**Базы данных**

Dbeaver- среда для работы с базами данными(IDE).

**База данных-** комплекс, набор таблиц. Но таблица не является БД. База данных- может состоять из 1 таблицы. Relation- это отношение. Реляции есть между таблицами 1 базы данных. Таблицы могут ссылаться друг на друга. Через параметр cascad- можно удалить все связи с других таблиц. Реляция нужна, что бы не создавать мусорных данных, не возможно было в связанных таблицах, увязать то чего нет. Это организованная структура, предназначенная для хранения, изменения, обработки взаимосвязанной информации. БД- это объект, набор данных. Для работы с БД, нужна СУБД.

Система управления базами данных (СУБД) — DBMS - Database management system - совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Виды СУБД- MySQL, MSSQL, Oracle, PostgreSQL, MongoDB.

Иерархическая модель. Модель представляет данные в виде иерархии. Модель ориентирована на описание объектов, находящихся между собой в неких отношениях. Например, структура кадров некоторой организации. Есть 1 корневой узел и от него идут разветвления к другим. Структура данных строится как у дерева.

Сетевая модель. Сетевая модель представляет собой развитие иерархической. Модель позволяет описывать более сложные виды взаимоотношений между данными. Однако расширение возможностей достигается за счет большей сложности реализации самой модели и трудности манипулирования данными.

Реляционная модель. В реляционной модели данные представляются в виде таблиц, состоящих из строк и столбцов. Каждая строка таблицы – информация об одном конкретном объекте, столбцы содержат свойства этого объекта. Взаимоотношения между объектами задаются с помощью связей между столбцами таблиц. Реляционная модель на сегодняшний день наиболее распространена. Она достаточно универсальна и проста в проектировании. Строки таблиц называют записями или кортежами, столбцы – полями или атрибутами. Для того чтобы можно было сослаться на отдельную запись (строку) в некоторой таблице, каждая запись этой таблицы должна содержать уникальный идентификатор. Поле таблицы, значения которого гарантированно уникальны для каждой записи этой таблицы, называют ключевым полем или ключом. Ключ не обязательно должен быть числовым. Иногда уникальным идентификатором может служить не одно поле, а комбинация полей. При этом сочетание значений этих полей должно быть уникальным. Такие поля образуют составной ключ таблицы.

Объектная модель. В этой модели данные представляются в форме объектов. Объект имеет набор свойств, называемых атрибутами, и может включать в себя также процедуры для обработки данных, которые называют методами. Объекты, имеющие одинаковые наборы атрибутов и различающиеся только их значениями, образуют некоторый класс объектов. Например, класс «клиент» может иметь следующие атрибуты: «фамилия», «имя», «отчество», «номер кредитной карты». Для каждого объекта из этого класса определены конкретные значения перечисленных атрибутов. Говорят, что объект является экземпляром класса. На основе существующего класса могут создаваться новые, наследующие свойства исходного. При этом исходный класс именуется родителем нового класса. Производный класс называют потомком исходного. При этом объекты – экземпляры класса-потомка принадлежат также и родительскому классу, поскольку обладают всеми его атрибутами. Пример: на основе класса «клиент» может быть определен класс «постоянный клиент»

Гибридные модели. В некоторых приложениях предпринимаются попытки смешения различных моделей представления данных. Пример такого смешения – объектно-реляционная модель. В ней использовано некоторое сходство между реляционной и объектной идеологией. Строки таблиц реляционной модели соответствуют объектам объектной модели, столбцы таблиц – атрибутам объектов. Таблицы в целом являются аналогом классов. Отсюда вытекает возможность введения наследования при определении таблиц – таблица-потомок содержит те же столбцы, что и родительская, и, кроме того – дополнительные, определенные при наследовании. По идее создателей, объектно-реляционная модель должна унаследовать от реляционной легкость описания и манипулирования данными, а от объектной – возможность определения более сложных взаимоотношений между объектами.

Не релиацинноая бд- как правило хранится в оперативной памяти. Их чаще используют для скорости доступа к данным. Данные отсортированные.

Primary key- идентификатор. Он укникальный. Вторичные ключи(foreign key)- если есть 2 таблицы в которые есть связанные данные. То эти таблицы можно связать с помощью вторичного ключа.

Types of sql commands

DDL(data definition lang.): create, alter, drop, rename, truncate, comment

DQL(data query language): select

DML(data manipulation lang.): insert, update, delete, merge, call, explain plan, lock table.

