Троицкий авиационный технический колледж – филиал федерального
государственного бюджетного образовательного
учреждения высшего образования «Московский государственный технический
университет гражданской авиации»

Методическое пособие
для учащихся заочного отделения

ПМ 02 Разработка и администрирование баз данных
МДК 02.02 Технология разработки и защиты баз данных
Тема 2 РАЗРАБОТКА И ЗАЩИТА УДАЛЕННЫХ БАЗ ДАННЫХ
V семестр (часть1)

г. Троицк, 2020 г.

Учебное пособие для студентов заочного отделения специальности 09.02.03. составлено в соответствии с требованиями Государственного образовательного стандарта к минимуму содержания и уровню подготовки выпускника по специальности 09.02.03 преподавателем Черевковой О.А.

Пособие состоит из 2 частей. В первой части рассматриваются основные понятия, технологии доступа к данным, основные принципы работы с удаленными базами данных, основы проектирования серверной части приложений, во второй – визуальные средства проектирования структуры базы данных, проектирование клиентской части приложения, администрирование и эксплуатация удаленных баз данных. Каждая часть имеет ряд практических упражнений для выработки навыков и закрепления теоретического материала.

Тема1.1: **Введение**
Тема 1.2: **Технологии доступа к данным**

1. Основные понятия и определения
2. Локальные и удаленные базы данных
3. Архитектуры СУБД, преимущества и недостатки
4. Доступ к данным посредством API-интерфейса
5. Универсальные механизмы доступа к данным
6. Технологии доступа к данным трехзвенных СУБД

1. Основные понятия и определения

Локальная сеть – два и более компьютеров, объединенных средствами связи для решения некоторого круга общих задач.

Рабочая станция – узел, подключенный к сети и активно участвующий в информационном обмене.

Сервер – узел сети, который предоставляет свои ресурсы другим узлам.

Базами данных называют электронные хранилища информации, доступ к которым осуществляется с одного или нескольких компьютеров.

Обычно БД создаются для хранения и доступа к данным, содержащим сведения о некоторой предметной области. Т.е. некоторой области человеческой деятельности или области реального мира.

СУБД – это программные средства, предназначенные для создания, наполнения, обновления и удаления баз данных**.**

1. Локальные и удаленные базы данных

В зависимости от расположения программы, использующей данные, и самих данных, а также способа разделения данных между несколькими пользователями различают **локальные и удаленные базы данных**.

Данные локальной базы данных (файлы данных) находятся на одном (локальном) устройстве, в качестве которого может выступать диск компьютера или сетевой диск (диск другого компьютера, работающего в сети).

Для обеспечения разделения данных (доступа к данным) между несколькими пользователями, в локальных базах данных применяется метод, получивший название блокировка файлов.

Суть этого метода заключается в том, что пока данные используются одним пользователем, другой пользователь не может работать с этими данными, т. е. данные для него закрыты, заблокированы.

Paradox, dBase, FoxPro и Access — это локальные базы данных.

Данные удаленной базы данных находятся на удаленном компьютере. Программа работы с удаленной базой данных состоит из двух частей: клиентской и серверной. Клиентская часть программы, работающая на компьютере пользователя, обеспечивает взаимодействие с серверной программой: посредством запросов, передаваемых на удаленный компьютер, предоставляет доступ к данным.

Серверная часть программы, работающая на удаленном компьютере, принимает запросы, выполняет их и пересылает данные клиентской программе.

Запросы представляют собой команды, представленные на языке SQL (Structured Query Language) — языке структурированных запросов.

Программа, работающая на удаленном сервере, проектируется таким образом, чтобы обеспечить одновременный доступ к информации нескольким пользователям. При этом для обеспечения доступа к данным вместо механизма блокировки файлов используют механизм транзакций.

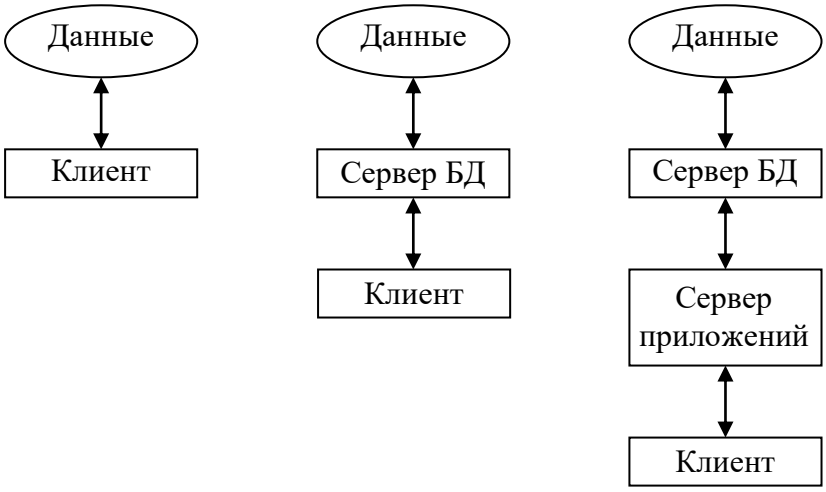
Транзакция — это некоторая последовательность действий, которая должна быть обязательно выполнена над данными перед тем, как они будут переданы. В случае обнаружения ошибки во время выполнения любого из действий вся последовательность действий, составляющая транзакцию, повторяется снова.

