Вопрос № 3.12

Типы баз данных. Реляционные БД. Нормальные формы РБД. Язык SQL.

База Данных (БД) — структурированный организованный набор данных, описывающих характеристики какой-либо физической или виртуальной системы.

Система Управления Базами Данных (СУБД). – программное обеспечение, предназначенное для организации и ведения баз данных.

Типы баз данных.

- *Картотеки* - упорядоченное собрание данных, как правило на карточках малого формата и являет собой каталог какой либо базы данных. Каждая карта является информационной единицей и предоставляет сведения о каком либо объекте базы данных, с целью облегчения поиска этого объекта по определённым признакам. Упорядочение осуществляется обязательно по логическим критериям, по алфавиту, дате и т.д.
- *Иерархические* - состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию. Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй — объекты второго уровня и т. д. В этой модели запрос, направленный вниз по иерархии, прост (например: какие заказы принадлежат этому покупателю); однако запрос, направленный вверх по иерархии, более сложен.
- *Сетевые* - к основным понятиям сетевой модели базы данных относятся: уровень, элемент (узел), связь. Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию. Выполнение простых запросов является достаточно сложным процессом, как и в иерархических БД.
- *Реляционные (*relation*)* – базы данных, основанные на реляционной модели. В реляционных базах данных все данные представлены в виде простых таблиц, разбитых на строки и столбцы, на пересечении которых расположены данные. Запросы к таким таблицам возвращают таблицы, которые сами могут становиться предметом дальнейших запросов. Подробнее в следующем разделе.
- *Объектно-ориентированные* – база данных, в которой данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями.
- *Многомерные* – данные БД позволяют реализовать множество различных представлений данных и характеризуются тремя основными чертами: многомерное представление данных; сложные вычисления над данными; вычисления, связанные с изменением данных во времени.

Реляционные БД.

Теория реляционных баз данных была разработана доктором Коддом из компании IBM в 1970 году. В реляционных базах данных все данные представлены в виде простых таблиц, разбитых на строки и столбцы, на пересечении которых расположены данные. Запросы к таким таблицам возвращают таблицы, которые сами могут становиться предметом

дальнейших запросов. Каждая база данных может включать несколько таблиц. Кратко особенности реляционной базы данных можно сформулировать следующим образом:

- Данные хранятся в таблицах, состоящих из столбцов ("атрибутов") и строк ("записей", "кортежей");
- На пересечении каждого столбца и строчки стоит в точности одно значение;
- У каждого столбца есть своё имя, которое служит его названием, и все значения в одном столбце имеют один тип.
- Запросы к базе данных возвращают результат в виде таблиц, которые тоже могут выступать как объект запросов.

Строки в реляционной базе данных неупорядочены - упорядочивание производится в момент формирования ответа на запрос.
Общепринятым стандартом языка работы с реляционными базами данных является язык SQL.

Нормальные формы РБД.

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

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется *нормализацией*. Нормализация позволяет обезопасить базу данных от логических и структурных проблем, называемых аномалиями данных. К примеру, когда существует несколько одинаковых записей в таблице, существует риск нарушения целостности данных при обновлении таблицы. Таблица, прошедшая нормализацию, менее подвержена таким проблемам, т.к. ее структура предполагает определение связей между данными, что исключает необходимость в существовании записей с повторяющейся информацией.
Основное назначение нормальных форм — приведение структуры базы данных к виду, обеспечивающему минимальную избыточность. Устранение избыточности производится за счёт декомпозиции отношений (таблиц) таким образом, чтобы свести к минимуму функциональные зависимости между их атрибутами (полями). Понятие функциональной зависимости здесь определяется следующим образом:
В паре атрибутов одного отношения, X и Y, атрибут Y функционально зависит от атрибута X, если в данном отношении одному значению X соответствует в точности одно значение Y.
Каждая нормальная форма представляет собой определённое условие, которому должна соответствовать таблица базы данных. Если таблица не соответствует нормальной форме, она может быть приведена к ней (нормализована) за счёт декомпозиции, то есть разбиения на несколько таблиц, связанных между собой.

Типы нормальных форм

Нормализация может применяться к таблице, первоначально отвечающей следующим требованиям:

- Таблица содержит нуль или более записей.
- Все записи таблицы имеют одно и то же множество полей, причём одноимённые поля относятся к одинаковым типам данных.
- Таблица не может содержать двух полностью идентичных записей.

Первая нормальная форма (1NF)

Таблица находится в первой нормальной форме, если каждый её атрибут атомарен и все строки различны. Под выражением «атрибут атомарен» понимается, что атрибут может содержать только одно значение. Таким образом, не соответствуют 1NF таблицы, в полях которых могут храниться списки значений. Для приведения таблицы к 1NF обычно требуется разбить таблицу на несколько отдельных таблиц.

Пример приведения таблицы к первой нормальной форме.