DCL(data control lang): grant, revoke

Кортеж — конечное множество взаимосвязанных допустимых значений атрибутов, которые вместе описывают некоторую сущность (строка таблицы).

Отношение — конечное множество кортежей (таблица).

Проекция — отношение, полученное из заданного путем удаления и (или) перестановки некоторых атрибутов.

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Метод нормальных форм (НФ) состоит в сборе информации о объектах решения задачи в рамках одного отношения и последующей декомпозиции этого отношения на несколько взаимосвязанных отношений на основе процедур нормализации отношений. Цель нормализации: исключить избыточное дублирование данных, которое является причиной аномалий, возникших при добавлении, редактировании и удалении кортежей(строк таблицы).

Что такое SQL

structured query language — «язык структурированных запросов» — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

"NoSQL" имеет абсолютно стихийное происхождение и не имеет общепризнанного определения или научного учреждения за спиной. Это название скорее характеризует вектор развития ИТ в сторону от реляционных баз данных. Расшифровывается как Not Only SQL. Базы данных NoSQL специально созданы для определенных моделей данных и обладают гибкими схемами, что позволяет разрабатывать современные приложения. Базы данных NoSQL получили широкое распространение в связи с простотой разработки, функциональностью и производительностью при любых масштабах.

Что такое нормальные формы

Нормализация- процесс приведения таблицы к нормальному виду. Исключить избыточное дублирование данных, которые являются причиной аномалий, возникших при добавлении, редактировании, удалении картежей( строк таблицы). Есть 2 формы нормальных 1nf- требует что бы каждый элемент таблицы имел только 1 значение( тоесть был атомарным) 2nf- требует, что бы таблица обязательно находилась в 1 нормальной форме и все поля должны зависеть от первичного ключа.

Есть процесс денормализации. Можно осуществить путём скоращения таблиц.

Отношения в бд. Есть 3 вида.

Один к одному.

Связть один ко многим

Связь многие ко многим

1к1- каждая запись в одной таблице соответствует только одной записи в другой таблице. Неоходимо объеденить 2 таблицы в одну.

Отношение один ко многим- означает что 1 или несколько записей в 1 таблице, соответствует одна или несколько записей в другой таблице. Реализуется тогдакогда объетку А может принадлежать или же соответствовать несколько объектов Б. В этом типе связей несколько строк из дочерней таблицы зависит от одной строки в родительской таблице. Например в 1 блоге может быть несколько статей. В этом случае таблица блогов является родительской, а таблицы статей- дочерней.

Связь многие ко многим- реализуется в том случае, когда несколько объектам из таблицы А. может соответсвовать несколько объектов из таблицы Б и в тоже время наскольким объектам из таблицы Б соотвествует несколько объектов из таблицы А.

Нормальные формы:

* Первая нормальная форма: Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.
* Вторая нормальная форма: Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК). Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.
* Третья нормальная форма: Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.
* Четвертая нормальная форма: Отношение находится в 4НФ, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей. В отношении R (A, B, C) существует многозначная зависимость R.A -> -> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от С.
* Пятая нормальная форма: Отношения находятся в 5НФ, если оно находится в 4НФ и отсутствуют сложные зависимые соединения между атрибутами. Если «Атрибут\_1» зависит от «Атрибута\_2», а «Атрибут\_2» в свою очередь зависит от «Атрибута\_3», а «Атрибут\_3» зависит от «Атрибута\_1», то все три атрибута обязательно входят в один кортеж. Это очень жесткое требование, которое можно выполнить лишь при дополнительных условиях. На практике трудно найти пример реализации этого требования в чистом виде.
* Шестая нормальная форма: Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ. Идея «декомпозиции до конца» выдвигалась до начала исследований в области хронологических данных, но не нашла поддержки. Однако для хронологических баз данных максимально возможная декомпозиция позволяет бороться с избыточностью и упрощает поддержание целостности базы данных.

Понятие триггера?

Три́ггер (англ. trigger) — хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Что такое индексы? (Indexes)

Индекс - объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путем последовательного просмотра таблицы строка за строкой может занимать много времени. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы и, таким образом, позволяет искать строки, удовлетворяющие критерию поиска. Ускорение работы с использованием индексов достигается в первую очередь за счет того, что индекс имеет структуру, оптимизированную под поиск — например, сбалансированного дерева. Различные типы индексов:

B-Tree index

Bitmap index

Clustered index

Covering index

Non-unique index

Unique index

Как тестировать загрузку данных при тестировании базы данных?

