# Проектирование, использование и оптимизация баз данных

***База данных*** – организованная определенным образом структура для хранения данных.

Классификация баз данных:

* Реляционные базы данных – в основе лежит реляционная модель, базирующаяся на связях и отношениях таблиц
* Нереляционные базы данных (NoSQL) – оперирует целыми документами

***Схема БД*** - описанная на формальном языке коллекция объектов, формирующая структуру базы данных.

Объекты схемы базы данных:

* Таблицы (table)
* Поля (field)
* Отношения (relation)
* Представления (view)
* Хранимые процедуры (stored procedure)
* Функции (function)
* Триггеры (trigger)
* Индексы (index)

SQL – формальный язык для написания запросов к базам данных.

Подмножества языка:

* DML (Data Manipulation Language) – язык манипулирования данными.

Операторы – SELECT, INSERT, UPDATE, DELETE

* DDL (Data Definition Language) – язык объявления данных.

Операторы – CREATE, ALTER, DROP

* DCL (Data Control Language) – языка управления данными.

Операторы – GRANT, REVOKE

* TCL (Transaction Control Language) – язык управления транзакциями

Операторы – BEGIN TRAN, COMMIT TRAN, ROLLBACK

*Таблица* – структурированный набор связных данных, хранящихся в БД. С точки зрения теории множеств, таблица является мультимножеством (multiset).

Записи в таблице называют строками (row, record). Структура каждой записи табоицы описывается одинаковым набором столбцов (column).

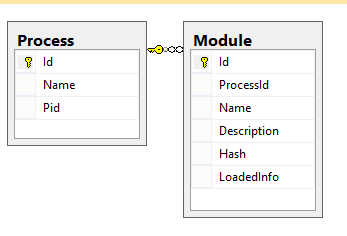
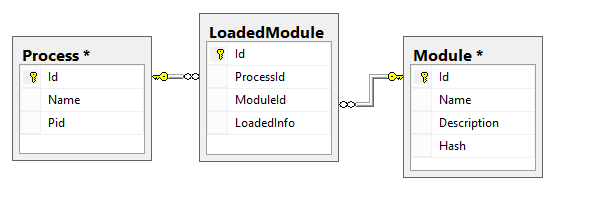
Столбцы являются способом обеспечения целостности данных. Каждый столбец имеет тип, который ограничивает область его допустимых значений. Также целостность данных также обеспечивается с помощью ограничений (constraints). К их числу относятся первичные и внешние ключи, значения по умолчанию (default), ограничения уникальности (unique).

В общем случае, каждая таблица должна иметь первичный ключ (primary key) – специальный столбец (или несколько), которая позволяет уникально идентифицировать строку в таблице.

Связи между таблицами описываются с помощью внешних ключей (foreign key). Внешний ключ устанавливает ссылочную связь между двумя таблицами, когда колонка одной таблицы ссылается на первичный ключ другой таблицы. Значения таких столбцов ограничены данными, хранящимися в соответствующей таблице, на которую они ссылаются.

# Нормализация

* Приведение схемы БД к нормальной форме.
* Устранение избыточности данных.
* Рост количества таблиц пропорционально количеству сущностей.
* Запросы на выборку становятся дольше за счет соединений.



Представление (view) – объект, который представляет SELECT запрос в виде виртуальной таблицы. Позволяет манипулировать данными, полученными с помощью запроса из разных таблиц в виде единой сущности.

Хранимая процедура (stored procedure) – набор команд SQL, которые единожды компилируются и хранятся на сервере.

Функции – в отличие от хранимых процедур, функции возвращают значение определенного типа. Функции можно вызывать в теле DML запросов, а табличные функции даже можно джойнить. В связи с этим, на них накладываются некоторые ограничения.

Триггер (trigger) – хранимая процедура особого типа. Она не вызывается явно, ее вызов происходит автоматически при выполнении некоторого события. Например, при добавлении/обновлении/удалении записей в таблице.

# Транзакции

ACID (*Atomicity, Consistency, Isolation, Durability*)

Атомарность - гарантирует, что никакая транзакция не будет зафиксирована в системе частично. Будут либо выполнены все её подоперации, либо не выполнено ни одной. Если транзакцию не удаётся полностью завершить, результаты всех её до сих пор произведённых действий будут отменены и система вернётся в исходное .

Согласованность – транзакция, достигающая своего нормального завершения и, тем самым, фиксирующая свои результаты, сохраняет согласованность базы данных. Каждая успешная транзакция по определению фиксирует только допустимые результаты. Согласованность на уровне транзакции означает, что все строки, затронутые любым из SQL-операторов, защищены от изменений на протяжении всей транзакции.

Изолированность – во время выполнения транзакции параллельные транзакции не должны оказывать влияние на её результат. Изолированность — требование дорогое, поэтому в реальных БД существуют режимы, не полностью изолирующие транзакцию.

Долговечность – Независимо от проблем на нижних уровнях (к примеру, сбой в оборудовании) изменения, сделанные успешно завершённой транзакцией, должны остаться сохранёнными после возвращения системы в работу. Если пользователь получил подтверждение от системы, что транзакция выполнена, он может быть уверен, что сделанные им изменения не будут отменены из-за какого-либо сбоя.

# Оптимизация запросов

Для оценки того, какие запросы наиболее сильно проседают по производительности, можно использовать профайлер. Он позволяет собрать некоторые метрики относительно выполняемых запросов, времени и частоты их выполнения.

Для анализа причин низкой производительности в отдельно взятом запросе используется план выполнения запроса (execution plan), который позволяет увидеть информацию о низкоуровневых операциях, которые будет производить система для выполнения запроса и относительные оценки сложности и затратности разных отдельных этапов. В MS SQL Server встроенный оптимизатор на основе плана выполнения даже дает советы по улучшению производительности.

# Индексы

Основной инструмент для улучшения производительности – индексы.

Индекс – вспомогательная древовидная структура, созданная для повышения производительности запросов на выборку в БД.

Индексы бывают кластерные и некластерные.

Кластерные упорядочивают данные физически, страницы со строками являются листьями дерева. Поэтому кластерный индекс может быть всего один на таблицу. По-умолчанию, создается при объявлении первичного ключа

Некластерные индексы не влияют на хранение данных, поэтому их может быть много.

# Домашнее задание:

Написать SQL скрипт, который создаст базу данных с иерархией таблиц (нормализированную) для такой сущности:

Человек: имя, фамилия, дата рождения, номера телефонов, домашний адрес, адрес работы, название фирмы, в которой работает.

Создать одну хранимую процедуру, которая принимает параметры – все атрибуты Человека, и инсертит записи в соответствующие таблицы.

Создать одну хранимую процедуру, которая принимает параметры – все атрибуты Человека, и апдейтит записи в соответствующих таблицах.

Скрипт должен быть написан так, чтоб его можно было выполнить несколько раз без ошибок и дублирования (идемпотентность).