

Модуль 4. Алгоритмы и структуры данных

Тема 4.9. Реляционная модель данных и СУБД

2 часа

Оглавление

4.9. Реляционная модель данных и СУБД	2
4.9.1. Реляционная модель данных.....	2
Таблицы: поля и записи	3
Ключевые поля	3
Идеи Кодда на языке таблиц.....	4
4.9.2. Проектирование реляционной БД из нескольких таблиц	4
Нормализация	5
Связи	6
Ссылочная целостность.....	7
Типы связей.....	7
Упражнение 4.9.1	9
4.9.3. Зачем нужны СУБД	9
4.9.4. Классификация СУБД.....	12
Классификация по месту размещения СУБД.....	12
Классификация по способу доступа к БД	12
Минипроект 4.1. Чемпионат России по футболу/хоккею	14
Источники.....	14
Благодарности	14

4.9. Реляционная модель данных и СУБД

С увеличением объёмов информации возникла потребность в её надёжном хранении, быстром поиске, и обработке. Для этих целей применяют механизм по обработке данных: создание базы данных(БД), специальных средств по её обработке. Рассмотрим примеры применения БД:

Пример 1. Когда вы в супермаркете оплачиваете товары на кассе, то кассир через сканер получает штрих-код и информационная система ищет товар с таким штрих-кодом в БД, получает его цену, записывают информацию, что данный товар куплен и одновременно удаляет купленный товар из списка тех, которые есть в наличии. Отдел закупок супермаркета со своей стороны видит сколько товаров в остатке, может проанализировать с какой скоростью товар раскупается и принимает решение сколько товара заказать на следующую неделю.

Пример 2. Для сдачи ЕГЭ все школьники должны зарегистрироваться в общероссийской БД. Для этого школами предоставляется информация об учениках: ФИО, номер удостоверения пенсионного страхования, паспортные данные, регион, школа, выбранные предметы и т.д. После сдачи ЕГЭ в соответствии с определенной технологией комиссии вводят результаты в БД. Затем выпускники и приемные комиссии вузов имеют возможность увидеть эти результаты.

В каком виде можно хранить информацию из приведенных примеров? Каким образом обеспечить различные права на доступ и изменение информации для разных сторон, которые используют одну и ту же информацию? Если у нас хранится информация о миллионах товаров или школьников как обеспечить быстрый поиск нужной информации? Обычные текстовые файлы не могут решить описанные выше задачи по ряду причин:

- любой поиск и изменение данных не эффективны, потому что текстовые файлы требуют последовательного чтения;
- один и тот же объект (товар, ученик) можно описать по разному, поэтому если использовать естественный язык, то информацию будет очень сложно найти и в принципе провести какой-то анализ или обработку;
- из-за того, что текстовый файл изначально не разделен на части невозможно обеспечить доступ множества пользователей к отдельным его частям с разным уровнем доступа: кому-то для чтения, кому-то для изменения (причем одновременно).

4.9.1. Реляционная модель данных

Исторический прорыв в этой области произошел в 1970 году, когда англичанин Эдгар Кодд из компании IBM предложил **реляционную модель данных**. С ее появлением пришли понятия “база данных” (БД) и система управления БД (СУБД).

База данных (БД) – это специальным образом организованная совокупность данных о некоторой предметной области, хранящаяся во внешней памяти компьютера [1].

Понятно, что можно придумать множество различных способов как хранить и обрабатывать данные в БД. Этот способ организации данных в БД определяется моделью данных.

Модель данных применительно к БД - это некоторые логические правила представления информации в памяти вычислительного устройства.

Позже Ченом была предложена более универсальная модель «сущность-связь» и Кодд предложил расширенную реляционную модель RM/V2 (1990). Модель “сущность-связь” очень

легко перекладывается на реляционную модель, которая на данный момент остается самой распространенной и лежит в основе большинства СУБД.

Таблицы: поля и записи

Реляционная модель данных - это представление совокупности данных в виде двумерных таблиц. Рассмотрим на примере понятия, лежащие в основе реляционной модели.

Пример 1

Пусть перед нами стоит задача хранить информацию по сотрудникам некоторой компании. Рассмотрим на примере сущности «Сотрудник» важные понятия.

Для проектирования и иллюстрации модели “сущность-связь” широко используют схемы БД, которые аналогичны диаграмме классов UML для отражения ОО модели.

