буфер очищается
* java.io.BufferedWriter – для буферизированной записи, ускоряет вывод
* java.io.BufferedReader: для буферизированного чтения, ускоряет чтение

1. Концепция произвольного доступа к данным файла. Произвольный (Random) ввод-вывод данных в ОС Android.

Обычно ввод-вывод выполняется последовательно: каждая операция чтения или записи начинается там, где закончилась предыдущая. Но при необходимости файл может быть прочитан и записан в произвольном порядке.

**Класс java.io.RandomAccessFile:** напоминает использование совмещенных в одном классе потоков DataInputStream и DataOutputStream (они реализуют те же интерфейсы DataInput и DataOutput). Метод seek() позволяет переместиться к определенной позиции и изменить хранящееся там значение.







1. Сериализация и десериализация данных. Формат XML. Работа с XML в ОС Android.

Сериализация - это процесс сохранения состояния объекта в последовательность бит

Десериализация - это процесс восстановления объекта из последовательности бит

Сериализация объекта - это способность объекта сохранять полную копию его и любых других объектов на которые он ссылается

Объект может быть воссоздан из сериализованной копии – десериализация

XML - Extensible Markup Language

Элементы, символьные данные

Древовидная структура документа

Корневой элемент, дочерние элементы и листья, атрибуты

Правила:

наличие корневого элемента;

каждый открывающий тег имеет соответствующий закрывающий тег;

правильное вложение элементов документа;

атрибут должен иметь значение, которое берется в кавычки.

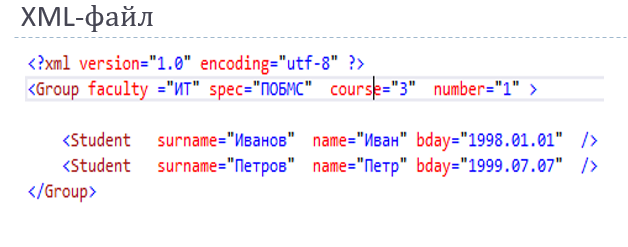
* Является подмножеством языка SGML – Standard Generalized Markup Language – метаязыка для определения языков разметки
* W3C – стандартизация
* Консорциум Всемирной паутины  - World Wide Web Consortium – организация, разрабатывающая и внедряющая технологические стандарты для web
* Глава – Тимоти Джон Бернерс-Ли
* Ок. 15 стандартов утверждены для XML:
  + XML Schema
  + XPath
  + XSLT
  + XQuery
* XML Schema — язык описания структуры XML-документа – предназначен для определения правил, которым должен подчиняться документ
* Создается модель данных документа, которая включает:
  + словарь (названия элементов и атрибутов);
  + модель содержания (отношения между элементами и атрибутами и их структура);
  + типы данных.
* Файл, содержащий XML Schema, обычно имеет расширение .xsd
* XPath - XML Path Language — язык запросов к элементам XML-документа.

Разработан для организации доступа к частям документа XML в файлах трансформации XSLT

* XSLT  — eXtensible Stylesheet Language Transformations — язык преобразования XML-документов

Правила выбора и преобразования данных пишутся на языке запросов XPath

* XQuery — язык запросов, разработанный для обработки данных в формате XML
* XML-языки – OASIS
* OASIS  — Organization for the Advancement of Structured Information Standards — глобальный консорциум
* Управляет разработкой и принятием промышленных стандартов электронной коммерции
* Синтаксические правила построения:
* Наличие корневого элемента;
* Каждый открывающий тег имеет соответствующий закрывающий тег;
* Правильное вложение элементов документа;
* Атрибут должен иметь значение, которое берется в кавычки
  + Древовидная структура документа



1. Сериализация и десериализация данных. Формат JSON. Работа с GSON в ОС Android.

Сериализация - это процесс сохранения состояния объекта в последовательность бит

Десериализация - это процесс восстановления объекта из последовательности бит

Сериализация объекта - это способность объекта сохранять полную копию его и любых других объектов на которые он ссылается

Объект может быть воссоздан из сериализованной копии – десериализация

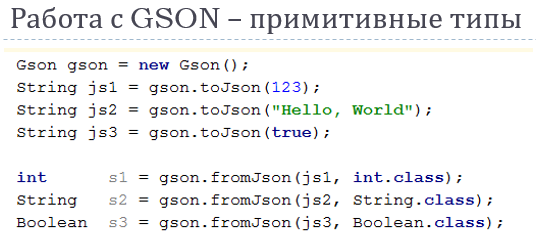
JSON  - JavaScript Object Notation — текстовый формат обмена данными

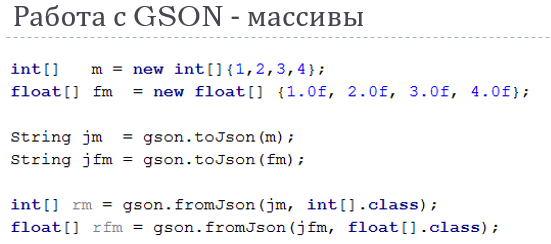
Объект, массив, строка, число, литералы, ключ: значение

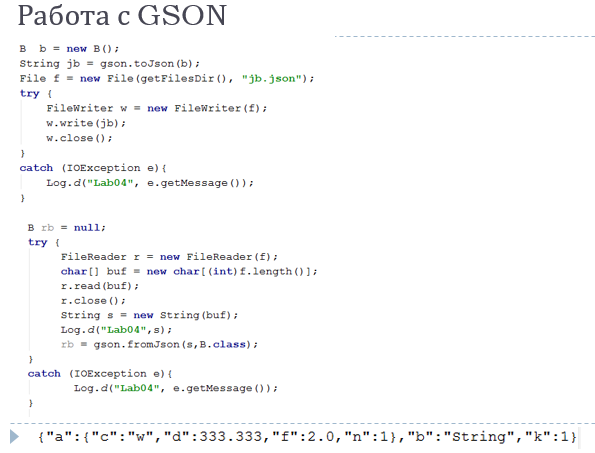


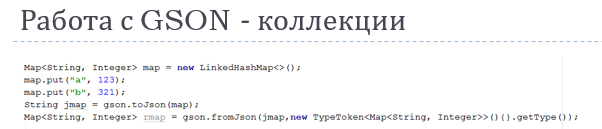
* Объект - неупорядоченное множество пар {ключ:значение}
* Ключ - строка
* Строка — это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки
* Число в десятичном формате
* Пары ключ-значение отделяются запятыми
* Массив - упорядоченное множество значений
* Массив заключается в [квадратные скобки]
* Значения в массиве разделяются запятыми
* Литералы true, false и null
* Основной класс – Gson
* Основные методы – toJson и fromJson



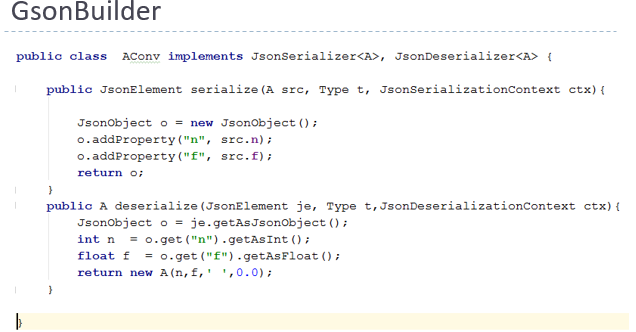


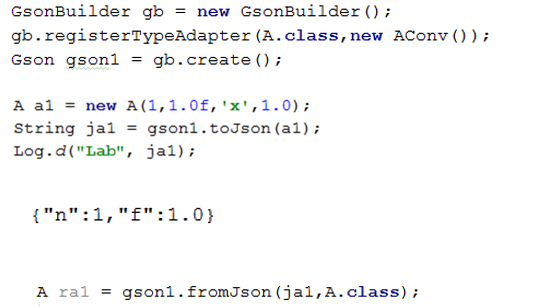






* GsonBuilder – разработка собственного сериализатора





1. SQLite. Создание и удаление SQLite - базы данных в ОС Android.

**База данных** (БД) - это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств.

**Система управления базами данных** (СУБД) - это программа, позволяющая создавать базы данных, а также обеспечивающая обработку (сортировку) и поиск данных.

**Сервер базы данных** - сервер БД обслуживает базу данных и отвечает за целостность и сохранность данных, а также обеспечивает операции ввода вывода при доступе клиента к информации.

**Реляционная база данных** - это базы, где вся информация хранится в таблицах, связанных друг с другом специальными отношениями.

**Масштабируемость** (англ. scalability) — в электронике и информатике означает способность системы, сети или процесса справляться с увеличением рабочей нагрузки (увеличивать свою производительность) при добавлении ресурсов (обычно аппаратных).

**Вертикальное масштабирование** — увеличение производительности каждого компонента системы с целью повышения общей производительности. Масштабируемость в этом контексте означает возможность заменять в существующей вычислительной системе компоненты более мощными и быстрыми по мере роста требований и развития технологий. Это самый простой способ масштабирования, так как не требует никаких изменений в прикладных программах, работающих на таких системах.

**Горизонтальное масштабирование** — разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию. Масштабируемость в этом контексте означает возможность добавлять к системе новые узлы, серверы, процессоры для увеличения общей производительности. Этот способ масштабирования может требовать внесения изменений в программы, чтобы программы могли в полной мере пользоваться возросшим количеством ресурсов.

Характеристики СУБД

* архитектура;

многопользовательская удаленная (клиент-сервер)

* носители данных;
* SQL/noSQL;
* стандарты;
* типы данных;
* ограничения целостности (констрейнты);
* объекты БД;
* встроенная процедурная система программирования;
* программные объекты;
* программные интерфейсы;

Для взаимодействия клиентского приложения с сервером MSS служит специальная DLL-библиотека, называемая SNAC (SQL Native Access Client) и включающая в себя программные интерфейсы: провайдер OLEDB (Object Linking and Embedding, Database) и драйвер ODBC (Open Data Connectivity). Приложения, предназначенные для работы в среде .NET, должны использовать специальный программный интерфейс – ADO.NET, основанный на провайдерах данных (программная компонента, обеспечивающая соединение с сервером БД, выполнение команд и получение результатов их выполнения) платформы .NET Framework.

Для доступа Java-приложения в MSS служит программный интерфейс JDBC (Java Database Connectivity), в основе работы которого лежит JDBC-драйвер, обеспечивающий соединение и взаимодействие с сервером MSS.

* типы данных

**SQLite**

**Кроссплатформенное** (межплатформенное) **программное обеспечение** — программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе. **SQLite** — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL. Как известно, в своем развитии SQL устремился в разные стороны. Крупные производители начали впихивать всякие расширения. И хотя принимаются всякие стандарты (SQL 92), в реальной жизни все крупные БД не поддерживают стандартов полностью + имеют что-то свое. Так вот, SQLite старается жить по принципу «минимальный, но полный набор». Она не поддерживает сложные штуки, но во многом соответствует SQL 92. Нельзя удалить или изменить столбец в таблице (ALTER TABLE DROP COLUMN…, ALTER TABLE ALTER COLUMN…).

Есть триггеры, но не настолько мощные как у крупных RDBMS.

Есть поддержка foreign key, но по умолчанию — она **ОТКЛЮЧЕНА**.

Нет встроенной поддержки UNICODE (но ее, в общем, нетрудно добиться).

Нет хранимых процедур.