Исходные данные должны быть известны

Целевые данные должны быть известны

Совместимость источника и цели должна быть проверена

В диспетчере SQL Enterprise запустите пакет DTS после открытия соответствующего пакета DTS.

Вы должны сравнить столбцы цели и источника данных

Количество строк цели и источника должны быть проверены

После обновления данных в источнике проверьте, появляются ли изменения в цели или нет.

Проверьте на NULL и ненужные символы

**Основные команды SQL?**

* Просмотр доступных баз данных

SHOW DATABASES;

* Создание новой базы данных

CREATE DATABASE;

* Выбор базы данных для использования

USE ;

* Импорт SQL-команд из файла .sql

SOURCE ;

* Удаление базы данных

DROP DATABASE ;

* Просмотр таблиц, доступных в базе данных

SHOW TABLES;

* Создание новой таблицы

CREATE TABLE ( , , PRIMARY KEY (), FOREIGN KEY () REFERENCES () );

* Добавление данных в таблицу

INSERT INTO (, , , …) VALUES (, , , …);  При добавлении данных в каждый столбец таблицы не требуется указывать названия столбцов. INSERT INTO VALUES (, , , …);

* Обновление данных таблицы

UPDATE SET = , = , ... WHERE ;

* Удаление всех данных из таблицы

DELETE FROM ;

* Удаление таблицы

DROP TABLE ;  SELECT используется для получения данных из определенной таблицы: SELECT , , … FROM ;

* Следующей командой можно вывести все данные из таблицы:

SELECT \* FROM ;  SELECT DISTINCT

* В столбцах таблицы могут содержаться повторяющиеся данные. Используйте SELECT DISTINCT для получения только неповторяющихся данных.

SELECT DISTINCT , , … FROM ;  WHERE

* Можно использовать ключевое слово WHERE в SELECT для указания условий в запросе:

SELECT , , … FROM WHERE ;

* В запросе можно задавать следующие условия:

сравнение текста; сравнение численных значений; логические операции AND (и), OR (или) и NOT (отрицание). Пример Попробуйте выполнить следующие команды. Обратите внимание на условия, заданные в WHERE: SELECT \* FROM course WHERE dept\_name=’Comp. Sci.’; SELECT \* FROM course WHERE credits>3; SELECT \* FROM course WHERE dept\_name='Comp. Sci.' AND credits>3;

* Оператор GROUP BY часто используется с агрегатными функциями, такими как COUNT, MAX, MIN, SUM и AVG, для группировки выходных значений.

SELECT , , … FROM GROUP BY ;  Пример Выведем количество курсов для каждого факультета: SELECT COUNT(course\_id), dept\_name FROM course GROUP BY dept\_name;

* Ключевое слово HAVING было добавлено в SQL потому, что WHERE не может быть использовано для работы с агрегатными функциями.

SELECT , , ... FROM GROUP BY HAVING   Пример Выведем список факультетов, у которых более одного курса: SELECT COUNT(course\_id), dept\_name FROM course GROUP BY dept\_name HAVING COUNT(course\_id)>1;

* ORDER BY используется для сортировки результатов запроса по убыванию или возрастанию. ORDERBY отсортирует по возрастанию, если не будет указан способ сортировки ASC или DESC.

SELECT , , … FROM ORDER BY , , … ASC|DESC;  Пример Выведем список курсов по возрастанию и убыванию количества кредитов: SELECT \* FROM course ORDER BY credits; SELECT \* FROM course ORDER BY credits DESC;

* BETWEEN используется для выбора значений данных из определенного промежутка. Могут быть использованы числовые и текстовые значения, а также даты.

SELECT , , … FROM WHERE BETWEEN AND ;  Пример Выведем список инструкторов, чья зарплата больше 50 000, но меньше 100 000: SELECT \* FROM instructor WHERE salary BETWEEN 50000 AND 100000;

* Оператор LIKE используется в WHERE, чтобы задать шаблон поиска похожего значения.