Таким образом, механизм транзакций обеспечивает защиту от аппаратных сбоев. Он также обеспечивает возможность многопользовательского доступа к данным.

2. Архитектуры СУБД, преимущества и недостатки

По своей архитектуре СУБД делятся на:

- одно-,
- двух-,
- трехзвенные



В 1) используется единственное звено – клиент- обеспечивающее необходимую логику управления данными и их визуализацию.

В 2) значительную часть логики управления данными берет на себя сервер БД, в то время как клиент в основном занят отображением данных в удобном для пользователя виде.

В 3) используется промежуточное звено – сервер приложений, являющийся посредником между клиентом и сервером БД. Сервер приложений призван полностью избавить клиента от забот по управлению данными и обеспечению связи с сервером БД.

В зависимости от места положения различают локальные и сетевые СУБД.

Все части **локальной** СУБД размещаются на компьютере пользователя БД. Чтобы с одной и той же БД одновременно могло работать несколько пользователей, каждый пользовательский компьютер должен иметь свою копию локальной БД. Проблемой СУБД такого типа является синхронизация копий данных.

Сетевые СУБД:

- файл-серверные
- клиент-серверные
- распределенные

В файл-серверных СУБД все данные размещаются в одном или нескольких каталогах достаточно мощной машины, подключенной к сети. Такой компьютер называется файл-сервером.

Достоинство: относительная простота ее создания и обслуживания – все сводится лишь к развертыванию локальной сети и установке на подключенных к ней компьютерах сетевых операционных систем.

Недостаток: значительная нагрузка на сеть.

Используются в небольших фирмах с количеством рабочих мест до нескольких десятков.

Клиент-серверные (двухзвенные) системы значительно снижают нагрузку на сеть, так как клиент общается с данными через сервер БД, который размещается на машине с данными. Сервер БД принимает запрос от клиента, отыскивает в БД нужную запись и передает ее клиенту. Запрос к серверу формируется на специальном языке структурированных запросов (Structured Server Language), поэтому часто БД называют SQL-серверами.

Серверы БД представляют собой относительно сложные программы, разрабатываемые различными фирмами: SQL Server, Sybase SQL Server корпорации Sybase, Oracle производства одноименной корпорации, DB 2 корпорации IBM и т.д.
Клиент-серверные СУБД масштабируются до сотен и тысяч клиентских мест.

Распределенные СУБД могут содержать несколько десятков и сотен серверов БД.
Количество клиентских мест в них может достигать десятков и сотен тысяч. Обычно такие СУБД работают на предприятиях государственного масштаба, подразделения которых разнесены на значительные расстояния. В распределенных СУБД некоторые серверы могут дублировать друг друга с целью достижения предельно малой вероятности отказов и сбоев, которые могут исказить жизненно важную информацию. Они используют собственные региональные средства связи. Опираясь на возможности Интернета, распределенные системы строят не только предприятия государственного масштаба, но и небольшие коммерческие предприятия, обеспечивая своим сотрудникам работу с корпоративными данными на дому или в командировках.

3. Доступ к данным посредством API-интерфейса

Подавляющее большинство систем управления базами данных содержит в своем составе библиотеки, предоставляющие специальный прикладной программный интерфейс (Application Programming Interface, API) для доступа к данным этой СУБД. Обычно такой интерфейс представляет собой набор функций, вызываемых из клиентского приложения. В случае настольных СУБД эти функции обеспечивают чтение/запись файлов базы данных, а в случае серверных СУБД инициируют передачу запросов серверу баз данных и получение от сервера результатов выполнения запросов или кодов ошибок, интерпретируемых клиентским

приложением. В последнее время Windows-версии клиентского программного обеспечения наиболее популярных серверных СУБД, в частности Microsoft SQL Server, Oracle, Informix, содержат также COM-серверы, предоставляющие объекты для доступа к данным и метаданным.

Преимущество: использование клиентского API является эффективным с точки зрения производительности.

Недостаток: приложение сможет использовать данные только СУБД этого производителя, и замена ее на другую повлечет за собой переписывание значительной части кода клиентского приложения - клиентские API и объектные модели не подчиняются никаким стандартам и различны для разных СУБД.