**Характеристика SQLite:**

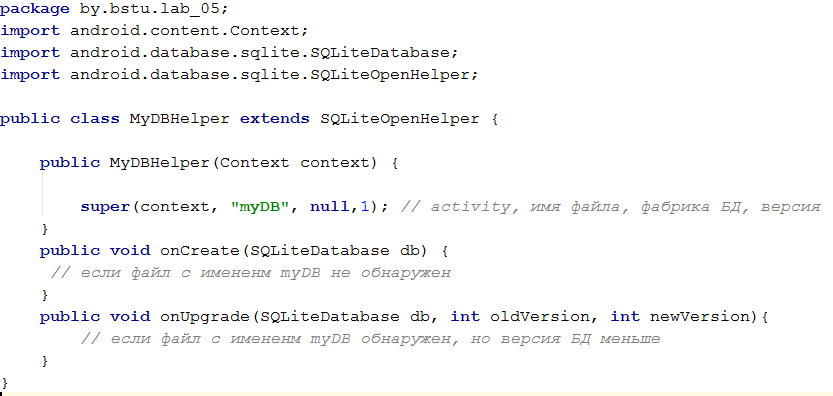
* встроенная, однопользовательская;
* файловая система: БД – это обычно 1 файл;
* ограниченный SQL;
* ограниченный SQL 92;

Начиная с 1986 года, комитеты ISO (International Organization for Standardization) и ANSI (American National Standards Institute) приступили к созданию ряда стандартов языка SQL, которые впоследствии были приняты и получили следующие названия: SQL86, SQL89, SQL92 и SQL99. **Стандарт SQL86** зафиксировал минимальный стандартный синтаксис языка SQL. **Стандарт SQL89** был принят в 1989 году. Он вводил набор операторов языка SQL, которые должны были реализовывать все СУБД, заявляющие поддержку стандарта SQL89. При создании схемы базы данных (впоследствии можно изменить) можно определить особенности национального набора символов, включая правила упорядочения, при этом могут определяться наборы символов, используемые как в хранимых текстовых строках, так и в идентификаторах. В новом стандарте появилась возможность создавать хранимые и представляемые таблицы и задавать или удалять привилегии доступа при помощи операторов (CREATE TABLE, CREATE VIEW, GRANT, REVOKE) в любой момент времени в любой транзакции вне оператора определения схемы. Появились операторы уничтожения таблиц (DROP TABLE и DROP VIEW), которые также можно выполнять внутри любой транзакции (при наличии соответствующих привилегий). Также впервые определён оператор ALTER TABLE, позволяющий динамически изменять характеристики ранее созданной таблицы (в частности добавлять к ней новые столбцы).

* практически не строго типизируемая;
* есть некоторые объекты (индексы, view);
* встроенной системы программирования нет;
* программные объекты: триггеры;
* программный интерфейс: библиотека функций;
* типы данных: **INTEGER** (все целочисленные), **TEXT** (все текстовое), **BLOB** (бинарные данные), **REAL** (с плавающей точкой), **NUMERIC** (с фиксированной точкой).

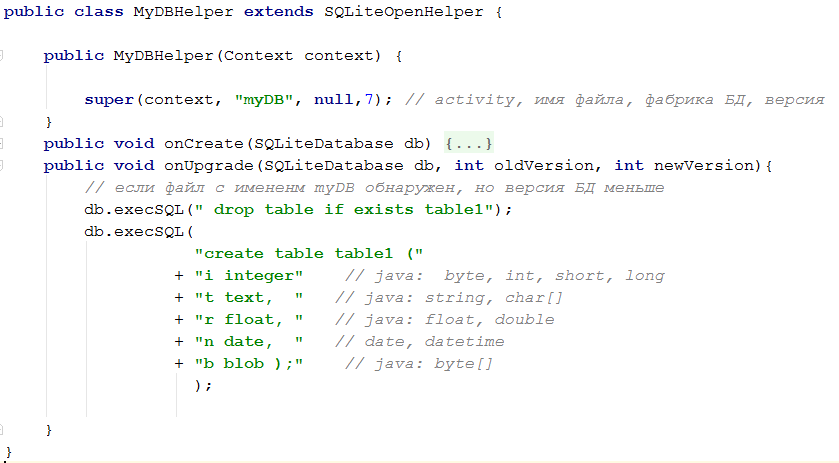
Для работы с БД в Андроид-приложении обычно создается класс-потомок SQLiteOpenHelper. В этом классе следует переопределить методы onCreate() для создания БД и onUpgrade() для обновления БД в случае изменений в схеме базы данных. Оба метода работают с объектом SQLiteDatabase.

SQLiteOpenHelper предоставляет методы getReadableDatabase() и getWriteableDatabase() для доступа к объекту SQLiteDatabase, который позволяет читать и писать в БД.









**SQLite:** не типы данных, а класс хранения, в любом классе можно хранить данные любого типа.

**SQLite классы хранения:** **NULL**(пустое значение), **INTEGER** (1,2,3,4,5,8 – байтов, целые числа), **REAL** (IEE-752), **TEXT** (UTF-8(1 байт), UTF-16(BE/LE)(2 байта)) , **BLOB** (как ввел).

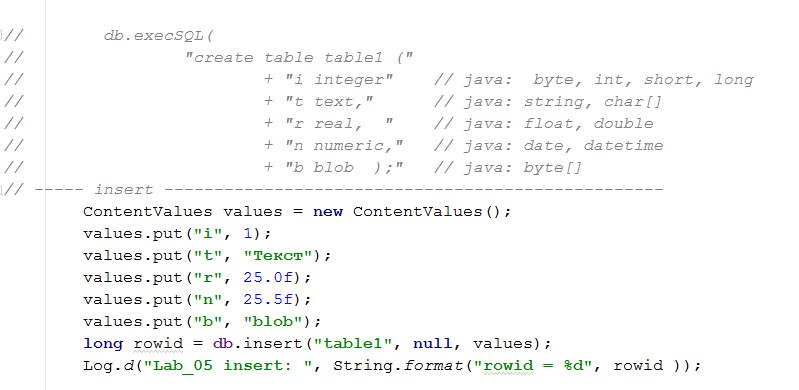
Один символ кодировки UTF-16 представлен последовательностью двух байтов или двух пар байтов. Который из двух идёт впереди, старший или младший, зависит от порядка байтов. Систему, совместимую с процессорами x86, называют little endian, а с процессорами m68k и SPARC — big endian.

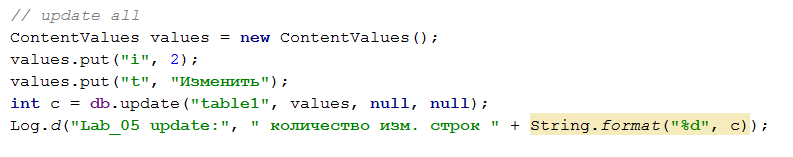
**SQLite:** концепция аффинированных типов

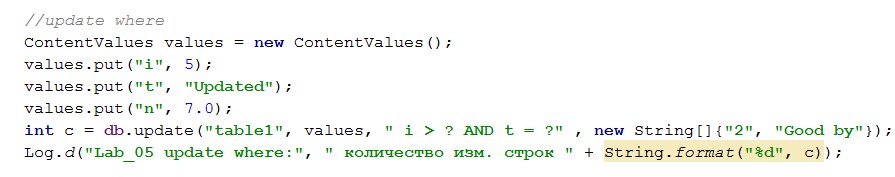
**Аффинированный тип** — это тот тип, который является рекомендуемым для сохраняемых в столбце значений. Важной идеей здесь является то, что **тип рекомендуется, но не является обязательным**. В любом столбце можно по-прежнему хранить данные любого типа. Просто в некоторых столбцах, при наличии выбора, одному классу хранения будет оказано предпочтение перед другим. Предпочтительный класс для хранения данных в столбце называется аффинированным.

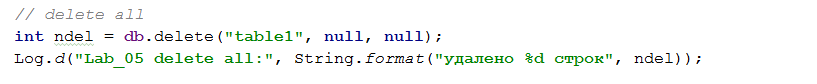
**SQLite аффинированные типы:** **TEXT** (BLOB, CHAR, TEXT), **NUMERIC** (в остальных случаях), **INTEGER** (INT), **REAL** (REAL, FLOAT, DOUBLE), **NONE**.

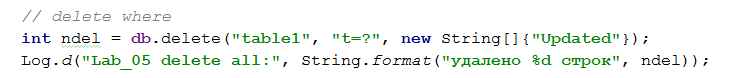
1. SQLite. DML-операции в SQLite - базе данных в ОС Android.

****

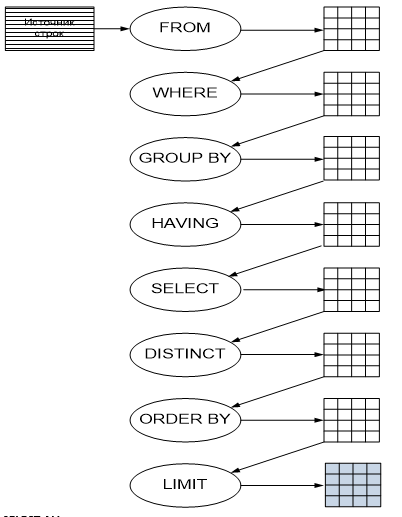
****

****

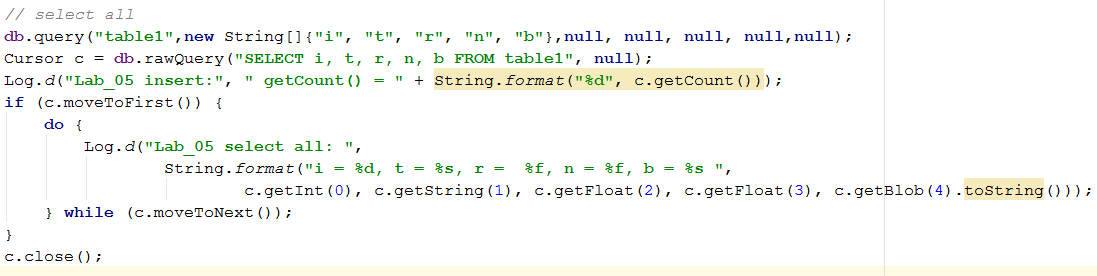
****

****



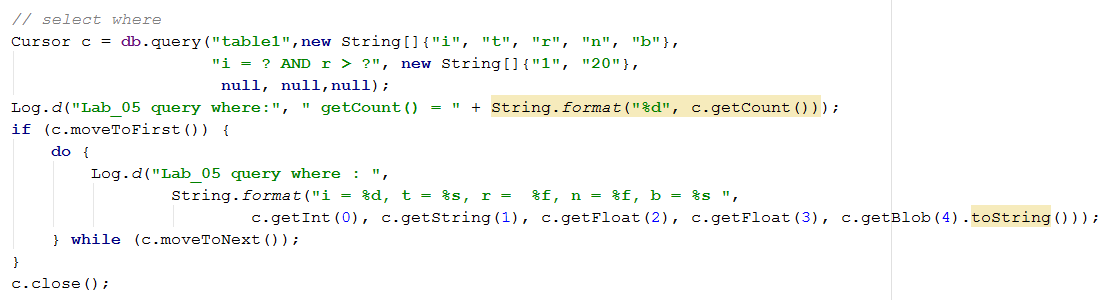


**SELECT ALL**

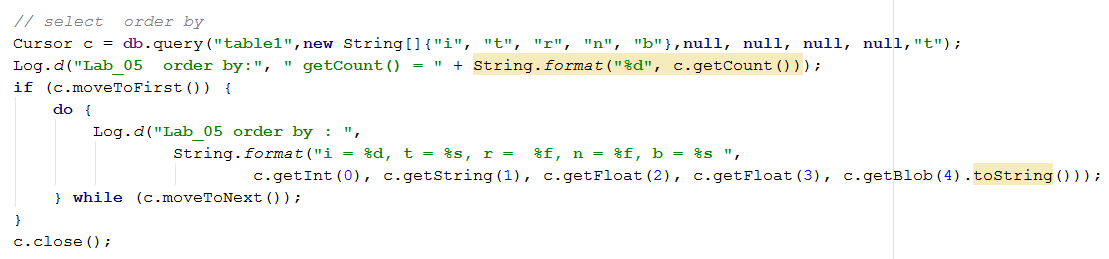
****

**SELECT WHERE**

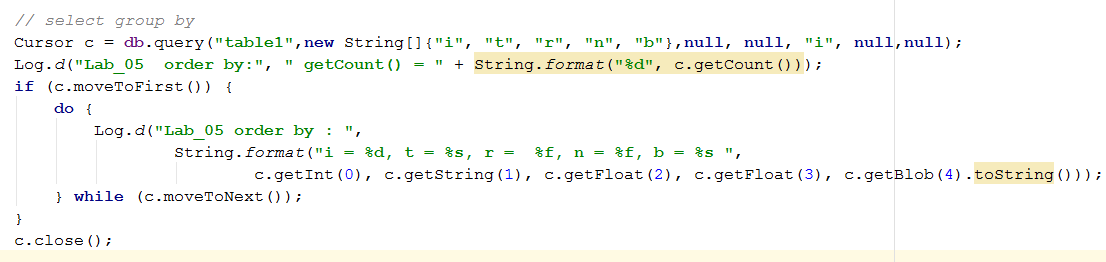
rawQuery("select \* from todo where \_id = ?", new String[] { id });