У каждой сущности есть ряд свойств, которые ее описывают - **атрибуты**. Пусть в нашем примере для каждого сотрудника мы хотим хранить такие поля: «Фамилия», «Имя», «Отчество», «Отдел», «Должность», «Адрес». Тогда сущность примет вид:

Сотрудник
Фамилия
Имя
Отчество
Отдел
Должность
Адрес

В заголовке прямоугольника мы указали наименование сущности, внутри список атрибутов. В реляционной модели этой картинке соответствует таблица “Сотрудник” с столбцами: «Фамилия», «Имя», «Отчество», «Отдел», «Должность», «Оклад».

Если мы заполним таблицу набором значений атрибутов, например, по двум сотрудникам, то в терминах реляционной модели мы получим 2 **записи**. А таблица имеет 6 **полей**.

Фамилия	Имя	Отчество	Отдел	Должность	Адрес
Иванов	Олег	Игоревич	Бухгалтерия	кассир	г. Тверь ул. Мира д.6 кв.2
Соколов	Петр	Петрович	Склад	кладовщик	г. Тверь ул. Весенняя д.9

Реляционная модель тесно связана с реляционной алгеброй, в терминах которой набор значений атрибутов (запись) носит название **кортеж**.

Ключевые поля

Каждый объект любой таблицы БД должен быть уникальным иначе могут быть неприятности связанные с неоднозначностью размещения одного и того же объекта несколько раз. Убирают эту неоднозначность ключевые поля, которые хранят только уникальную информацию об объекте и никогда не повторяются.

Как быть в ситуации, когда в БД у людей полностью совпадают ФИО и дата рождения? Тогда ключом человека может быть серия и номер паспорта, а этого же человека в налоговой службе – ИНН, в пенсионном фонде – номер СНИЛС.

Пример с человеком, описывает важность ключа, а также тот факт, что один и тот же человек может храниться в разных БД, а значит иметь разные свойства в таблицах. Т.е. человек один, а

сущности разные. Ещё примеры ключевых данных: серийный номер изделия, регистрационный номер авто, ну и конечно же номер на денежной банкноте, который хранится в БД казначейства и различает друг от друга одинаковые купюры.

Т.о. у каждой сущности должно быть определено специальное поле - ключ. **Ключ** - это атрибут или группа атрибутов, значения которых уникальны для каждого экземпляра сущности. Ключ позволяет быстро идентифицировать каждый экземпляр сущности.

Для ранее рассмотренного примера 1 ключом может быть поле "Табельный номер" - он уникален для каждого сотрудника и по нему легко найти конкретного сотрудника:

Сотрудник
Табельный номер
Фамилия
Имя
Отчество
Отдел
Должность
Адрес

В схемах БД для выделения ключевые атрибуты отделяются от остальных чертой либо какими-то знаками (ключиками, звездочками и т.д.)

Идеи Кодда на языке таблиц

- каждая таблица описывает одну сущность/отношение
- порядок расположения полей в таблице не имеет значения
- все значения одного поля имеют одинаковый тип данных
- в таблице не может быть двух полностью одинаковых записей
- порядок записей в таблице не определён

При работе с реляционной БД характерно:

- ✓ Количество и состав полей определяет разработчик БД, пользователи же работают с записями таблицы (добавляют, удаляют, редактируют).
- ✓ Любое поле должно иметь уникальное имя.
- ✓ Поля имеют тип (схоже с понятием тип переменной). Как правило, в СУБД можно задать следующие типы полей:
 - строка символов, текст
 - целое число
 - вещественное число
 - денежная сумма
 - дата, время
 - логическое поле (истина или ложь, да или нет)
- ✓ Поля могут быть обязательными для заполнения или нет.
- ✓ Таблица может содержать столько записей, сколько позволяет объем памяти.
- ✓ Для работы с ними используют язык запросов SQL.

Программисты не работают напрямую с файлами БД, для этого используют специальные программы - системы управления базой данных (СУБД), о которых пойдет речь далее.

4.9.2. Проектирование реляционной БД из нескольких таблиц

До сих пор мы рассматривали примеры отдельных таблиц, но очевидно, что весь объем данных предметной области нельзя уместить в одну таблицу. Причем это связано не только с тем соображением, что это будет слишком большая таблица. Вспомним классы в ООП. Здесь в чем-то


похожий подход - данные группируются в соответствии с логикой принадлежности к той или иной сущности. На практике БД - это набор таблиц, которые связаны между собой.