4. Универсальные механизмы доступа к данным

Другой способ манипуляции данными в приложении базируется на применении универсальных механизмов доступа к данным. Универсальный механизм доступа к данным обычно реализован в виде библиотек и дополнительных модулей, называемых драйверами или провайдерами. Библиотеки содержат некий стандартный набор функций или классов. Дополнительные модули, специфичные для той или иной СУБД, реализуют непосредственное обращение к функциям клиентского API конкретных СУБД.

Преимущество: применения одного и того же абстрактного API для доступа к разным типам СУБД.

Недостатки: невозможность доступа к уникальной функциональности, специфичной для конкретной СУБД, снижение производительности приложений, а также усложнение процедуры поставки приложения - ведь в его состав нужно включать библиотеки, ответственные за реализацию универсальных механизмов, драйверы для тех или иных СУБД, а также обеспечивать настройки, необходимые для их правильного функционирования.

Наиболее популярными среди универсальных механизмов доступа к данным можно назвать следующие:

- Open Database Connectivity (ODBC).
- OLE DB.
- ActiveX Data Objects (ADO).
- Borland Database Engine (BDE).

5. Технологии доступа к данным трехзвенных СУБД

DCOM

Достоинством технологии DCOM является ее органичное единство с Windows и простота реализации, так как основные инструменты этой технологии представляют собой неотъемлемые части ОС Windows. К недостаткам DCOM можно отнести, во-первых, требование, чтобы серверная машина работала под управлением относительно дорогостоящей ОС Windows NT/2000/XP Server, и, во-вторых, полное отсутствие в DCOM средств, характерных для многопользовательского доступа (координация работы нескольких серверов приложений, управление транзакциями). Эти недостатки и стали причиной появления технологии MTS.

MTS

Технология MTS может реализовать удаленный доступ средствами DCOM или сокетов: она не является технологией удаленного доступа как таковой, но служит весьма полезным инструментом управления транзакциями, то есть ориентирована на поддержку трехзвенной архитектуры. Дополнительной особенность является разграничение прав доступа к данным на основе ролей, приписываемых как пользователю, так и данным. Технология MTS является собственностью Microsoft. Она не поддерживается в Delphi, поэтому средства для ее поддержки должны приобретаться отдельно.

CORBA

Хотя в технологии CORBA, как и в DCOM, для удаленного доступа используется механизм интерфейсов, она реализуется на принципах, отличных от COM, что позволяет ей работать в

смешанных (гетерогенных) системах, то есть в сети, где одновременно обслуживаются машины разного типа, работающие под управлением разных ОС, например сервер на «большой конторской» машине (мэйн-фрейме), работающей под управлением ОС UNIX или VMS, и клиентские места на IBM-совместимых ПК, работающих под управлением 32-разрядной версии Windows, Linux или OS/2. CORBA имеет гибкие средства управления загрузкой нескольких серверов приложений и переключения клиентов на работоспособные серверы, а также возможность дополнительной защиты данных с помощью современных криптографических средств.

SOAP

Технология SOAP введена в Delphi 6. В ней используется простой проток доступа к объекту (Simple Object Access Protocol, SOAP) и Транспортный протокол HTTP. Хотя в некоторых случаях (в особенности при необходимости публикации БД в Интернете) эта технология может быть задействована и в трехзвенных БД, ее основное назначение — программирование для Интернета.

Тема 1.3: Введение в работу с УБД. Сервер MySQL
Тема 2.1: Создание БД в СУБД MySQL

1. Технические характеристики MySQL
2. Структура каталогов MySQL
3. Типы таблиц MySQL
4. Типы данных MySQL
5. Создание и удаление базы данных
6. Создание, модификация и удаление таблицы

1. Технические характеристики MySQL

MySQL - это торговая марка MySQL AB (Швеция). MySQL AB - компания, в состав которой входят основатели MySQL и основные разработчики. MySQL AB создана в Швеции Дэвидом Аксмарком (David Axmark), Аланом Ларссом (Allan Larsson) и Майклом Монти Видениусом (Michael Monty Widenius). Является собственностью компании Oracle Corporation

Программное обеспечение MySQL имеет двойное лицензирование. Это означает, что пользователи могут выбирать, использовать ли ПО MySQL бесплатно по общедоступной лицензии GNU General Public License (GPL) или приобрести одну из стандартных коммерческих лицензий MySQL AB. (http://www.gnu.org/licenses/).

Написан на C и C++. Протестирован на множестве различных компиляторов. Работает на различных платформах. Полностью многопоточный т.е, можно легко организовать работу с несколькими процессорами. Очень быстрая базирующаяся на потоках система распределения памяти. SQL-функции реализованы при помощи хорошо оптимизированной библиотеки классов, поэтому они выполняются настолько быстро, насколько это возможно. Последняя версия MySQL 5.1.39

Логотип MySQL с изображением дельфина был создан финским рекламным агентством Priority в 2001 году.