****

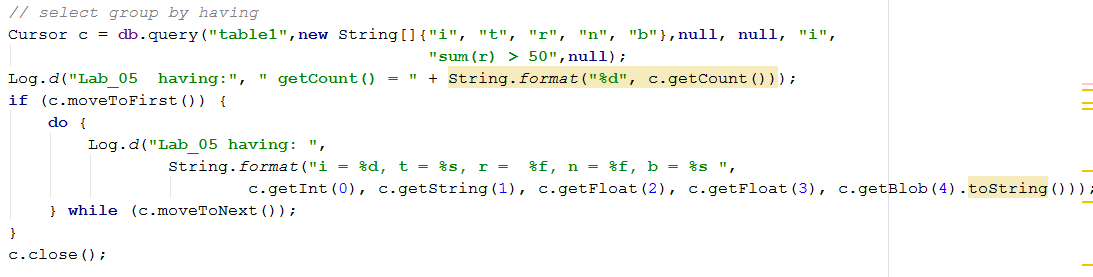
**SELECT ORDER BY**

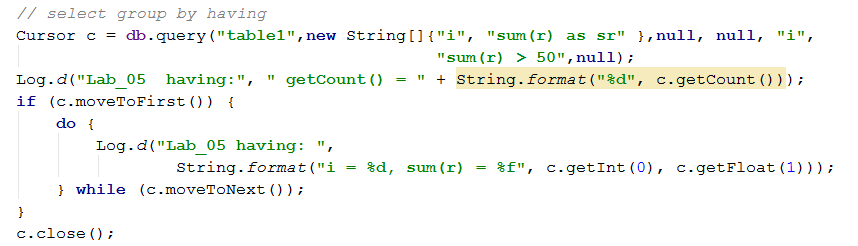
****

**SELECT GROUP BY**

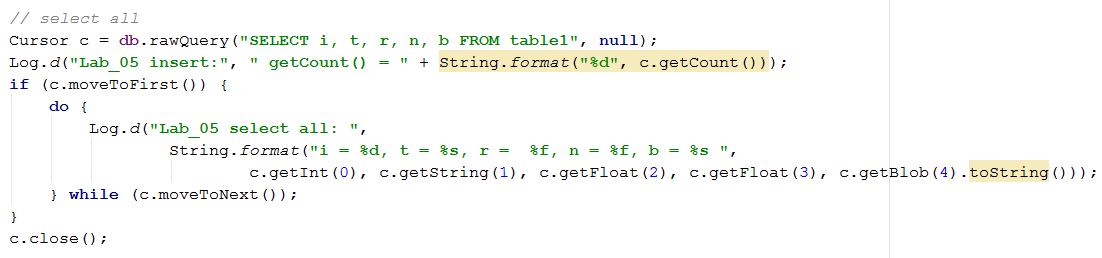
****

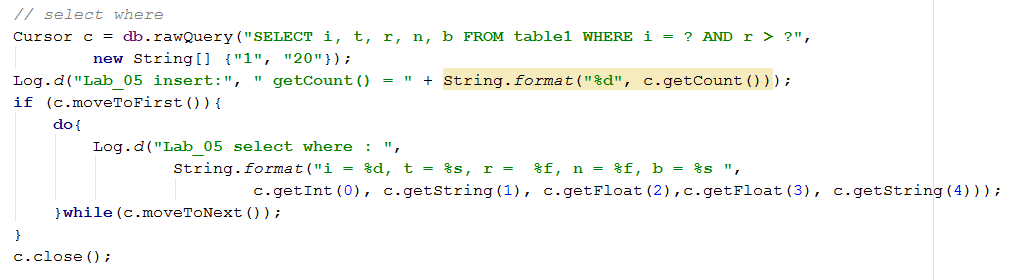
**SELECT GROUP BY HAVING**

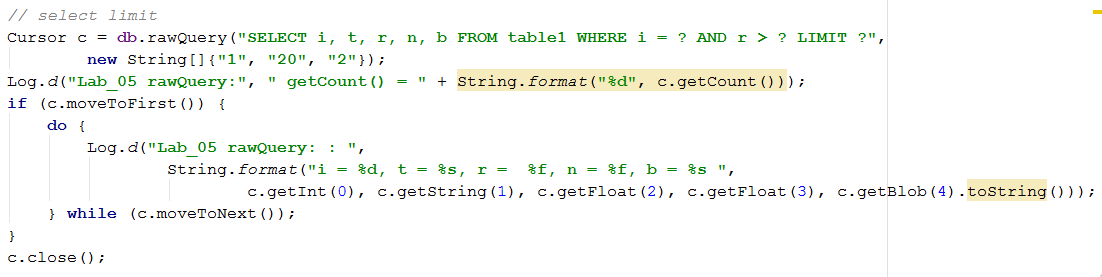
****

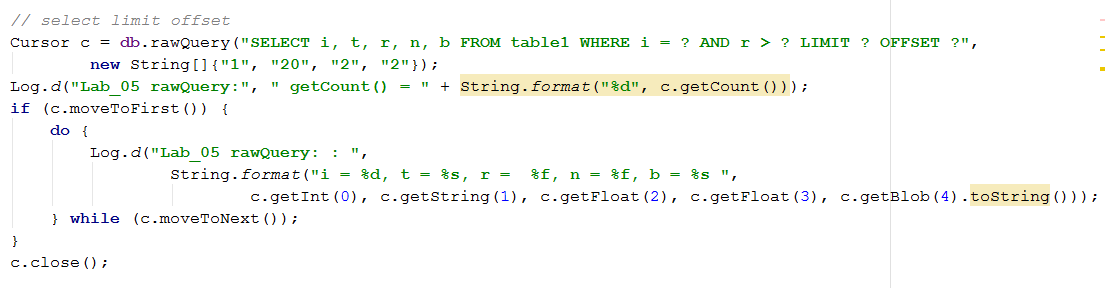
****

**RAWQUERY**

****

****

****

****

1. SQLite. Классы хранения и аффинированность типов.

Аффинированный тип — это тот, который является рекомендуемым для сохраняемых в столбце значений

Не является обязательным

Аффинированность столбца определяется неявно во время создания таблицы при объявлении типа данных, который будет храниться в столбце

**NULL** (пустое значение)

**INTEGER** (1,2,3,4,5,8 – байтов, целые числа)

**REAL** (IEE-752)

* Могут использоваться для хранения значений всех классов хранения
* Если поместить текстовое значение, то SQLite постарается преобразовать его в INTEGER или REAL
* Если преобразование не может произойти, то SQLite сохранит значение с классом TEXT
* Для REAL – целочисленные значения будут преобразованы в числа с плавающей точкой

**TEXT** (UTF-8, UTF-16)

* предназначены для хранения значений с классом NULL, TEXT или BLOB
* Если добавить в столбец с аффинированным типом данных TEXT числовое значение, то оно будет преобразовано в строку

**BLOB**

* не предпочитает какой-либо класс хранения
* не предпринимает попыток преобразования данных из одного класса в другой

1. SQLite. Особенности сравнения, сортировки и группировки дат, строк и числовых данных в SQLite - базе данных в ОС Android.

Допустимо сравнивать значения с разными типами данных

Нет булевого типа данных – целые числа 0 (false) и 1 (true)

Нет типа данных даты и времени – хранится тип TEXT в виде строки формата: «YYYY-MM-DD HH:MM:SS.SSS»

**REAL** как числа юлианского календаря - число дней с полудня 24 ноября 4714 г. до н.э.

**INTEGER** как время Unix - количество секунд с 1970-01-01 00:00:00 UTC.

«=», «==», «<», «<=», «>», «>=», «!=», «<>», «BETWEEN», «IN», «IS», «NOT IN» и «IS NOT»

Значения с классом NULL считаются меньше любого другого значения, даже другого значения NULL

Значение INTEGER или REAL считается меньше, чем значение TEXT или BLOB

При сравнении INTEGER или REAL используется числовое сравнение

Значение TEXT меньше значения BLOB

Сравнение значений BLOB определяется с использованием функции memcmp()

Выражение, которое является ссылкой на значение столбца, имеет такую же аффинированность, как и столбец.

Выражение "CAST (*выражение* AS *тип*)" имеет ту же аффинированность

{INTEGER REAL NUMERIC} и {TEXT BLOB} 🡪 NUMERIC;

{TEXT} и { } 🡪 TEXT;

В остальных случаях преобразование не применяется

Все математические операторы преобразуют операнды в NUMERIC до выполнения операций

Преобразование происходит даже в том случае, если оно необратимое и с потерями

Операнд NULL приводит к пустому (NULL) результату любую математическую операцию

Операнд в математической операции, который не приводится каким-либо образом к числовому и не является NULL, преобразуется в 0 или 0.0.

* a BETWEEN b AND c
  + a >= b AND a <= c
  + в каждом из сравнений применяются различные аффинированности

1. SQLite. Операторы объединения и соединения в SQLite - базе данных в ОС Android.

INNER JOIN

CROSS JOIN

JOIN … USING

NATURAL JOIN

Self JOIN

OUTER JOIN

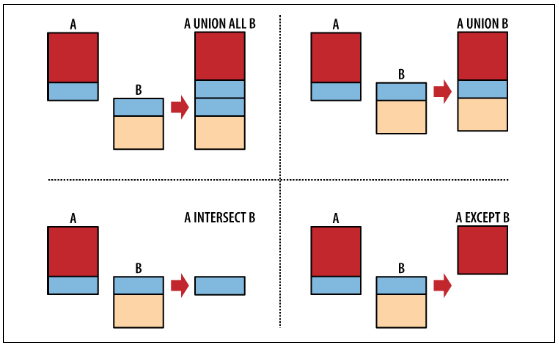
* Поддерживается только LEFT OUTER JOIN
* Поддерживаются все конструкции USING, NATURAL, ON
* Как получить FULL OUTER JOIN?

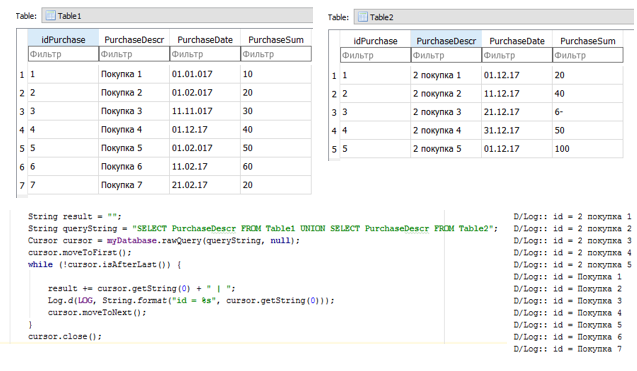
UNION ALL

UNION

INTERSECT

EXCEPT





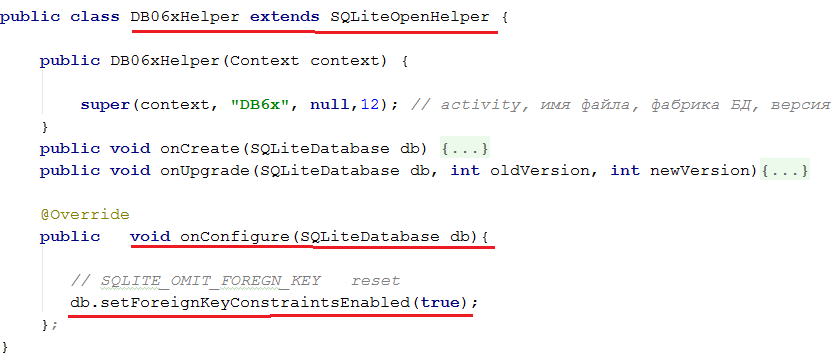
1. SQLite. Использование обобщённых табличных выражений.

временно именованный результирующий набор. Он получается при выполнении простого запроса и определяется в области выполнения одиночной инструкции SELECT, INSERT, UPDATE, DELETE или MERGE. Это предложение может использоваться также в инструкции CREATE VIEW как часть определяющей ее инструкции SELECT. Обобщенное табличное выражение может включать ссылки на само себя. Такое выражение называется рекурсивным обобщенным табличным выражением.