Пример 2

Перед нами стоит задача разработать БД для ведения семейного бюджета, главная задача которой фиксировать доходы и расходы семьи, чтобы затем можно было провести их анализ.


Начнем с того, что БД должна хранить данные о всех расходах и доходах, т.е. какие происходили денежные операции. И для каждой операции еще нужно знать, на что были потрачены деньги или откуда пришли (зарплата, кредит и т.д.).

Рассмотрим таблицу «Операция» со следующей структурой:

Операция	
	ID
	Название
	Дата_операции
	Сумма
	Примечание

ID – это ключевое поле, поэтому на схеме слева от наименования нарисован ключик.

В этой структуре плохо то, что мы храним все детальные операции, но не можем провести их анализ, сгруппировать: например посчитать сколько за месяц мы истратили на питание, сколько на транспорт и т.д. Значит, нам нужно указание из какой категории каждая операция:

Операция	
	ID
	Название
	Категория
	Дата_операции
	Сумма
	Примечание

Заполним таблицу «Операция»:

ID	Название	Категория	Дата операции	Сумма	Примечания
1	Начальный баланс	Остаток	01.03.2015	120000	
2	Хлеб, мясо, овощи, фрукты	Продукты	06.03.2015	-3300	
3	Цветы	Разное	08.03.2015	-1800	Поздравление на 8 марта
4	Билеты в цирк	Развлечения	08.03.2015	-1000	
5	Хлеб, творог, сметана	Продукты	09.03.2015	-210	
6	Зарплата Сергея, февраль	Зарплата	10.03.2015	50000	

Нормализация

Данные этой таблицы страдают избыточностью, из-за совпадений некоторых значений поля «Категория». Причем, из-за того, что это текстовые поля, будет расходоваться много памяти. Более того, если мы захотим изменить наименование категории, то нам придется сделать это во всех записях таблицы! Убрать избыточность позволит отдельная таблица или таблицы. В нашем случае можно создать таблицу, которая хранит сущность «Категория»:

ID	Название	Приход_Расход
1	Остаток	1
2	Продукты	0
3	Разное	0
4	Развлечения	0
5	Зарплата	1
6	Транспорт	0
7	Медицина	0
8	Образование	0
9	Кредит	1
10	Выплата по кредиту	0

В таблицу мы дополнительно включили новое поле, в котором будем фиксировать, какая это категория: доход или расход. Это позволит группировать операции не только по категориям, но и еще более укрупненно: например, все расходы за месяц. Если приход, то ставим в таблице 1, если расход, то – 0.

Связи

Кроме этого мы должны связать таблицу операций с категориями. Для этого нужно выяснить какая из них будет главная, а какая подчинённая. Поскольку таблица «Категория» хранит неповторяющиеся значения, она будет **главной**, а таблица «Операция» будет **подчинённой**.

Как физически связываются таблицы? Для этого подчинённой таблице добавляют ещё одно поле для связи с главной. Т.е. таблице «Операция» нужно добавить поле **ID_категории**.

Затем свяжем – ключевое поле главной таблицы с новым полем подчинённой таблицы. Новое поле подчинённой таблицы называют **внешний ключ** и оно отображает информацию, которая хранится в другой таблице, а именно в ключевом поле главной таблицы. То ключевое поле главной таблицы называется **главным**. Для отображения связи между сущностями на схемах БД используют линии:



Наполнение таблицы «Операция» станет следующим:

ID	Наименование	Дата операции	ID категории	Сумма	Примечания
1	Начальный баланс	01.03.2015	1	120000	
2	Хлеб, мясо, овощи, фрукты	06.03.2015	2	-3300	
3	Цветы	08.03.2015	3	-1800	Поздравление 8 Марта
4	Билеты в цирк	08.03.2015	4	-1000	
5	Хлеб, творог, сметана	09.03.2015	1	-210	
6	Зарплата Сергея, февраль	10.03.2015	5	50000	

Ссылочная целостность

Подумайте, что случится, если в Примере 3 в таблице «Категория» удалить любую запись? Очевидно, что возникнет проблема в таблице «Операция», потому что для операции по заданному коду категории мы не сможем получить нужную информацию. Т.е. сущность «Операции» **зависимая** от сущности «Категории».