2. Структура каталогов MySQL

MySQL устанавливается в один каталог — обычно это "c:\mysql', — но его можно произвольно изменить.

Каталог «bench» содержит набор скриптов и данных для выполнения теста производительности сервера и сравнения его с другими серверами.

Каталог «bin» — один из самых важных в дистрибутиве, в нем хранятся исполняемые модули самого сервера, клиентской части, а также утилиты и другие файлы. **Файл mysqld.exe** — это, собственно, и есть сам сервер MySQL, который принимает SQL-запросы, обрабатывает их и возвращает результат. **Файл mysql.exe** — это консольный клиент для работы с сервером, который позволяет, вводя команды в консоли, удаленно работать с базами и таблицами, а также администрировать сервер.

Библиотека libmySQL.dll предназначена для обеспечения программного интерфейса (API) для различных программ, работающих напрямую с сервером MySQL.

В каталоге data хранятся базы и таблицы. База mysql — системная. Каталог «data» хранит базы и таблицы, с которыми работает MySQL.

Каждый подкаталог — это отдельная база (имя каталога = имя базы данных), а файлы содержат информацию о структуре и данных в таблицах.

Файлы *.FRM хранят структуру таблиц,

*.MYD — данные таблиц,

*.MYI — индексы таблиц.

Имена файлов идентичны именам таблиц. Каталог «data» и подкаталог «mysql» со всеми файлами являются обязательными для работы.

Каталог «ibdata» создается, только если вы используете таблицы с поддержкой транзакций (InnoDB), и в нем хранятся журналы транзакций.

Каталог «lib» содержит сборки необходимых библиотек для функционирования сервера и клиентских программ, а также утилит сторонних разработчиков.

3. Типы таблиц в MySQL

Для хранения данных в MySQL используется различные типы таблиц.
Каждый из вышеперечисленных механизмов хранения данных обладает своими особенностями и предназначается для решения различных задач.

MyISAM

Каждая таблица хранится в трех файлах. Названия файлов совпадают с названия таблицы, а расширения файлов указывают на тип хранимых данных.
*.frm - содержит формат таблицы
*.myd (MYData) - содержит данные
*.myi (MYIndex) - содержит индексы

MEMORY (HEAP)

Каждая таблица типа MEMORY ассоциируется с файлом на диске. Данный файл содержит информацию о формате таблицы, сама таблица с данными хранится в оперативной памяти. Так как данные хранятся в памяти, то данный тип таблиц обладает большим быстродействием, но при этом велика вероятность потери данных связанных с возможными сбоями в работе сервера (при перезагрузке сервера все данные для таблиц с типом MEMORY пропадают). Идеально подходит для временного хранения данных.

InnoDB

InnoDB это транзакционная база данных с возможностью отмены транзакции, а также с блокировкой доступа на уровне строк. Таблицы InnoDB также позволяют использовать внешние

ключи (FOREIGN KEY).В таблицах InnoDB все действия пользователей осуществляются при помощи транзакций.

MERGE

Таблица MERGE (или таблица MRG_MyISAM) представляет собой совокупность идентичных таблиц MyISAM, которые могут использоваться как одна таблица. Под идентичными таблицами подразумеваются таблицы, созданные с одинаковой структурой и ключами.

BDB (BerkeleyDB)

BerkeleyDB обеспечивает транзакционный обработчик таблиц для MySQL. Использование BerkeleyDB повышает для таблиц шансы уцелеть после сбоев, а также предоставляет возможность осуществлять операции COMMIT и ROLLBACK для транзакций.

4. Типы данных MySQL

Тип	Размер	Описание
CHAR[длина]	Число байтов, равное [длина]	Поле фиксированной от 0 до 255 символов
VARCHAR[длина]	[длина] + 1 байт	Поле переменной длины от 0 до 255 символов
TINYTEXT	[длина] + 1 байт	Строка максимальной длиной 255 символов
TEXT, BLOB	[длина] + 2 байта	Строка максимальной длиной 65535 символов
INT[длина]	4 байта	Целое от -2147483648 до 2147483648
FLOAT	4 байта	Небольшое число с плавающей точкой
DOUBLE[длина.десятичные знаки]	8 байт	Небольшое число с плавающей точкой
DECIMAL[длина.десятичные знаки]	[длина] + 2 байта	Число типа Double, хранящееся в формате с фиксированной десятичной точкой
DATE	3 байта	Дата в формате ГГГГ-ММ-ДД
DATETIME	8 байт	Дата в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIME	3 байта	Время в формате ЧЧ:ММ:СС
ENUM	2 байта	Перечисление, означающее, что в колонке может храниться одно из нескольких возможных значений
SET	8 байт	Перечисление, означающее, что в колонке может храниться более одного из