-Табличными выражениями называются подзапросы, которые используются там, где ожидается наличие таблицы

1. SQLite. Ограничения целостности в SQLite - базе данных в ОС Android.
2. SQLite. Указания PRAGMA в SQLite - базе данных в ОС Android.

Команда PRAGMA позволяет включить внешние ключи в базах данных SQLite

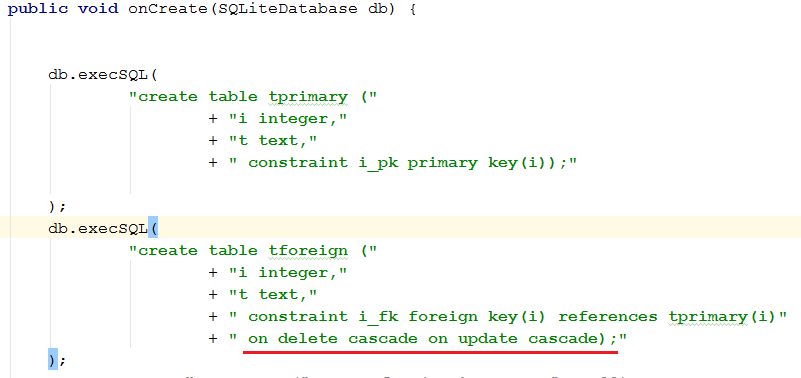


После включения внешнего ключа мы можем использовать:

* RESTRICT – явное указание на запрет изменять или удалять PRIMARY KEY значение, если на него ссылаются FOREIGN KEY – действует по умолчанию
* NO ACTION – явное указание на то, что никакие не предпринимать при изменении или удалении PRIMARY KEY значения (фактически отсутствует ограничение)
* SET NULL

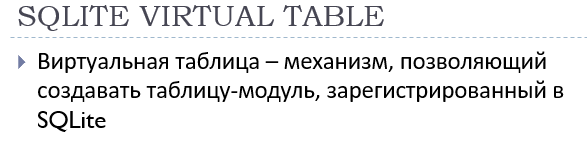


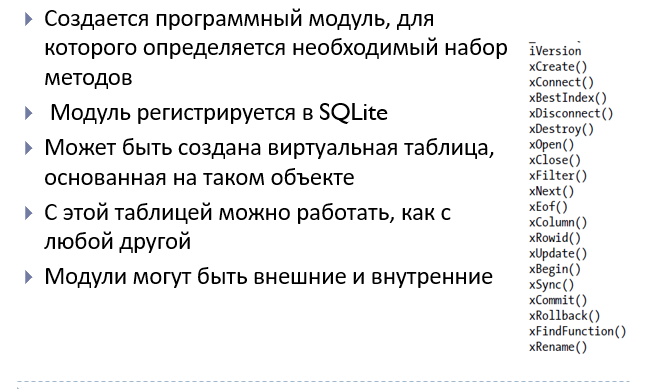
* SET DEFAULT
* CASCADE

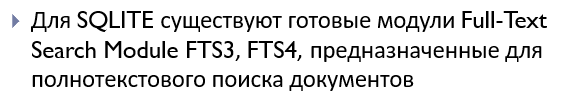
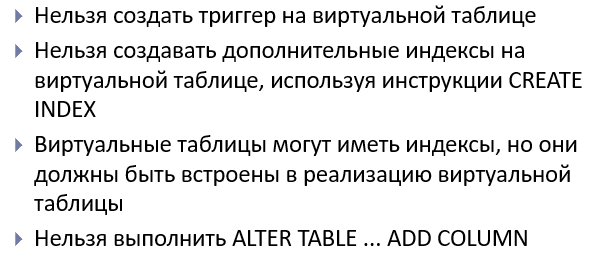
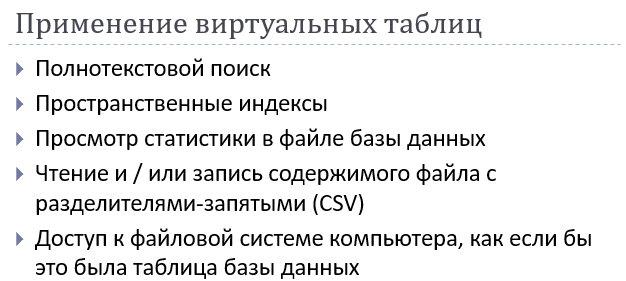


1. SQLite. Виртуальные таблицы в SQLite - базе данных в ОС Android.

Это обьект , который зарегистрирован при открытом подключении к бд . Вызывает методы обратного вызова обьекта виртуальной таблицы вместо чтения и записи.



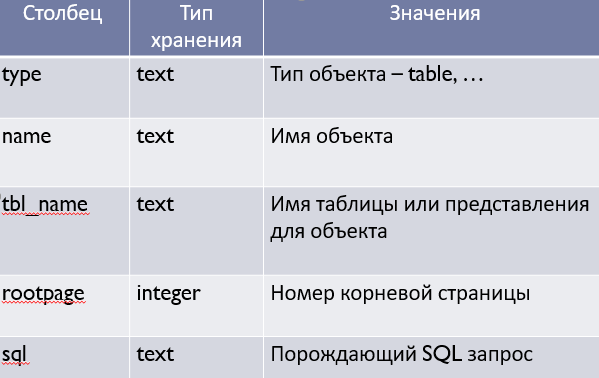


1. SQLite. Внутренние таблицы в SQLite - базе данных в ОС Android.



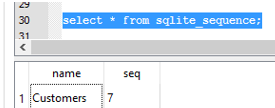
* SQLITE\_MASTER
* Таблица с метаданными
* Индекс PRIMARY KEY не показывается через SQLITE\_MASTER
* Индекс по ROWID не виден в SQLITE\_MASTER



* SQLITE\_SEQUENCE

необходима для реализации **AUTOINCREMENT**

можно добавлять, изменять и удалять строки из таблицы, но нельзя удалять саму таблицу



* SQLITE\_STATN
* Таблицы сбора статистики

1. SQLite. Индексы в SQLite - базе данных в ОС Android.

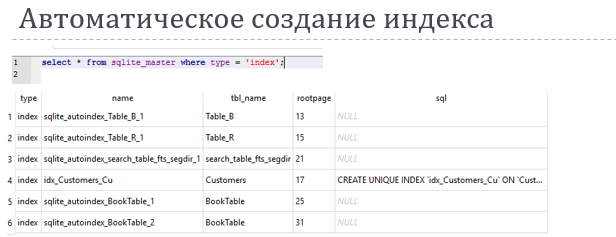
Интеграция индекса с таблицей:

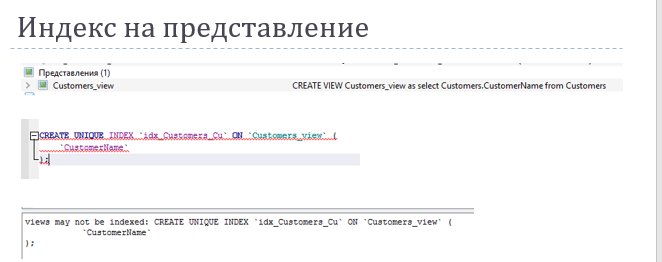
кластеризованные

некластеризованные

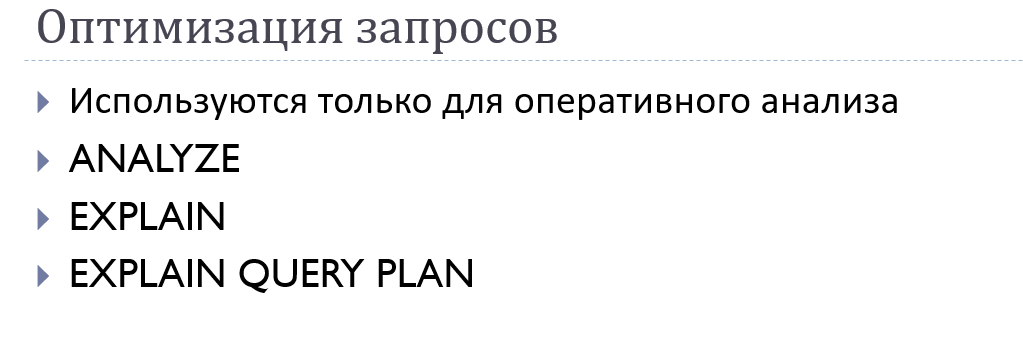
Структура индекса:

* + деревья (tree-index)
  + двоичные таблицы (bitmap, Oracle)
  + пространственные индексы (spatial)
  + полнотекстовые индексы (full text)
  + XML-индексы
  + колоночные индексы (для OLAP-приложений)
* Индекс строится на столбцах одной таблицы
* На представлении индекс построить нельзя
* На виртуальной таблице индекс построить нельзя
* Нет ограничений на количество индексов для одной таблицы
* Количество столбцов в индексе ограничено
* Если используется ключевое слово UNIQUE дублирование записей индекса не допускается
* При указании UNIQUE и PRIMARY KEY
* Не могут быть удалены DROP INDEX
* Показаны в sqlite\_master
* Индекс по RowID не показывается в sqlite\_master





1. SQLite. Возможности оптимизации запросов в SQLite - базе данных в ОС Android.

ANALYZE -Сбор статистики о таблицах и индексах

Хранится в sqlite\_stat1

Оптимизатор запросов может получить доступ к информации и использовать ее, чтобы помочь улучшить выбор планирования запросов

EXPLAIN- Пошаговое исполнение оператора

Оператор при этом не выполняется

QUERY RAW - Показывает, каким образом будет проводится поиск в таблице, используемые индексы.

1. SQLite. Представления в SQLite - базе данных в ОС Android.

Представление – VIEW – поименованный SELECT-запрос

CREATE VIEW или CREATE TEMP VIEW для временного (Существует в рамках одного соединения)

* Структурируйте данные так, чтобы пользователи или классы пользователей находили естественные или интуитивно понятные.
* Ограничьте доступ к данным таким образом, чтобы пользователь мог видеть только ограниченные данные вместо полной таблицы.
* Суммируйте данные из различных таблиц, которые можно использовать для создания отчетов.

1. SQLite. Триггеры в SQLite - базе данных в ОС Android.

Триггер – это особая разновидность хранимых процедур, исполняемых в ответ на какое-то событие

Используется для:

Обеспечения целостности данных

Реализации сложной бизнес-логики

Аудита изменений

Каскадного удаления/обновления данных

* Классификация по группам событий:
  + DML
* Классификация по событиям:

INSERT

DELETE

UPDATE

INSERT OR UPDATE

INSERT OR UPDATE OR DELETE

* Классификация по моменту выполнения относительно события:

BEFORE

AFTER

INSTЕAD OFF

* Классификация по отношению к объекту выполнения:

Операторный – выполняется один раз для оператора

Строчный – выполняется для каждой строки

* Триггер INSTEAD OF применятся к представлениям

Ограничения:

Один триггер на одно запускающее триггеры действие (UPDATE, DELETE или INSERT)

* + DDL:
    - уровень сервера (если есть)
      * Создание объектов сервера
      * Изменение объектов сервера
      * Удаление объектов сервера
      * Подключение к серверу
    - уровень БД
      * Создание, изменение и и удаление объектов БД

1. SQLite. Транзакции и блокировки в SQLite-базе данных в ОС Android.

Группа операций, которая может быть выполнена или не выполнена вместе

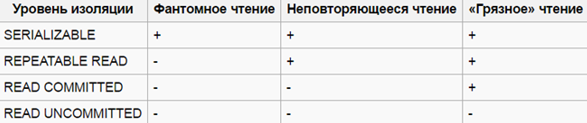
TCL:

COMMIT, ROLLBACK

* Транзакция – ACID:
* Atomicity
* Consistency
* Isolation
* Durability
* Изолированность – свойство независимости результата выполнения транзакций от параллельно работающих транзакций

Транзакция – уровни изолированности:

* READ UNCOMITED
* READ COMMITTED
* REPEATABLE READ
* SERIALIZABLE
* SNAPSHOT



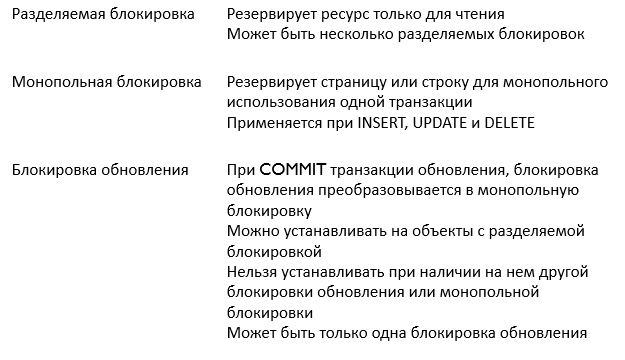
* Транзакция – режимы выполнения
* autocommit
* implicit transaction – режим неявных транзакций
* explicit transaction – режим явных транзакций

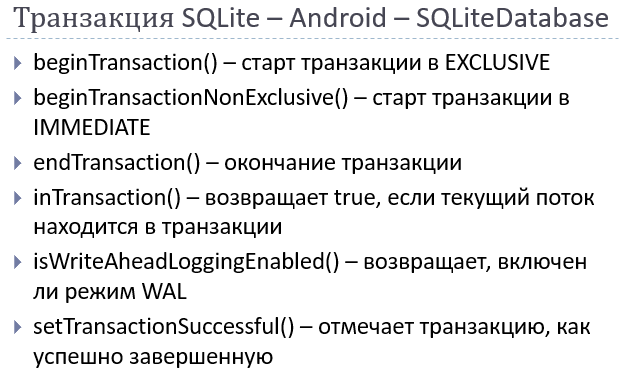
Транзакция – SAVEPOINT:

* Точка сохранения – контрольная точка – SAVEPOINT
* Точка сохранения определяет такую точку в транзакции, что все последующие изменения данных могут быть отменены без отмены всей транзакции
* SAVE TRANSACTION создает метку для последующей инструкции ROLLBACK, имеющей такую же метку, как и данная инструкция SAVE TRANSACTION

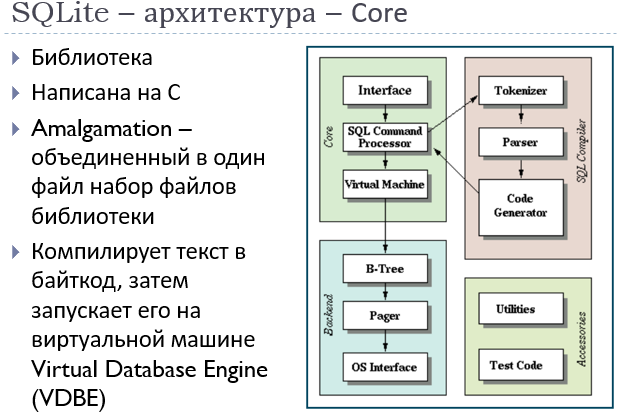
Транзакция – блокировки:

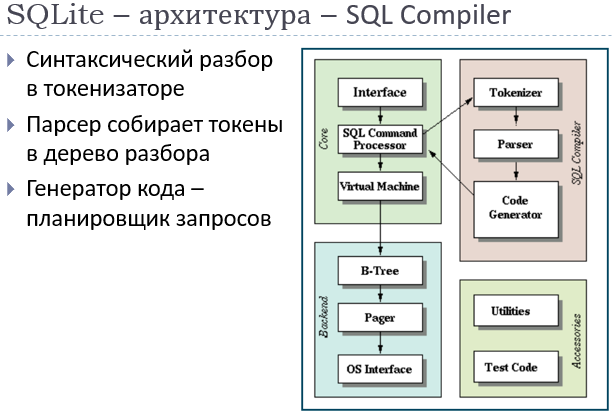
* Блокировки – механизм обеспечения согласованности данных в случае одновременного обращения к данным нескольких пользователей
  + Разделяемая (shared lock)
  + Монопольная (exclusive lock)
  + Обновления (update lock)
* Гранулярность блокировки определяет, какой объект блокируется
* Процесс преобразования большого числа блокировок уровня строки, страницы или индекса в одну блокировку уровня таблицы называется эскалацией блокировок

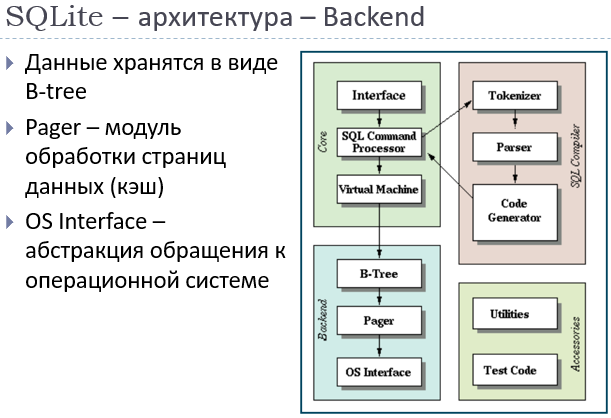


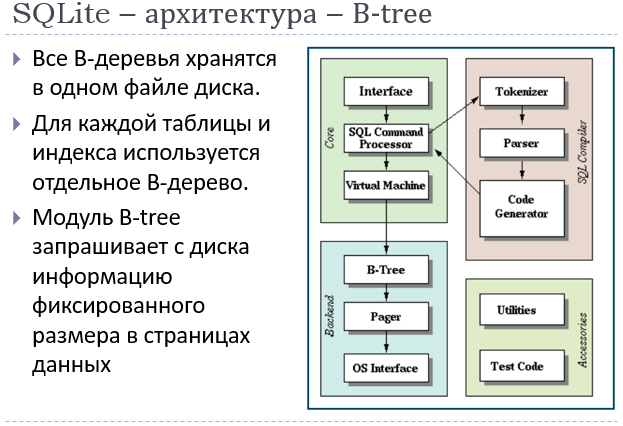


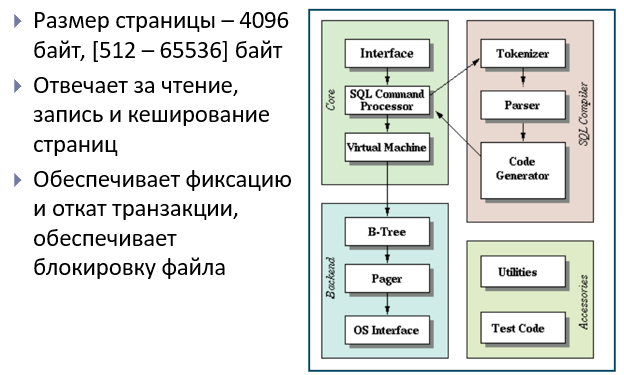
1. SQLite. Архитектура SQLite.











Архитектура разделения одновременного доступа:

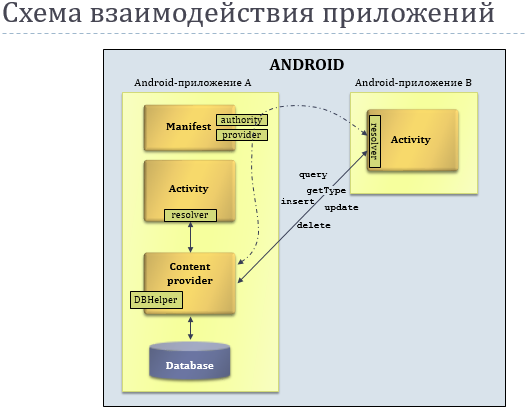


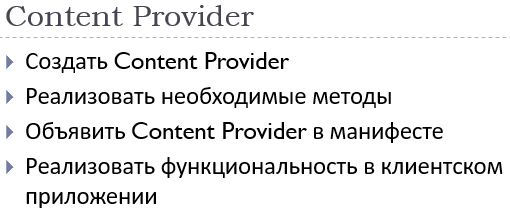
* Вначале в журнал отката записываются неизмененные данные и размер файла БД
* Затем изменяются данные в БД
* Затем стирается журнал

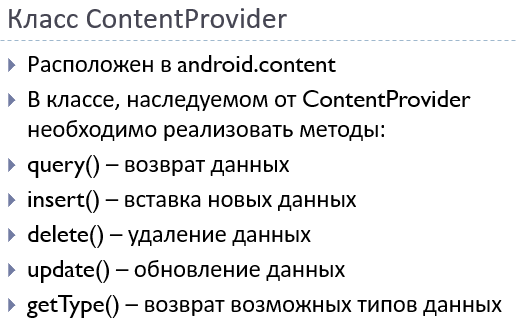
1. Концепция контент-провайдера. Применение контент-провайдера для работы с SQLite-базой данных в ОС Android.

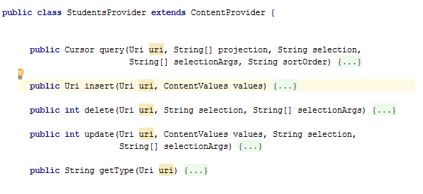
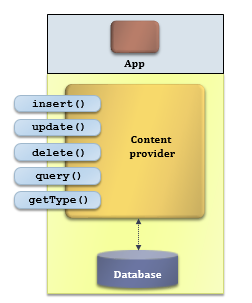
Content Provider – механизм, позволяющий обеспечить доступ нескольких приложений к общим данным

Используются встроенные поставщики – пакет android.provider

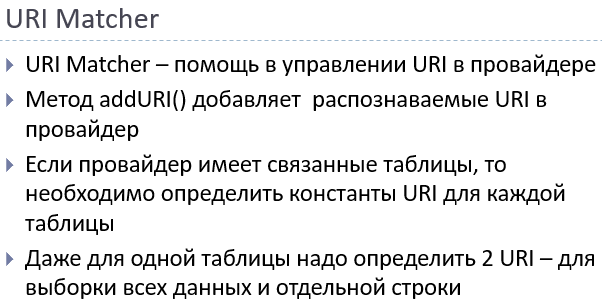
*  ContentProvider создается и регистрируется в системе как часть приложения
* Клиент вызывает ContentProvider через объект класса ContentResolver

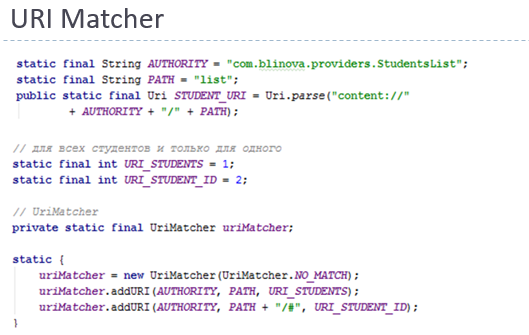






* URI — унифицированный идентификатор ресурса –строка, позволяющая идентифицировать ресурс
* **content://authority/Path/Id**
* authority – уникальная строка для каждого провайдера
* Path – выбор между различными наборами данных, которые провайдер может предоставить
* Id – указание на специфическую строку





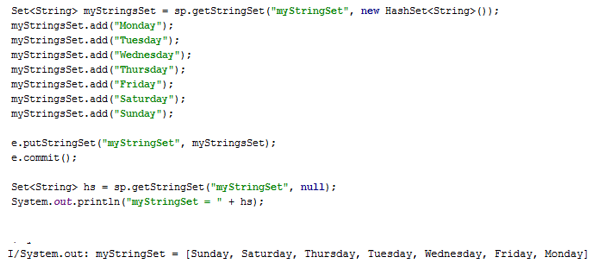
1. Концепция контент-провайдера. Применение контент-провайдера Contact в ОС Android.

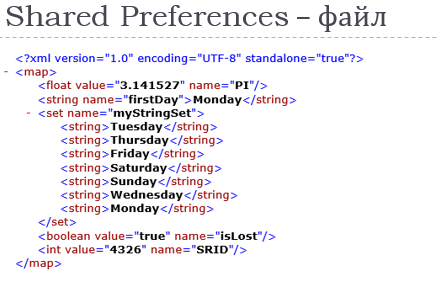
Content Provider – механизм, позволяющий обеспечить доступ нескольких приложений к общим данным

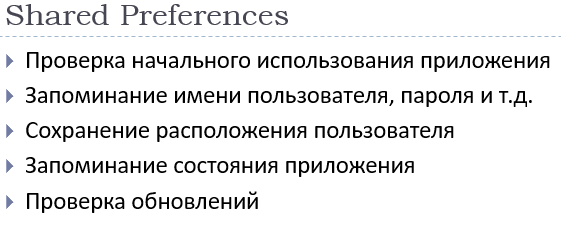
1. Хранение пар ключ-значение Shared Preferences для обеспечения работы приложения в ОС Android.