Если в таблице есть внешний ключ, то он должен обязательно совпадать с одним из значений первичного ключа в другой таблице. Это называется соблюдением **ссылочной целостности**. Современные СУБД имеют различные средства поддержки ссылочной целостности БД. Например, пользователю не разрешат удалить таблицу, если есть таблицы, которые от нее зависят.

Типы связей

Мы уже разобрались, что при проектировании БД и работы с ней очень важно устанавливать связи между таблицами. Не менее важно понимать какого типа эти связи. Рассмотрим основные из них.

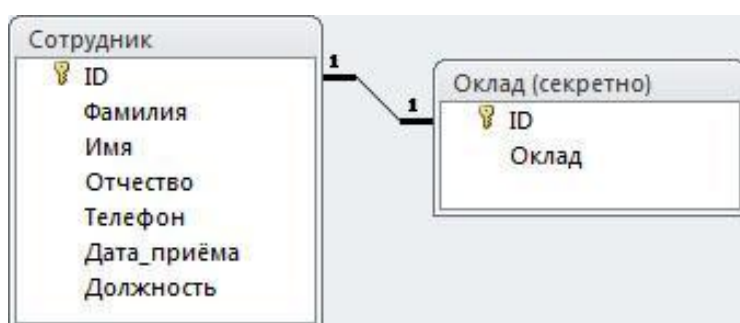
1. Связь «один ко многим»

На рисунке со схемой БД в Примере 3 на связи, соединяющей таблицы отображены 1 и ∞ . Это условное обозначение типа связи 1:M – «один ко многим». Это самая распространенная связь.

Связь «один ко многим» означает, что с одной записью в таблице, на стороне которой стоит «1», могут быть связаны сколько угодно записей из другой таблицы, где стоит « ∞ ». Т.е. во второй таблице может быть несколько записей с одинаковым значением внешнего ключа, что мы и наблюдали в примере 3.

2. Связь «один к одному»

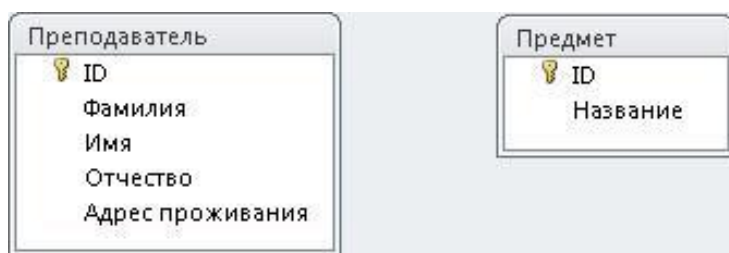
Редко, но встречается связь «1:1», когда каждой записи в первой таблице соответствует 0 или 1 запись в связанной таблице. Это обычно используют для вертикального разделения большой таблицы на две части. Например, сделать часть полей таблицы доступной всем, а остальные поля вынести в другую таблицу и показывать в зависимости от прав пользователя.



3. Связь «многие ко многим»

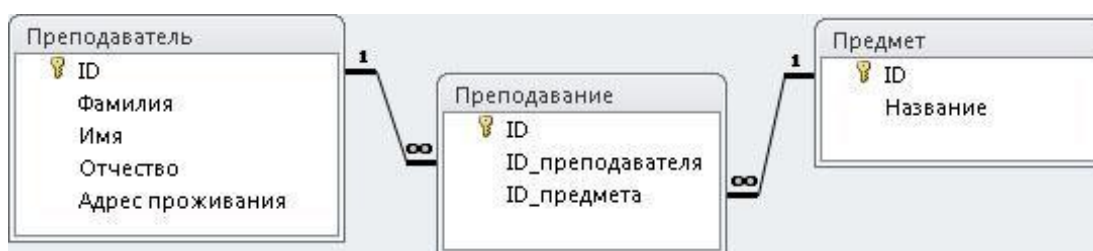
Наиболее сложная, но при этом часто встречаемая связь «M:M».

Например, в школе преподаватель может вести несколько предметов и в то же время предмет с таким наименованием могут преподавать несколько учителей. Значит есть сущности «Преподаватель» и «Предмет», между которыми связь «многие ко многим»:



Реляционные БД не могут на физическом уровне явно реализовать связь «многие ко многим». И действительно, невозможно в плоской таблице к одной записи без ее дублирования привязать разные значения внешнего ключа.