Исходная, ненормализованная, таблица:

Сотрудник	**Номер телефона**
	283-56-82
Иванов И. И.	390-57-34
Петров П. Ю.	708-62-34

Таблица, приведённая к 1NF:

Сотрудник	**Номер телефона**
Иванов И. И.	283-56-82
Иванов И. И.	390-57-34
Петров П. Ю.	708-62-34

Атомарность атрибутов

Вопрос об атомарности атрибутов решается на основе семантики данных, то есть их смыслового значения. Атрибут атомарен, если его значение теряет смысл при любом разбиении на части или переупорядочивании. И наоборот, если какой-либо способ разбиения на части не лишает атрибут смысла, то атрибут неатомарен. Одно и то же значение может быть атомарным или неатомарным в зависимости от смысла этого значения. Например, значение «4286» является

- атомарным, если его смысл — «пин-код кредитной карты» (при разбиении на части или переупорядочивании смысл теряется)
- неатомарным, если его смысл — «четные цифры» (при разбиении на части или переупорядочивании смысл не теряется)

Хорошим способом принятия решения о необходимости разбиения атрибута на части является вопрос: «будут ли части атрибута использоваться по отдельности?». Если да, то атрибут следует разделить (но так, чтобы сохранились осмысленные части атрибута). Далее необходимо снова задаться тем же вопросом для новой структуры и так до тех пор, пока не останется атрибутов, допускающих разбиение.

Вторая нормальная форма (2NF)

Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего первичного ключа, но при этом не находится в функциональной зависимости от какой-либо его части.

Пример приведения таблицы к второй нормальной:

Сотрудник	Должность	Зарплата
Гришин	Менеджер	30000
Васильев	Программист	40000
Петров	Программист	40000

В результате приведения к 2NF получим две таблицы:

Сотрудник	Должность
Гришин	Менеджер
Васильев	Программист
Петров	Программист

Должность	Зарплата
Менеджер	30000
Программист	40000

Третья нормальная форма (3NF)

Таблица находится в третьей нормальной форме, если она находится во второй нормальной форме, и при этом любой её неключевой атрибут функционально зависит только от первичного ключа.

При решении практических задач в большинстве случаев третья нормальная форма является достаточной. Процесс проектирования реляционной базы данных, как правило, заканчивается приведением к 3NF.

Пример приведения таблицы к третьей нормальной форме:

id_сотрудника	Фамилия	Отдел	Телефон
1	Гришин	1	11-22-33
2	Васильев	1	11-22-33
3	Петров	2	44-55-66

В результате приведения к 3NF получим две таблицы:

id_сотрудника	Фамилия	Отдел
1	Гришин	1
2	Васильев	1
3	Петров	2

Отдел	Телефон
1	11-22-33
2	44-55-66

Нормальная форма Бойса-Кодда (BCNF)

Таблица находится в BCNF, если она находится в третьей нормальной форме, и при этом отсутствуют функциональные зависимости атрибутов первичного ключа от не-ключевых атрибутов.

Данная нормальная форма — это модификация третьей нормальной формы. Таблица может находиться в 3NF, но не в BCNF, только в одном случае: если она имеет, помимо первичного ключа, ещё по крайней мере один составной возможный ключ, и по крайней

мере один из атрибутов таблицы входит и в первичный, и в возможный ключи. Такое бывает достаточно редко, в остальном 3NF и BCNF эквивалентны.

Четвёртая нормальная форма (4NF)

Таблица находится в 4NF, если она находится в BCNF и не содержит нетривиальных многозначных зависимостей. Многозначная зависимость не является функциональной, она существует в том случае, когда из факта, что в таблице содержится некоторая строка X, следует, что в таблице обязательно существует некоторая определённая строка Y. То есть, таблица находится в 4NF, если все ее многозначные зависимости являются функциональными.

Пятая нормальная форма (5NF)

Таблица находится в 5NF, если она находится в 4NF и любая многозначная зависимость соединения в ней является тривиальной.

Пятая нормальная форма в большей степени является теоретическим исследованием, и практически не применяется при реальном проектировании баз данных. Это связано со сложностью определения самого наличия зависимостей «проекции — соединения», поскольку утверждение о наличии такой зависимости должно быть сделано для всех возможных состояний БД.

Язык SQL.

SQL (англ. Structured Query Language — язык структурированных запросов) — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. SQL является информационно-логическим языком, а не языком программирования.

SQL основывается на реляционной алгебре.

Язык SQL делится на три части:

- операторы определения данных (Data Definition Language, DDL)
- операторы манипуляции данными (Data Manipulation Language, DML)
- операторы определения доступа к данным (Data Control Language, DCL)

Преимущества

- *Независимость от конкретной СУБД.* Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально закладывались на применение по меньшей мере нескольких СУБД (например: система электронного документооборота Documentum может работать как с Oracle Database, так и с Microsoft SQL Server и IBM DB2)
- *Наличие стандартов.* Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка.
- *Декларативность.* С помощью SQL программист описывает только какие данные нужно извлечь или модифицировать. То, каким образом это сделать решает СУБД непосредственно при обработке SQL запроса.

Недостатки

Несоответствие реляционной модели данных.

Создатель реляционной модели данных Эдгар Кодд, Кристофер Дейт и их сторонники указывают на то, что SQL не является истинно реляционным языком. В частности они указывают на следующие проблемы SQL[1]:

- Повторяющиеся строки;
- Неопределённые значения (nulls);
- Явное указание порядка колонок слева направо;
- Колонки без имени и дублирующиеся имена колонок;
- Отсутствие поддержки свойства «=»;
- Использование указателей;
- Высокая избыточность.