Shared Preferences – хранение пар ключ-значение

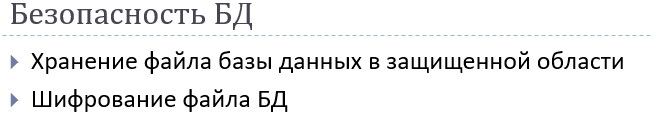


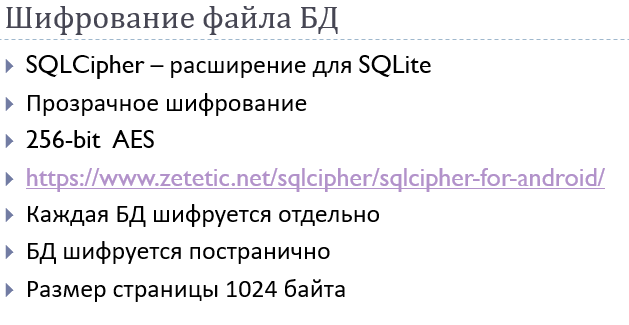




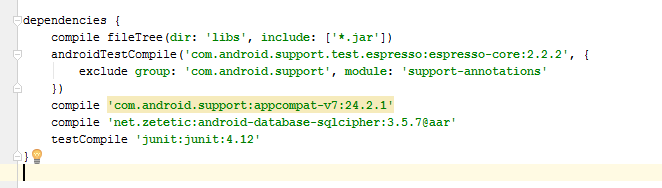


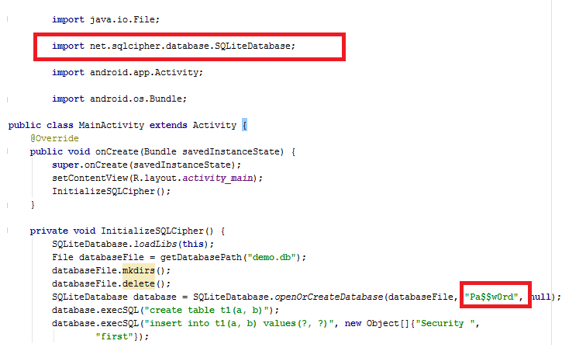
1. Аспекты безопасности локальных БД в ОС Android.

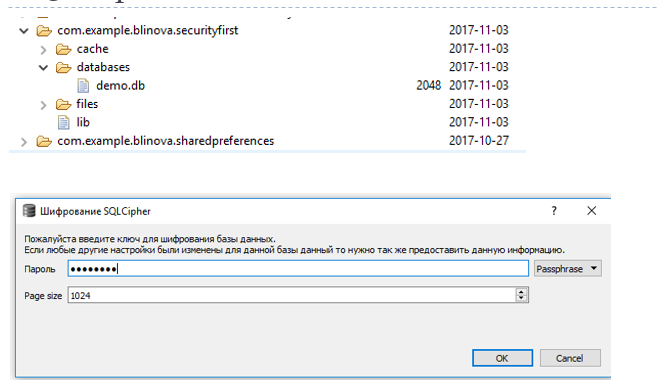




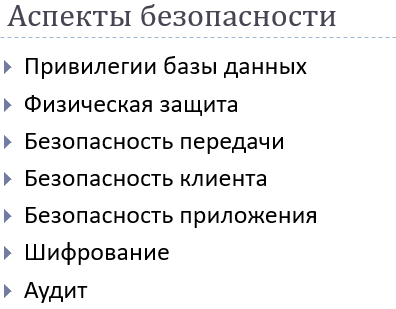
SQLCipher

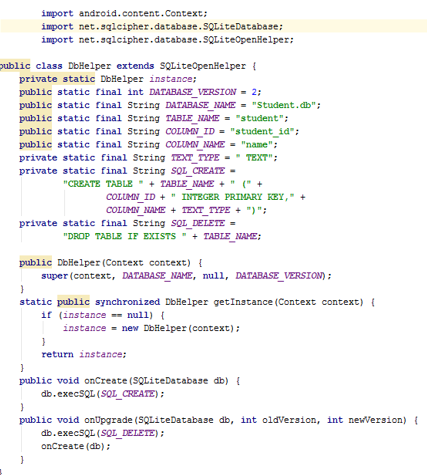


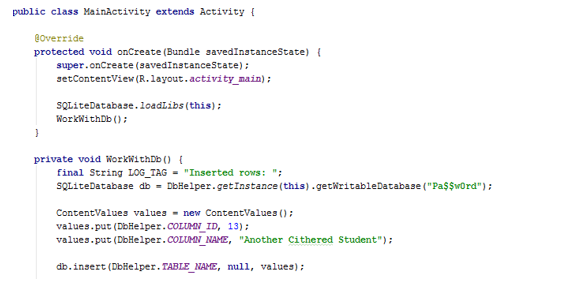


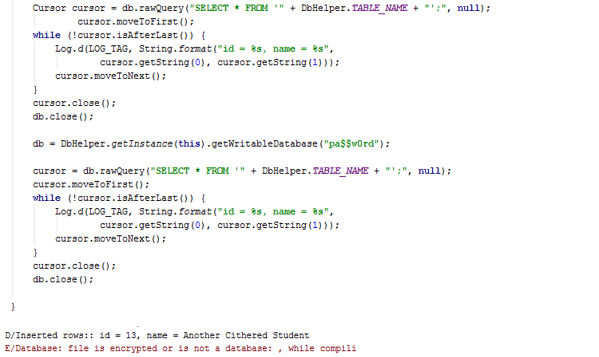


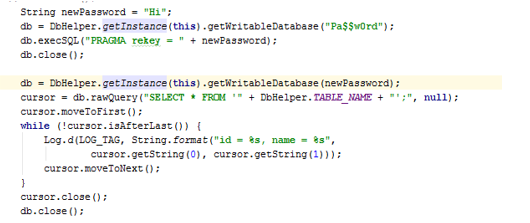








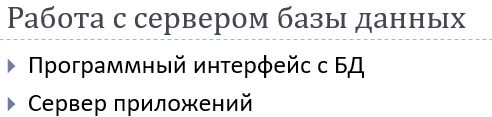


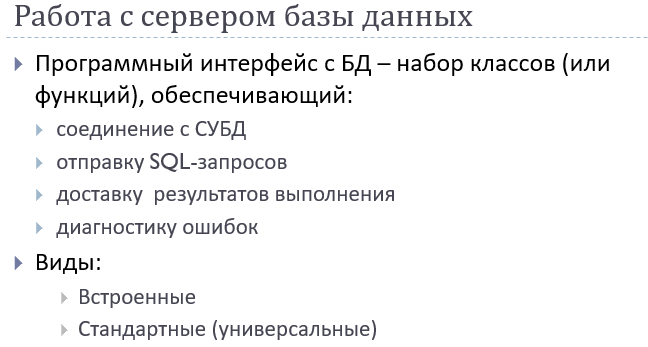


1. Обеспечение безопасности данных приложения в моделях с использованием удаленного сервера баз данных.

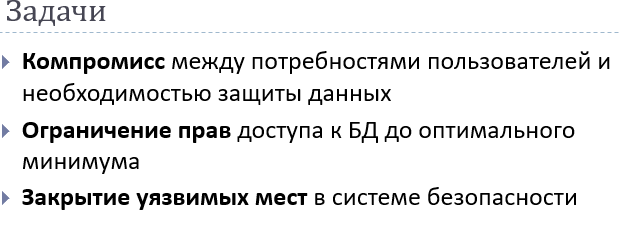


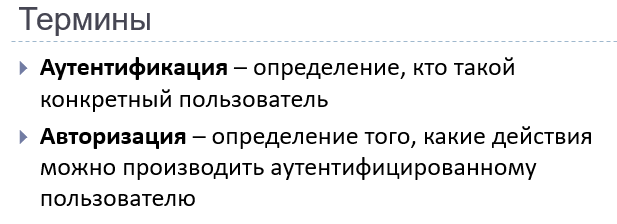


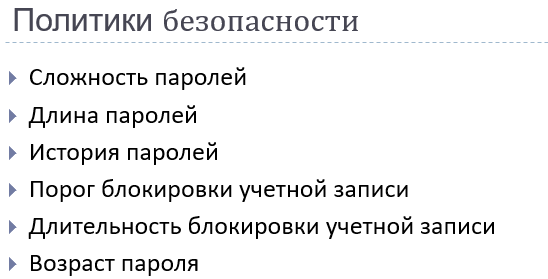




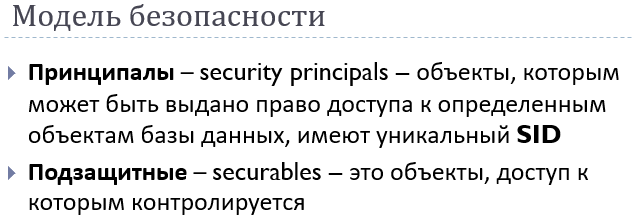
Аспекты безопасности удаленных БД:







* **Привилегия** – право выполнения определенной операции
* **Роль** – набор привилегий

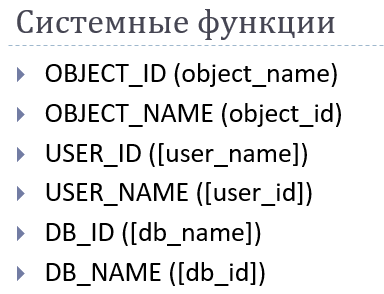


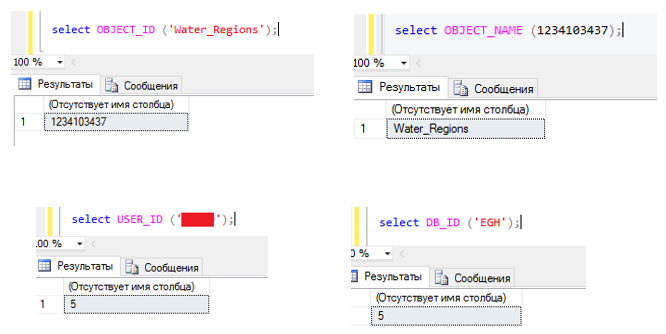
1. Удаленные серверы баз данных. Подключение к СУБД, типичные объекты баз данных и использование хранимых процедур.

Системные хранимые процедуры применяются для выполнения административных и пользовательских задач:

* переименование объектов
* идентификация пользователей
* мониторинг









1. Программные интерфейсы с СУБД. Применение интерфейса JDBC для работы с удаленными серверами СУБД в ОС Android.