На практике все просто! Появление при проектировании связи «многие ко многим» всегда говорит о том, что не хватает третьей сущности, которая описывает процесс взаимодействия этих двух сущностей. В данном случае явно не хватает сущности «Преподавание». Т.е. нужно просто ввести новую сущность, которая и будет хранить все нужные соответствия ключей из таблиц «Преподаватель» и «Предмет»:



Пример содержимого таблиц:

Таблица «Преподаватель»

ID	Фамилия	Имя	Отчество	Адрес проживания
1	Иванов	Иван	Иванович	г. Москва, ул. Орджоникидзе 3
2	Петров	Петр	Петрович	г. Москва, ул. Рабочая 12
3	Сидорова	Анна	Николаевна	г. Москва, 1-й Зборовский пер. 3

Таблица «Предмет»

ID	Название
1	Математика
2	Алгебра
3	Геометрия
4	География
5	Информатика
6	Литература
7	Русский язык

Таблица «Преподавание»

ID	ID преподавателя	ID предмета
1	1	1
2	1	2
3	1	3
4	2	6
5	2	7

6	3	5
7	3	1
8	3	2

Из третьей таблицы легко определить, что, к примеру, преподаватель Петров ведет литературу и русский язык.

Обратим ещё внимание на то, что в полях **ID_преподавание** и **ID_предмета**, только целочисленные значения. Это существенно увеличивает производительность при поиске информации. Но главное – не приведёт к избыточности и не позволит допустить синтаксическую ошибку при ручном вводе данных.

Упражнение 4.9.1

Дана база данных “Заказы товаров”, состоящая из 3-х таблиц:

- Таблица 1: Менеджер(ФИО, Телефон)
- Таблица 2: Товар(Наименование, Страна, Цена)
- Таблица 3: Заказ(ДатаЗаказа)

Нарисуйте схему БД на бумаге. Для этого выполните следующие задания:

- Добавьте ключевые поля
- Определите связи между таблицами, какая в каждой паре таблица главная, а какая подчинённая
- Добавьте подчинённой таблице поле для связи с главной
- Соедините таблицы и укажите типы связей между таблицами

4.9.3. Зачем нужны СУБД

Мы познакомились с понятием “база данных” и поэтому можем рассмотреть более подробно инструменты, которые позволяют работать с БД - системы управления базами данных (СУБД).

Система управления базами данных СУБД (DataBase Management System, DBMS) – программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также получать к ней контролируемый доступ.

СУБД - это программа, которая:

- создаёт БД и редактирует в ней информацию (добавление, удаление и изменение информации),
- позволяет выполнять вычисления над данными
- предоставляет или ограничивает доступ к БД
- обеспечивает одновременный доступ к данным для нескольких пользователей
- обеспечивает целостность БД
- позволяет восстановить БД в случае сбоя аппаратного или программного обеспечения.

В истории вычислительной техники можно проследить развитие двух основных областей ее использования. Первая область — применение вычислительной техники для выполнения численных расчетов, которые слишком долго или вообще невозможно производить вручную. Характерной особенностью данной области применения вычислительной техники является наличие сложных алгоритмов обработки, которые применяются к простым по структуре данным, объем которых, как правило, сравнительно невелик.

Вторая область - это использование средств вычислительной техники в автоматических или автоматизированных информационных системах. Информационная система представляет собой программно-аппаратный комплекс, обеспечивающий выполнение следующих функций:

- надежное хранение информации в памяти компьютера;
- выполнение специфических для данного приложения преобразований информации и вычислений;
- предоставление пользователям удобного и легко осваиваемого интерфейса.

Обычно такие системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру, поэтому используют БД. Классическими примерами информационных систем являются банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных или железнодорожных билетов, мест в гостиницах и т. д.

Не стоит путать понятия «информационная система» и СУБД. СУБД и БД являются лишь частями информационной системы. Помимо них в ИС всегда присутствует прикладное программное обеспечение (ПО), которое реализует удобный интерфейс пользователя для работы с данными: формы ввода и поиска, отчеты и т.д. Что это значит? Например, в ИС по продаже билетов можно огромным количеством способов выдавать информацию из БД, но в системе реализуют только те формы и отчеты, которые необходимы, например отчет, который показывает, сколько и каких свободных мест есть в поезде на заданную дату, каким образом можно добраться из пункта А в пункт Б с заданной даты и т.д.