		нескольких возможных значений

5. Создание и удаление базы данных

Создание БД:
CREATE DATABASE <имя базы даных>

Например: CREATE DATABASE Example

Удаление БД:
DROP DATABASE <имя базы даных>

Например: DROP DATABASE Example

6. Создание, модификация и удаление таблицы

Создание таблицы:

CREATE TABLE имятаблицы [(описания столбцов)]

описания столбцов:
<имя поля> <тип поля> [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [PRIMARY KEY] [определение ссылочной целостности]

Модификация таблицы:

ALTER TABLE <имя таблицы> {ADD|DROP} [описание столбца]

Примеры:
ALTER TABLE t2 ADD d DATE NOT NULL

ALTER TABLE t2 DROP name

Удаление таблицы:

DROP TABLE <имя таблицы>

Тема 2.2: Ссылочная целостность

1. Ограничения столбцов. Первичный и уникальный ключи
2. Ссылочная целостность
3. Создание и удаление индексов

1. Ссылочная целостность

Структуру таблицы определяют следующие элементы:
- описания столбцов
- ограничения столбцов

- описания ключей
- описания индексов
- ограничения таблицы

Ограничения столбца:
Являются необязательными элементами, однако их использование позволяет автоматизировать процесс ввода значений, предотвращать ошибки ввода.

Формат:
[default <значение>]
[not null]

Операнд default определяет для столбца значение по умолчанию, которое автоматически заносится в столбец при добавлении к таблице новой записи.

Операнд NOT NULL указывает, что столбец не может быть пустым и должен содержать значение допустимого типа и диапазона.

Описание ключей
Описание первичного ключа:

Primary key (<список столбцов ключа>)

```
Create table personnel2
(code int not null,
Name varchar(30),
Primary key(code));
```

Или так:

```
Create table personnel2
(code int not null primary key,
Name varchar(30));
```

Описание уникального ключа:

Unique (<список столбцов ключа>)

Уникальный ключ служит для обеспечения единственности значений ключевого столбца(столбцов) и ссылочной целостности.

2. Ссылочная целостность

Действие ограничений ссылочной целостности заключается в следующем: если для записи главной таблицы есть связанные с ней записи в подчиненной таблице(таблицах), то эту запись нельзя удалить, а также изменить значения столбцов образующих ключ.

Формат определения ссылочной целостности:

[Constraint <имя ограничения>]
Foreign key (<список столбцов ключа>)
References <имя главной таблицы>[<список столбцов ключа главной таблицы>]

При задании ограничений ссылочной целостности ключу (первичному или уникальному) главной таблицы ставится в соответствие внешний ключ починенной таблицы. Для описания внешнего ключа используется операнд foreign key. Внешний ключ используется для ограничения ссылочной целостности.

Пример:
Create table store
(s_code int not null primary key,
S_name varchar(20) not null,
S_price float,
S_number float
);

Create table cards
(c_code int not null primary key,
C_code2 int not null,
C_move varchar(20) not null,
C_date date,
Constraint rStoreCards
Foreign key (c_code2) references Store
);

Удалить ограничения ссылочной целостности:

Alter table <имя таблицы>
Drop <имя ограничения ссылочной целостности>;

3. Создание и удаление индексов

Создание индекса: **CREATE INDEX <имя индекса> ON <имя таблицы> (список полей)**

Пример: CREATE INDEX part_of_name ON customer (name(10));

Удаление индекса: **DROP INDEX <имя индекса> ON <имя таблицы>**

Тема 2.3: Визуальные средства проектирования

1. Понятия о средствах Case - проектирования баз данных
2. Избыточность данных и аномалии
3. Нормализация данных

1. Понятия о средствах Case - проектирования баз данных

С ростом размера базы данных, когда в нее включаются нескольких десятков и сотен различных таблиц, возникает проблема сложности организации данных, в том числе установления взаимосвязей между таблицами. Для облегчения решения этой проблемы предназначены системы автоматизации разработки приложений, или средства CASE (Computer Aided Software Engineering).

Средства CASE представляют собой программы, поддерживающие процессы создания и/или сопровождения информационных систем, такие как анализ и формулировка требований, проектирование БД и приложений, генерация кода, тестирование, обеспечение качества, управление конфигурацией и проектом. То есть средства CASE позволяют решать более масштабные задачи, чем просто проектирование БД.

2. Избыточность данных и аномалии

При разработке структуры БД могут возникнуть проблемы, связанные:
- □ с избыточностью данных;
- □ с аномалиями.

Под дублированием данных понимается множественное повторение одних и тех же данных.