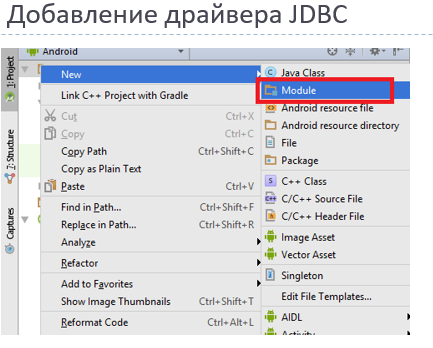
JDBC – Java Database Connectivity

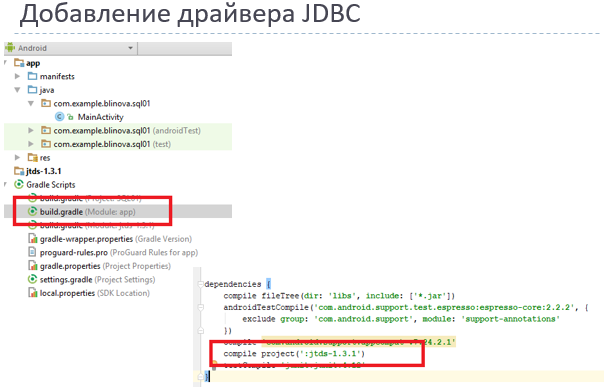
платформенно-независимый стандарт взаимодействия Java-приложений с СУБД

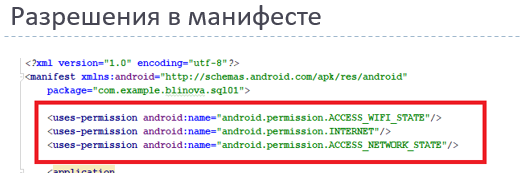
входит в состав Java SE

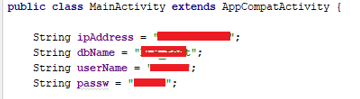
Имеет модель провайдера: стандартный набор классов и драйверы

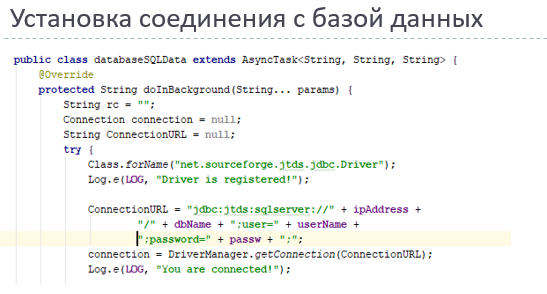
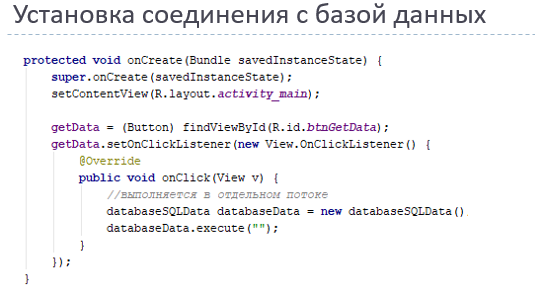
* Для подключения к отдельным базам данных JDBC требует наличия драйверов для каждой базы данных
* Для Microsoft SQL Server:
* <http://jtds.sourceforge.net/>
* <https://www.microsoft.com/ru-ru/download/details.aspx?id=11774>

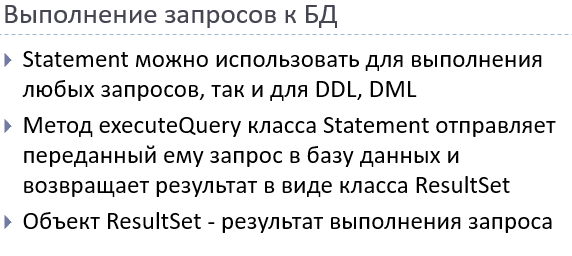




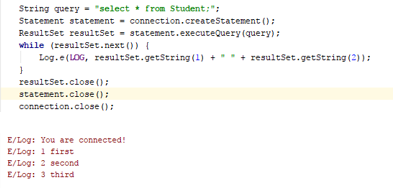


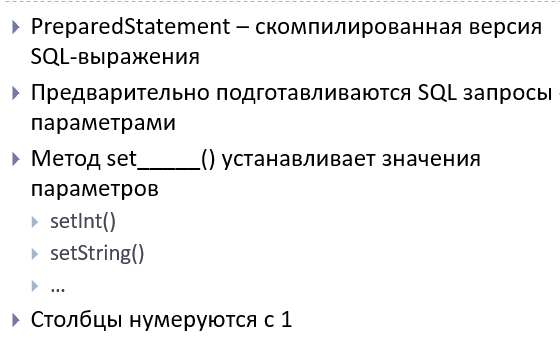


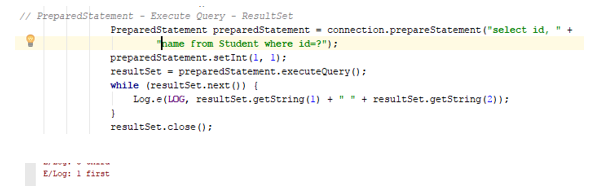


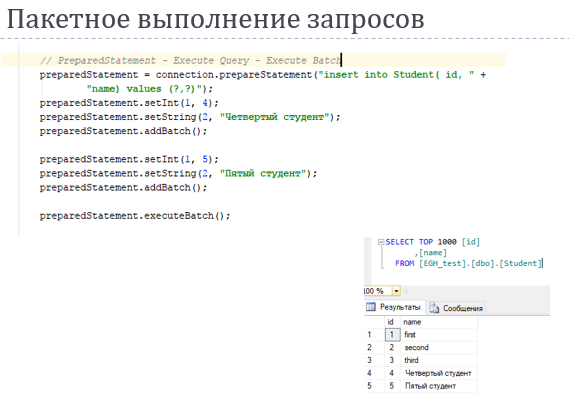
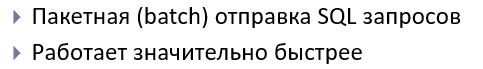






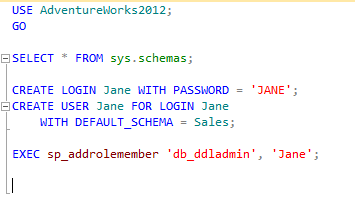








1. Обеспечение безопасности данных приложения в моделях с использованием удаленного сервера баз данных (логины, пользователи, роли, …).

* **Принципалы** – security principals – объекты, которым может быть выдано право доступа к определенным объектам базы данных, имеют уникальный **SID**
* **Подзащитные** – securables – это объекты, доступ к которым контролируется
* Операционная система
  + - Учетная запись домена Windows
    - Локальная учетная запись Windows
    - Группа Windows
* SQL Server
  + - Учетная запись SQL Server
    - Учетная запись SQL Server, сопоставленная с асимметричным ключом или сертификатом
* База данных
  + - Пользователь БД, сопоставленный с ключом или сертификатом
    - Пользователь БД, сопоставленный с учетной записью Windows
    - Роль приложения
    - Роль БД
* Сервер
  + - Текущий экземпляр сервера
    - БД
    - Учетная запись (login)
    - Роль сервера
* База данных
  + - Роли БД и приложения
    - Пользователи
* Схема
  + - Процедуры и функции
    - Таблицы и представления
    - Коллекция схем XML
* Login – уровень сервера
* User – уровень базы данных
* 
* Роли сервера
  + имеют предопределенный набор разрешений на уровне сервера
  + предназначены для администрирования
  + им могут назначаться имена входа без наличия учетной записи пользователя в базе данных
* Роли базы данных
  + имеют предопределенный набор разрешений для управления группами разрешений
* Bulkadmin – копирование БД
* Dbcreator – создание БД
* Diskadmin – упр-е дисковыми файлами
* Processadmin – управление процессами
* Securityadmin – безопасность
* Serveradmin  – конфигурация сервера
* Setupadmin – процедуры запуска
* Sysadmin – любые действия
* public
* db\_accessadmin
* db\_backupoperator
* db\_datareader
* db\_datawriter
* db\_denydatareader
* db\_denydatawriter
* db\_ddladmin
* db\_owner
* Имперсонификация – разрешение заимствовать права у другого пользователя для выполнения определенного блока кода
* Цепочка владения – Ownership Chain
* Цепочка владения – это последовательность объектов базы данных, которые обращаются друг к другу
* SQL Server пропускает проверку привилегий для улучшения производительности для двух объектов с одинаковым владельцем

ИТОГО

* Настройка физического доступа к серверу
* Разрешение устанавливать соединение с экземпляром
* Разрешение на доступ к конкретной базе данных
* Разрешение на доступ к объектам базы данных

1. Обеспечение безопасности данных приложения в моделях с использованием удаленного сервера баз данных (шифрование, хеширование, ЭЦП, …).

Шифрование SQL Server

* Service Master Key – шифрует мастер-ключи базы данных
* Database Master Key – шифрует симметричные ключи
* Симметричные ключи – шифруют данные

Service Master Key

* Производительность - симметричные ключи
* Возможны другие иерархии шифрования с дополнительными уровнями
* Модуль расширенного управления ключами хранит ключи вне SQL Server.
* Главный ключ службы и все главные ключи базы данных являются симметричными ключами
* Service Master Key – мастер-ключ службы
* Автоматически создается при создании экземпляра
* ALTER, RESTORE, BACKUP
* Обязательно создавать резервную копию!

Database Master Key

* Database Master Key – мастер-ключ базы данных
* MASTER KEY
  + CREATE, ALTER, DROP
  + RESTORE, BACKUP
  + OPEN, CLOSE
* Обязательно создавать резервную копию!

СИММЕТРИЧНЫЕ КЛЮЧИ

* SYMMETRIC KEY
  + CREATE, ALTER, DROP
  + OPEN, CLOSE
  + CLOSE ALL SYMMETRIC KEYS
  + Пара ключей – public и private
  + Private – длина 512, 1024, 2048 бит
  + Алгоритм – RSA
  + CREATE, ALTER, DROP

Сертификат — предложение с цифровой подписью, которое привязывает значение открытого ключа к определенному лицу, владеющему соответствующим открытым ключом

* + Сертификаты подписываются центром сертификации
  + CREATE, ALTER, DROP CERTIFICATE
  + 
* Сертификаты содержат следующую информацию:
  + значение открытого ключа субъекта
  + информацию, идентифицирующую субъекта
  + информацию, идентифицирующую издателя сертификата
  + цифровую подпись издателя сертификата

Прозрачное шифрование

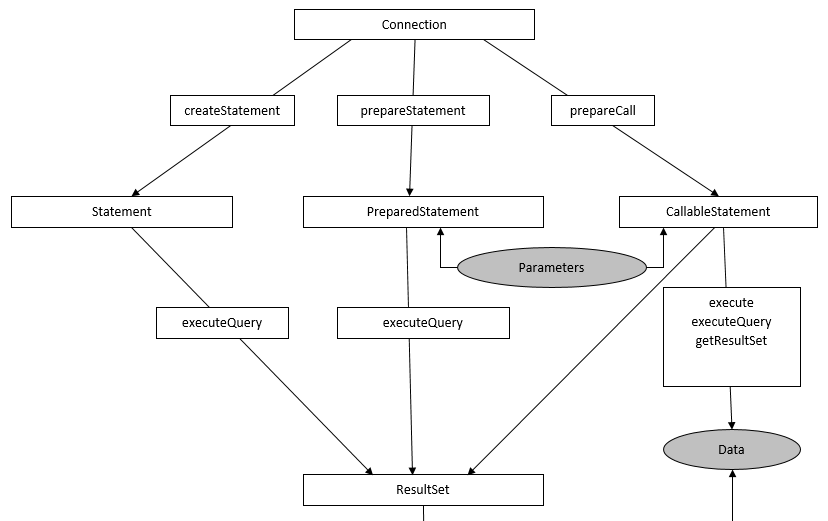
* TDE – transparent data encryption – прозрачное шифрование данных
* Шифрует каждую страницу
* Расшифровывает при каждом запросе
* Обнуление и шифрование журнала транзакций
* Шифрование tempdb

1. Программные интерфейсы с СУБД. Применение интерфейса JDBC для работы с удаленными серверами СУБД в ОС Android.