Первые БД основывались на иерархической и сетевой моделях. В 1965 году на конференции CODASYL (Conference on Data Systems Languages) была сформирована рабочая группа Data Base Task Group (DBTG), которая занялась определением спецификаций среды, которая допускала бы разработку баз данных и управление данными. В 1971 году DBTG выделила две группы языков:

- **Язык определения данных (Data Definition Language, DDL)**, который позволит описать структуру БД
- **Язык манипулирования данными (Data Manipulation Language, DML)**, предназначенный для управления данными в БД.

Несмотря на то что этот отчет официально не был одобрен Национальным Институтом Стандартизации США (American National Standards Institute - ANSI), большое количество систем было разработано в полном соответствии с этими предложениями группы DBTG. Теперь они называются CODASYL-системами, или DBTG-системами. CODASYL-системы и системы на основе иерархических подходов представляют собой СУБД первого поколения.

В 1970 году Э. Ф. Кодд, работавший в корпорации IBM, опубликовал статью о **реляционной модели данных**, позволявшей устранить недостатки прежних моделей. И это дало начало общеизвестным результатам:

- появился **структурированный язык запросов SQL**, который стал стандартным языком любых реляционных СУБД;
- в 80-х годах были созданы различные коммерческие реляционные СУБД - например, DB2 или SQL/DS корпорации IBM, Oracle корпорации Oracle , др.



Существует несколько сотен различных реляционных СУБД для мейнфреймов и персональных ЭВМ. В качестве примера многопользовательских СУБД может служить система CA-OpenIngres фирмы Computer Associates и система Informix фирмы Informix Software, Inc. Примерами реляционных СУБД для персональных компьютеров являются Access фирмы Microsoft, Paradox и Visual dBase фирмы Borland, а также R-Base фирмы MicroRIM. Реляционные СУБД относятся к СУБД второго поколения.

Однако реляционная модель также обладает некоторыми недостатками - в частности, ограниченными возможностями моделирования. Для решения этой проблемы был выполнен большой объем исследовательской работы. В 1976 году Чен предложил модель "сущность-связь" (Entity-Relationship model - ER-модель), которая

стала основой методологии концептуального проектирования баз данных и методологии логического проектирования реляционных баз данных. В 1979 году Кодд сделал попытку устранить недостатки собственной основополагающей работы и опубликовал расширенную версию реляционной модели - RM/T (1979), затем еще одну версию - RM/V2 (1990).

Попытки создания модели данных, позволяющей более точно описывать реальный мир, называют семантическим моделированием данных (*semantic data modeling*). В ответ на все возрастающую сложность приложений баз данных появились две новых разновидности: объектно-ориентированные СУБД, или ОО СУБД (*Object-Oriented DBMS - OODBMS*), и объектно-реляционные СУБД, или ОР СУБД (*Object-Relational DBMS - ORDBMS*). Попытки реализации подобных моделей представляют собой СУБД третьего поколения.

4.9.4. Классификация СУБД

Существует множество способов классификации СУБД, рассмотрим наиболее значимые.

Классификация по месту размещения СУБД

СУБД делят на локальные, удаленные и распределенные.

1. Локальные

Это самый простой случай, когда прикладное ПО, СУБД и БД находятся на одном устройстве пользователя (например, компьютер, планшет или смартфон). Такой способ реализации хорошо подходит для небольших ИС, где предполагается один пользователь.

В ОС Android для создания локальных БД уже встроена СУБД SQLite.

Для ОС Windows удобнее использовать СУБД, которые дополнительно предоставляют готовые инструменты для создания прикладной части ИС – средства конструирования интерфейса пользователя (Microsoft Access, OpenOffice Base).

Преимущества локальных БД: независимость от работы компьютерных сетей.

Недостатки: невозможность работать нескольким пользователям одновременно.

2. Удаленные и распределенные

Их реализуют, когда предполагается работа множества пользователей с одной БД. Для этих систем зачастую характерен большой объем хранимых данных и они строятся на клиент-серверной архитектуре.