Есть два свободных оператора, которые используются в LIKE: % (ни одного, один или несколько символов); \_ (один символ). SELECT , , … FROM WHERE LIKE ;  Пример Выведем список курсов, в имени которых содержится «to», и список курсов, название которых начинается с «CS-»: SELECT \* FROM course WHERE title LIKE ‘%to%’; SELECT \* FROM course WHERE course\_id LIKE 'CS-\_\_\_';

* С помощью IN можно указать несколько значений для оператора WHERE:

SELECT , , … FROM WHERE IN (, , …);  Пример Выведем список студентов с направлений Comp. Sci., Physics и Elec. Eng.: SELECT \* FROM student WHERE dept\_name IN (‘Comp. Sci.’, ‘Physics’, ‘Elec. Eng.’);

* JOIN используется для связи двух или более таблиц с помощью общих атрибутов внутри них.
* View — это виртуальная таблица SQL, созданная в результате выполнения выражения. Она содержит строки и столбцы и очень похожа на обычную SQL-таблицу. View всегда показывает самую свежую информацию из базы данных.

Создание CREATE VIEW AS SELECT , , … FROM WHERE ;  Удаление DROP VIEW ;  Пример Создадим view, состоящую из курсов с 3 кредитами:

* 24. Агрегатные функции - эти функции используются для получения совокупного результата, относящегося к рассматриваемым данным. Ниже приведены общеупотребительные агрегированные функции:

COUNT (col\_name) — возвращает количество строк; SUM (col\_name) — возвращает сумму значений в данном столбце; AVG (col\_name) — возвращает среднее значение данного столбца; MIN (col\_name) — возвращает наименьшее значение данного столбца; MAX (col\_name) — возвращает наибольшее значение данного столбца.

* Вложенные подзапросы — это SQL-запросы, которые включают выражения SELECT, FROM и WHERE, вложенные в другой запрос.

Пример Найдем курсы, которые преподавались осенью 2009 и весной 2010 годов: SELECT DISTINCT course\_id FROM section WHERE semester = ‘Fall’ AND year= 2009 AND course\_id IN ( SELECT course\_id FROM section WHERE semester = ‘Spring’ AND year= 2010 );

Подробнее о джойнах? (Join)

Как и было сказано выше, различные виды JOIN помогают объединить некие данные из нескольких таблиц каким-либо образом.

Так чем отличается INNER JOIN от LEFT JOIN? Чаще всего ответ примерно такой: "inner join — это как бы пересечение множеств, т.е. остается только то, что есть в обеих таблицах, а left join — это когда левая таблица остается без изменений, а от правой добавляется пересечение множеств. Для всех остальных строк добавляется null". Еще, бывает, рисуют пересекающиеся круги. Это понимание и подобные ответы – по сути не верны, т.к. все джойны – декартово произведение (cross join) с фильтрами (предикатом и, возможно, UNION). Также стоит обратить внимание на порядок таблиц при различных джойнах.

**Типы данных в SQL?**

* Exact Numeric SQL Data Types:
  + bigint = Range from -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807) int = Range from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647) smallint = Range from -2^15 (-32,768) to 2^15-1 (32,767) tinyint = Range from 0 to 255 bit = 0 and 1 decimal = Range from –10^38 +1 to 10^38 -1 numeric = Range from -10^38 +1 to 10^38 -1 money = Range from -922,337,203,685,477.5808 to +922,337,203,685,477.5807 small money = Range from -214,748.3648 to +214,748.3647
* Approximate Numeric SQL Data Types:
  + float = Range from -1.79E + 308 to 1.79E + 308 real = Range from -3.40E + 38 to 3.40E + 38
* Date and Time SQL Data Types:
  + datetime = From Jan 1, 1753 to Dec 31, 9999 smalldatetime = From Jan 1, 1900 to Jun 6, 2079 date = To store a date like March 27, 1986 time = To store a time of day like 12:00 A.M.
* Character Strings SQL Data Types:
  + char = Maximum length of 8,000 characters varchar = Maximum of 8,000 characters varchar(max) = Maximum length of 231 characters text = Maximum length of 2,147,483,647 characters.
* Unicode Character Strings SQL Data Types:
  + nchar = Maximum length of 4,000 characters nvarchar = Maximum length of 4,000 characters nvarchar(max) = Maximum length of 231 characters ntext = Maximum length of 1,073,741,823 characters
* Binary SQL Data Types:
  + binary = Maximum length of 8,000 bytes varbinary � = Maximum length of 8,000 bytes varbinary(max) = Maximum length of 231 bytes image = Maximum length of 2,147,483,647 bytes