* Программный интерфейс с БД – набор классов (или набор функций), обеспечивающий:
* соединение с СУБД
* отправку SQL-запросов
* доставку результатов выполнения запросов
* диагностику ошибок



* Встроенные
* Стандартные (универсальные)
* Встроенный программный интерфейс – программный интерфейс ориентирован на конкретную СУБД
* Как правило используется для локального однопользовательского соединения
* Типичный пример – SQLite
* 
* Стандартный программный интерфейс – не зависит от СУБД
* Как правило используется для соединения с СУБД через сеть (TCP/IP, Named Pipe, …)
* Типичный пример – ODBC, OLEDB, BDE (Borland Database Engine) ADO, ADO.NET, JDBC
* 
* Стандартный программный интерфейс имеет модель провайдера:
* стандартный набор методов
* драйверы (реализации этих методов) СУБД
* Должна быть указана строка подключения:
* местоположение сервера (ip-адрес, порт),
* имя/пароль,
* имя базы данных (службы),
* специфичная для СУБД информация (обычно связанная с режимом доступа)
* JDBC – Java Database Connectivity
* платформенно-независимый стандарт взаимодействия Java-приложений с СУБД
* входит в состав Java SE
* Имеет модель провайдера: стандартный набор классов и драйверы
* Для подключения к отдельным базам данных JDBC требует наличия драйверов для каждой базы данных
* Добавить модуль в проект



Выполнение запросов к БД

* Statement можно использовать для выполнения любых запросов, так и для DDL, DML
* Метод executeQuery класса Statement отправляет переданный ему запрос в базу данных и возвращает результат в виде класса ResultSet
* Объект ResultSet - результат выполнения запроса
* Метод next() класса ResultSet возвращает true, если еще есть данные в стеке ответа
* После выполнения next() получаем очередную запись
* Объекты Connection, Statement и ResultSet после использования необходимо закрывать методом close()
* PreparedStatement – скомпилированная версия SQL-выражения
* Предварительно подготавливаются SQL запросы с параметрами
* Метод set\_\_\_\_\_() устанавливает значения параметров

1. setInt()
2. setString()
3. …

* Столбцы нумеруются с 1
* Пакетная (batch) отправка SQL запросов
* Работает значительно быстрее

1. Понятие репликации. Организация репликации в SQL Server

Репликация – механизм синхронизации нескольких копий объекта

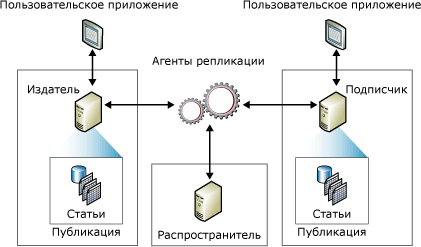
* 1. Позволяет распределять данные из одной базы данных в одну или несколько других
  2. Может выполняться либо непрерывно, либо по расписанию
  3. Используется, если нужно периодически управлять данными на нескольких серверах

Когда репликация не требуется

* Однократно создать копию БД – копирование БД
* Скопировать данные с одного сервера на другой с преобразованием – процедура импорта-экспорта

Когда репликация требуется

* Синхронизация изменений удаленных баз данных с центральной БД
* Создание нескольких экземпляров БД, позволяющих распределить рабочую нагрузку
* Перемещение определенных наборов данных с центрального сервера распределение их на другие
* Преобразование данных и распределение их среди подписчиков



Компоненты репликации

* Дистрибьютор (distibutor)-

Серверы, распространяющие реплицируемые данные

На каждом дистрибьюторе хранится БД распространения, метаданные, исторические данные и транзакции.

* Издатель (publisher)-

-Издатели - серверы, предоставляющие данные для репликации на другие серверы

-Отслеживают изменения в данных и поддерживают другую информацию об исходных БД

-Каждая группа данных имеет только одного издателя

* Подписчик (subscriber)-

-Серверы назначения, которые хранят реплицированные данные и получают обновления

-Подписчики также могут вносить изменения в данные

-Допустимо публиковать данные для нескольких подписчиков

* Статья (article) и
* Публикация (publication)-
* Данные, публикуемые для репликации, организовываются в **статьи** и **публикации**
* Статьи являются базовыми единицами, публикации — это совокупность статей, предназначенных для подписчиков.

СТАТЬИ:

таблица

* только определенные столбцы из таблицы, получаемые наложением вертикального фильтра
* только определенные строки из таблицы, получаемые наложением горизонтального фильтра
* подмножество данных таблицы, состоящее из определенных строк и столбцов
* представление, индексированное представление или пользовательскую функция
* хранимая процедура

Виды репликаций

* Репликация моментальных снимков (Snapshot Replication):

1. Создает моментальный снимок текущих данных
2. Гарантирует согласованность данных между издателем и подписчиком
3. Можно обрабатывать несколько статей одновременно (параллельная обработка)
4. Автоматическое возобновление прерванной доставки моментальных снимков
5. Увеличивает накладные расходы и загрузку сети при работе с большими БД
6. Подписчики не имеют самой свежей информации

* Репликация сведением (слиянием, объединением) (Merge Replication)

1. Позволяет подписчикам вносить изменения в реплицированные данные независимо друг от друга
2. Применяется в средах с топологией сервер-клиент
3. Использует механизм разрешения конфликтов для определения изменений
4. Обрабатывает изменения построчно

* Транзакционная репликация (репликация транзакций) (Transact Replication):

1. В начале подписчикам посылается моментальный снимок данных
2. Выбранные транзакции в журнале транзакций издателя отмечаются для репликации и направляются каждому подписчику в отдельности
3. Производится репликация отдельных транзакций, а не всего набора данных
4. Можно инициализировать подписку из резервной копии
5. Можно производить непрерывно или периодически

* Одноранговая
* Двунаправленная
* Для не SQL Server БД

Планирование репликации

* Выбор топологии репликации
* Выбор типа репликации
* Выполнение необходимых подготовительных задач
* Настройка дистрибьютора
* Создание баз данных публикаций
* Настройка издателя
* Создание публикации
* Создание подписок на публикацию и назначение подписчиков

Агенты и задания репликации

* Snapshot Agent (snapshot.exe) - Создает моментальные снимки, включающие структуру данных и сами данные
* Distribution Agent (distrib.exe) - Применяет к подписчикам данные репликации моментальных снимков или репликации транзакций
* Merge Agent (replmerg.exe) - Синхронизирует изменения, сделанные на подписчике после копирования с издателя первоначального моментального снимка
* Log Reader Agent (logread.exe) - Перемещает транзакции, отмеченные для репликации, из журнала транзакций на издателе в базу дистрибьютора
* Queue Reader Agent (qrdrsvc.exe)- Сохраняет изменения БД в очередь, при помощи которой обновления могут быть распространены издателю асинхронно

Агенты и задания репликации - очистка

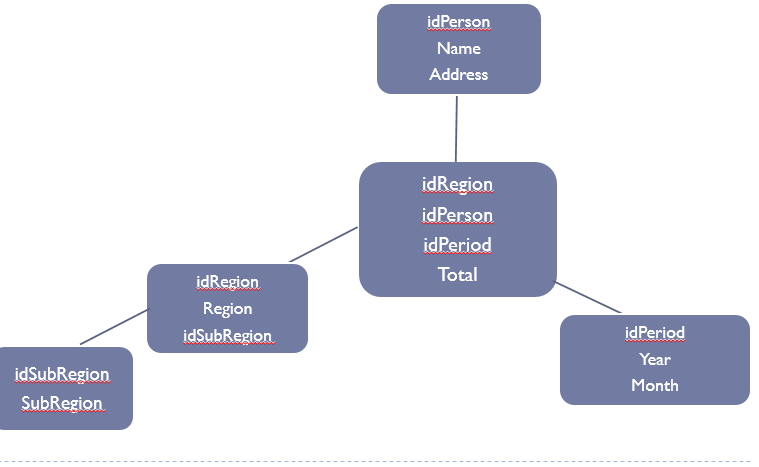
* **Reinitialize Subscriptions Having Data Validation Failures** – отмечает все ошибочные подписки, по умолчанию выключено
* **Replication Agents Checkup** – проверка агентов репликации, по умолчанию запускается каждые 10 минут.
* **Replication Monitoring Refresher For** – обновляет кэшированные запросы, используемые Replication Monitor, по умолчанию запускается автоматически при запуске SQL Server Agent и работает непрерывно

1. Основы BI. Понятия ETL, куба, измерения, меры и пр. Работа с кубами в SQL Server.

* Бизнес-аналитика – business intelligence – BI – область технологии баз данных
* OLTP – online transaction processing – оперативная обработка транзакций
* Системы бизнес-аналитики
* Процесс интегрирования всех необходимых данных в один источник данных, к которому конечные пользователи могут осуществлять нерегламентированные запросы для анализа этих данных
* Цель: принятие лучших бизнес решений
* данные собираются из различных источников
* данные приводятся в согласованное состояние
* данные регулярно загружаются в базу данных
  + хранилище данных – data warehouse
  + витрина данных – data mart
* основной режим – только чтение
* количество одновременных пользователей небольшое
* создание отчетов по различным аспектам
* выполнение сложных запросов для сравнения данных
* Хранилище данных – база данных всех данных организации, к которым пользователи могут иметь единообразный доступ
* Проблемы:
  + большой объем данных
  + получены в разное время
  + хранятся в разных базах данных
* Решение: Консолидация данных
* Консолидация данных – все одинаковые запросы, выполняемые к хранилищу данных в разное время, должны возвращать одинаковые результаты
* Процесс консолидации – ETL:
  + сбор данных из различных источников
  + очистка данных
  + обеспечение качества данных
* Витрина данных – база данных всех данных на уровне подразделения организации, к которым пользователи могут иметь единообразный доступ

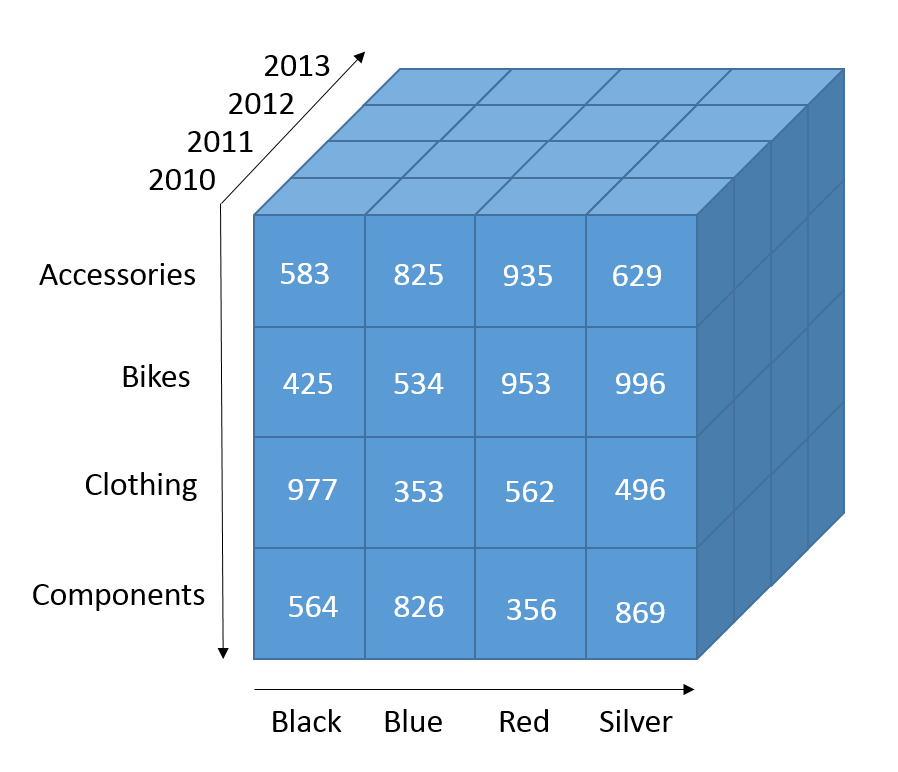
Проектирование хранилищ данных

* Для проектирования хранилищ и витрин данных применяется размерная - или пространственная модель
* Центральная таблица – таблица фактов – fact table
* Таблицы измерений – dimension table
* Числовые столбцы таблицы фактов, кроме первичного ключа, называются мерами – measure
* Имена столбцов таблиц измерений используются в качестве заголовков в отчетах



ЗВЕЗДА СНЕЖИНКА

* Куб — это подмножество данных из хранилища данных, которое можно организовать в многомерную структуру



Виды архитектуры

* реляционная OLAP – ROLAP – relational online analytical processing – предварительно вычисленные данные не сохраняются
* многомерная OLAP – MOLAP – multidimensional online analytical processing – данные на уровне листьев и их агрегированные данные сохраняются в многомерном кубе
* гибридная OLAP – HOLAP – hybrid online analytical processing – агрегированные данные сохраняются, данные на уровне листьев не сохраняются

Создание и обработка многомерного куба

* Создание проекта бизнес-аналитики
* Определение источников данных
* Создание представлений источников данных
* Создание куба
* Агрегирование
* Обработка куба
* Просмотр куба
* MDX – язык запросов для извлечения данных из многомерных БД
* Разработан в Microsoft 1998
* Является частью спецификации OLEDB для OLAP
* SELECT задает результирующий набор, который содержит подмножество многомерных данных, выбранных из куба
* Необходимо указать оси (подмножества измерений) - ON axis(…):
* ON COLUMNS
* ON ROWS
* ON PAGES
* Набор – совокупность кортежей, определенных с помощью одних и тех же измерений