Отличие между удаленной и распределенной базами данных в том, что для первой характерно размещение СУБД и БД на одном сервере. Распределенная же БД размещается на нескольких серверах.

Классификация по способу доступа к БД

1. Файл-серверные

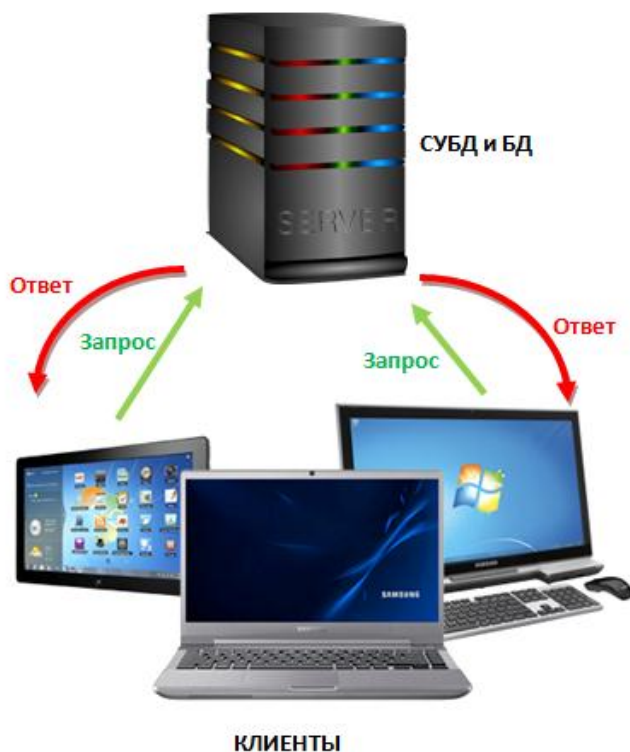
В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на процессор файлового сервера. Недостатки: потенциально высокая загрузка локальной сети; затруднённая или невозможность централизованного управления; затруднённая или невозможность обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД; в системах с низкой интенсивностью обработки данных и низкими пиковыми нагрузками на БД.

На данный момент файл-серверная технология считается устаревшей, а её использование в крупных информационных системах — недостатком.

Примеры: Microsoft Access, Paradox, dBase, Visual FoxPro.

2. Клиент-серверные

Главной отличительной чертой построения клиент-серверной СУБД в том, что прикладное ПО (клиент) физически отделено от СУБД и БД. При этом к БД клиент может обращаться только через СУБД (см. рисунок).



Клиентское приложение на устройстве пользователя по сети отправляет команды (запросы) к СУБД на выполнение различных операций. Команды клиента к СУБД формируются на специальном языке запросов SQL (Structured Query Language – язык структурных запросов).

Что делает сервер:

- ожидает запросы от клиентов по сети;
- выставляет запросы в очередь на выполнение;
- выполняет запросы;
- отправляет ответы обратно клиенту.

В частном случае серверная и клиентская части могут быть установлены на одном компьютере.

Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу (высокая стоимость).

Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность.

Примеры: Oracle, Firebird, Interbase, IBM DB2, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Caché.

3. Встраиваемые

Встраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в

виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL либо через специальные программные интерфейсы [2].

Примеры: SQLite (мы познакомимся с этой СУБД на следующем занятии), BerkeleyDB, Firebird Embedded, Microsoft SQL Server Compact.

Минипроект 4.1. Чемпионат России по футболу/хоккею

Общее задание на минипроект. В простейшем варианте приложение должно позволять вводить расписание матчей чемпионата и результаты игр. По усмотрению учащегося можно усложнить задание, добавив возможность просмотра и поиска информации о членах команд чемпионата, информации, кто забил гол, результативные передачи и т.д.

Задание по текущей теме 4.9: спроектируйте структуру БД приложения в виде схемы.

Источники

1. Поляков К.Ю. Информатика. Углубленный уровень: учебник для 11 класса: Ч.1. М.:БИНОМ. Лаборатория знаний, 2014. Глава 3.
2. Система управления базами данных // Википедия. [2015—2015]. Дата обновления: 27.11.2015. URL: <http://ru.wikipedia.org/?oldid=74774434>

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателям ИТ ШКОЛЫ SAMSUNG Деникиной Наталье Владимировне, Деникину Антону Витальевичу, Шурахтенкову Кириллу Анатольевичу.