При этом различают простое (неизбыточное) дублирование и избыточное дублирование данных.

Избыточность данных при выполнении операций с ними приводит к различным аномалиям — нарушению целостности БД.

Выделяют аномалии:
- □ удаления;
- □ обновления;
- □ ввода.

Аномалия ввода заключается в том, что при вводе в таблицу новой строки для ее полей могут быть введены недопустимые значения. Например, значение не входит в заданный диапазон или не задано значение поля, которое в обязательном порядке должно быть заполнено (не может быть пустым).

Аномалия удаления заключается в том, что при удалении одного из дублированных значений (или соответствующей строки таблицы) будет потеряна информация об экземпляре таблицы.

Аномалия обновления заключается в том, что при изменении значения некоторого поля для одной записи, эти же значения для других записей не изменяются.

Неизбыточное дублирование является естественным и допустимым.

Пример неизбыточного дублирования данных

Сотрудник	**Телефон**
Иванов П. Л.	123
Петров А. Ф.	123
Сидоров О. Е.	456
Кузнецова В. А.	789
Васин И. Г.	123

Пример избыточного дублирования данных

Сотрудник	**Кабинет**	**Телефон**
Иванов П. Л.	17	123
Петров А. Ф.	17	---
Сидоров О. Е.	22	456
Кузнецова В. А.	8	789
Васин И. Г.	17	---

От него можно избавиться, например, с помощью разбиения таблицы на две новых **Разбиение** — это процесс деления таблицы на несколько таблиц с целью поддержания целостности данных, т. е. устранения избыточности данных и аномалий.

Список сотрудников и номеров кабинетов	
Сотрудник	**Кабинет**
Иванов П.Л.	17
Петров А.Ф.	17
Сидоров О.Е.	22
Кузнецова В.А.	8
Васин И.Г.	17

Список номеров кабинетов и телефонов	
Кабинет	**Телефон**
17	123
8	789
22	456

Таблицы связаны между собой по номеру комнаты. Для получения информации о номере телефона сотрудника из первой таблицы по фамилии сотрудника нужно считать номер его комнаты, после чего из второй таблицы по номеру комнаты считывается номер телефона.

3. Нормализация данных

Нормализация БД – это процесс уменьшения избыточности информации в БД.

Процесс проектирования БД с использованием метода нормальных форм является итерационным (пошаговым) и заключается в последовательном переводе по определенным правилам отношений из первой нормальной формы в нормальные формы более высокого порядка. Каждая следующая нормальная форма ограничивает определенный тип функциональных зависимостей, устраняет соответствующие аномалии при выполнении операций над отношениями БД и сохраняет свойства предшествующих нормальных форм.

Выделяют следующую последовательность нормальных форм:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- усиленная третья нормальная форма, или нормальная форма Бойса-Кодда;
- четвертая нормальная форма (4НФ);
- пятая нормальная форма (5НФ).

Требования нормальных форм:

1 НФ

1. Наличие первичного ключа
2. Поля должны быть неделимы
3. Отсутствуют повторяющиеся группы полей

2НФ

1. Таблица должна удовлетворять требованиям первой нормальной формы
2. Неключевые поля не должны зависеть от части составного первичного ключа

3НФ

1. Таблица должна удовлетворять требованиям второй нормальной формы
2. Ни одно из неключевых полей не должно зависеть от другого неключевого поля (полей).

Тема 3.1: Способы доступа к данным
Тема 3.2: Запросы на выборку данных. Оператор Select

1. Операторы языка SQL
2. Оператор Select. Простые и сложные запросы на выборку данных. Представления

1. Операторы языка SQL

SQL (Structured Query Language**)** – это структурированный язык запросов к реляционным базам данных (БД). SQL является декларативным языком, основанным на операциях реляционной алгебры.

Существуют два стандарта SQL, определённые американским национальным институтом стандартов (ANSI): SQL-89 (SQL-1) и SQL-92 (SQL-2).

SQL включает основные три группы средств:

- DDL (Data Definition Language) – язык определения данных
- DML (Data Manipulation Language) – язык манипулирования данными
- DCL (Data Control Language) – язык управления данными

2. Оператор Select. Простые и сложные запросы на выборку данных

Извлечение данных из отношений выполняется с помощью команды **SELECT** (селекция). Результатом выполнения команды *SELECT* является временное отношение, которое помещается в курсор (специальную область памяти СУБД) и обычно сразу выводится на экран.

Синтаксис команды:

```
SELECT * | { [ ALL | DISTINCT ] <список выбора>.,..}
FROM {<имя таблицы> [<алиас>] }.,..
[ WHERE <условие>]
[ GROUP BY {<имя поля> | <целое>}.,.. [ HAVING <условие>] ]
[ ORDER BY {<имя поля> | <целое> [ ASC | DESC ] }.,..]
[ UNION [ALL] SELECT …];
```

Основные предложения команды *SELECT*:

SELECT – после этого ключевого слова указывается **список выбора** – список выражений, которые будут образовывать результирующее отношение. Выражению можно сопоставить временный синоним (алиас), который будет названием поля результирующего отношения, например:

sal*0.87+bonus as salary

Если надо вывести все поля из тех отношений, к которым обращается данный запрос, можно указать символ * (если в отношениях нет полей с одинаковыми именами). В этом случае сначала будут выведены поля таблицы, стоящей первой в предложении *FROM*, затем – второй и т.д. Поля, относящиеся к одной таблице, будут выводиться в том порядке, в каком они были записаны при создании таблицы.

FROM – в этом предложении указывается имя таблицы (имена таблиц), в которой будет производиться поиск.

WHERE – содержит условия выбора отдельных записей.

GROUP BY – группирует записи по значению одного или нескольких полей. Каждой группе в результирующем отношении соответствует одна запись.

HAVING – позволяет указать условия выбора для групп записей. Может использоваться только после *group by*.

ORDER BY – упорядочивает результирующие записи по значению одного или нескольких полей: *ASC* – по возрастанию, *DESC* – по убыванию.

Порядок выполнения операции *SELECT* такой:

1. Выбор из указанной таблицы тех записей, которые удовлетворяют условию отбора (*where*).
2. Группировка полученных записей (*group by*).
3. Выбор тех групп, которые удовлетворяют условию отбора (*having*).
4. Сортировка записей в указанном порядке (*order by*).
5. Извлечение из записей полей, заданных в списке выбора, и формирование результирующего отношения.

Если во фразе *FROM* указаны две и более таблицы, то эта последовательность действий выполняется для декартова произведения указанных таблиц.

Пример 1: Запрос по двум таблицам. Список сотрудников с детьми:

```
select e.name, c.name child, c.born
from emp e, children c /* e, c – алиасы */
where e.tabno = c.tabno /* условие соединения */
order by e.name, c.born;
```

Расширение возможностей команды *SELECT* достигается за счёт применения различных операторов, предикатов и функций.

Операторы:

- сравнения: =, >, <, >=, <=, <>;
- логические: AND, OR, NOT.

Пример 2: Составить список сотрудников второго и третьего отдела, имеющих оклады выше 4600 рублей:

```
select depno, name, salary
from emp
where salary>4600 and (depno=2 or depno=3)
order by name;
```

Предикаты, используемые в запросах:

- IN:

field IN (*список значений*)

– определяет множество значений, с которыми будет сравниваться значение указанного поля *field*. Предикат считается истинным, если значение поля *field* равно хотя бы одному из элементов множества.

- BETWEEN:

field BETWEEN *значение1* AND *значение2*

– определяет, входит ли значение поля *field* в указанные границы. Если значение поля меньше, чем *значение1*, или больше, чем *значение2*, предикат возвращает "ложь".

- LIKE:

field LIKE '*образец*'

– используется для поиска подстрок, применяется только в полям типа CHAR, VARCHAR. Возможно использование шаблонов: '_' – один любой символ и '%' – произвольное количество символов (в т.ч., ни одного);

- IS [NOT] NULL:

field IS [NOT] NULL

– определяет, установлено ли значение поля. Все другие предикаты и операторы сравнения возвращают неопределённый результат (*null*), если хотя бы один из операндов имеет значение *null*.

Пример 3: Список программистов и ведущих программистов:

```
select depno, name, post from emp
where post like ('%программист%');
```

Пример 4: Список сотрудников старше 40 лет из 1-го и 3-го отделов:

```
select depno, name from emp
where depno in (1, 3) and
(extract (year from curdate()) – extract(year from born)) > 40; /*(текущий год) – (год рождения)*/
```

Пример 5: Список сотрудников, не имеющих телефонов:

```
select tabno, name, post
from emp
where tel is null;
```

Функции агрегирования:

- COUNT – определяет в результате количество строк (записей) или значений поля, не являющихся *NULL*-значениями.
- SUM – определяет арифметическую сумму значений указанного числового поля в результирующем множестве записей.
- AVG – определяет среднее арифметическое значений указанного числового поля в результирующем множестве записей;
- MAX, MIN – определяет максимальное (минимальное) значение указанного поля в результирующем множестве.

Пример 6: Посчитать количество сотрудников по отделам:

```
select depno, count(*), ‘ сотрудник(а)’
from emp
group by depno;
```

Пример 7: Сумма зарплаты по отделам:

```
select depno, sum(salary) as sal
from emp
group by depno;
```

Предложение *UNION* позволяет объединять результаты нескольких запросов *SELECT*.

Результаты этих запросов должны быть построены по одной схеме.

Предложение *ORDER BY* может встречаться в таком запросе один раз – в конце последнего предложения *SELECT*.

Пример 8: Посчитать количество сотрудников по всем отделам:

```
select depno, count(name), ‘сотрудников’
from emp
group by depno
union
select depno, 0, ‘сотрудников’
from depart
where depno not in (select distinct depno from emp)
order by 1; /* упорядочение по первому столбцу */
```

Работа с представлениями

Представление (view, обзор) – это хранимый запрос, создаваемый на основе команды *SELECT*. Представление реально не содержит данных. Запрос, определяющий представление, выполняется тогда, когда к представлению происходит обращение с другим запросом, например, *SELECT*, *UPDATE* и т.д.

Создание представления выполняется командой **CREATE VIEW**:

CREATE VIEW <имя представления> [(<имя столбца>.,..)] AS <запрос>;

Запрос, на основании которого создаётся представление, называется **определяющим запросом**, а таблицы, к которым происходит обращение в определяющем запросе – **базовыми таблицами**. Определяющий запрос не может включать предложение *ORDER BY*.

Если не указывать имена столбцов, то они получат названия по именам, перечисленным в списке выбора определяющего запроса. Указывать имена столбцов представления обязательно, если список выбора содержит агрегирующие функции или столбцы с одинаковыми именами из разных таблиц.

Пример 9: Создание представления "Сотрудники с детьми":

```
create view emp_child(name, child, born)
as select e.name, c.name, c.born
from emp e, children c
where e.tabno = c.tabno;

select * from emp_child order by name, born;
```

Подзапросы

Подзапросы можно разделить на следующие группы в зависимости от возвращаемых результатов:

- запросы, возвращающие от 0 до нескольких элементов (начинаются с оператора *IN* или модифицированного оператора сравнения);
- запросы на существование (начинаются с оператора *EXISTS*);
- запросы, возвращающие единственное значение (начинаются с немодифицированного оператора сравнения).

Подзапросы бывают коррелированные и некоррелированные. Коррелированные подзапросы содержат условия, зависящие от значений полей в основном запросе. Запросы на существование обычно являются коррелированными.

Рассмотрим операторы, которыми модифицируются операторы сравнения:

- ALL – оператор, эквивалентный понятию "все". Например:

> ALL (< ALL) – больше (меньше) каждого значения элементов результирующего множества.

- ANY (SOME) – оператор, эквивалентный понятию "любой".

= ANY – равно одному из значений элементов результирующего множества (эквивалентно использованию предиката IN).

> ANY (< ANY) – больше (меньше) любого значения элементов результирующего множества.

- EXISTS – оператор, эквивалентный понятию "существует". Часто используется с логическим оператором *NOT*.

Если список, модифицированный оператором *ALL*, содержит *NULL*-значение, то результирующий запрос будет пуст, т.к. нельзя сравнить никакое значение с *NULL*-значением.

Выражение *<>ANY*(…) не эквивалентно *NOT IN*: оно выполняется всегда, кроме случаев *NULL*-значений.

Пример 10: Выдать список бездетных сотрудников:

```
select * from emp e
where not exists (select * from children c where e.tabno=c.tabno);
```

Пример 11: Выдать список сотрудников, имеющих детей:

```
select * from emp
where tabno in (select distinct tabno from children);
```

Пример 12: Список сотрудников, оклад которых меньше среднего на предприятии:

```
select depno, name, salary
from emp
where salary < ANY(select avg(salary) from emp);
```

Тема 3.3: Добавление, обновление и удаление данных

1. Оператор добавления данных
2. Оператор обновления данных
3. Оператор удаление данных

1. Оператор добавления данных

INSERT – добавление записи в таблицу.

Синтаксис:

```
INSERT INTO <имя таблицы> [(<имя поля>.,..)]
VALUES (<список значений полей>);
```

Пример: Добавить в таблицу "Сотрудники" новую запись:

```
insert into emp

values(3, '112', 'Попов В.Г.', 'экономист', 400*13.5, '1979–12–23', '5–34–11');
```

2. Оператор обновления данных

UPDATE – обновление данных в таблице.

Синтаксис:

```
UPDATE <имя таблицы>
SET {<имя поля> = <выражение>}.,..
[WHERE <условие>];
```

Пример: Изменить должность и зарплату сотрудника Попова В.Г., табельный номер 112:

```
update emp
set post = 'ст. экономист', salary = salary+1000
where tabno = '112';
```

3. Оператор удаление данных

DELETE – удаление записей из таблицы.

Синтаксис:

```
DELETE FROM <имя таблицы> [WHERE <условие> ];
```

Пример: Удалить запись о сотруднике Попове В.Г., табельный номер 112:

```
delete from emp where tabno = '112';
```