МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждения высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«23» 03 2023 г.

БЕЗОПАСНОСТЬ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.65

Составитель: А.В. Митрофанов

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Безопасность распределенных баз данных: методические указания по выполнению практических работ по дисциплине «безопасность распределенных баз данных» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов. – Курск, 2023. – 35 с.: ил. 9. – Библиогр.: с. 35.

Содержат теоретические сведения о системе управления базами данных, основные понятия реляционной модели данных и её структуры, реализации защиты данных в SQL, созданию запросов на языке SQL, реализации резервного копирования данных. Указывается порядок выполнения практической работы.

Методические указания по выполнению практических работ по дисциплине «Безопасность систем баз данных», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 80. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

Практическая работа №1

ПОНЯТИЕ СУБД, ПРОЕКТИРОВАНИЕ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Система управления базами данных— это универсальное программное средство, предназначенное для организации хранения и обработки логически взаимосвязанных данных и обеспечения быстрого доступа к ним.

СУБД дают возможность программистам и системным аналитикам быстро разрабатывать более совершенные программные средства обработки данных, а конечным пользователям осуществлять непосредственное управление данными. СУБД должна обеспечивать пользователю поиск, модификацию и сохранность данных, оперативный доступ, защиту целостности данных от аппаратных сбоев и программных ошибок, разграничение прав и защиту от несанкционированного доступа, поддержку совместной работы нескольких пользователей с данными.

СУБД представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная база данных. Программные средства включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование БД, трансляторы.

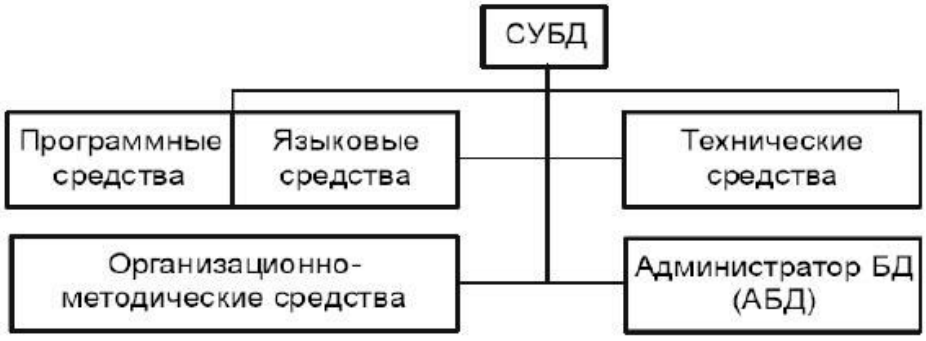


Рис. 1. Состав СУБД

Базовыми внутренними языками программирования являются языки четвертого поколения. В качестве базовых языков могут использоваться C, C++, Pascal, Object Pascal. Язык C++ позволяет строить программы на языке Visual Basic с широким спектром возможностей, более близком и понятном даже пользователю-непрофессионалу, и на непроцедурном (декларативном) языке структурированных запросов SQL. Следует отметить, что исторически для системы управления базой данных сложились три языка:

1. язык описания данных (ЯОД), называемый также языком описания схем, - для построения структуры ("шапки") таблиц БД;
2. язык манипулирования данными (ЯМД) - для заполнения БД данными и операций обновления (запись, удаление, модификация);
3. язык запросов - язык поиска наборов величин в файле в соответствии с заданной совокупностью критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД (язык преобразования критериев в систему команд).

В настоящее время функции всех трех языков выполняет язык SQL, относящийся к классу языков, базирующихся на исчислении кортежей (кортеж чаще всего является единицей информации), языки СУБД FoxPro, Visual Basic for Application (СУБД Access) и т.д.

Вместе с тем сохранились и языки запросов, например язык запросов по примеру Query By Example (QBE) класса исчисления доменов. Отметим, что эти языки в качестве "информационной единицы" БД используют отдельную запись. С помощью языков БД создаются приложения, базы данных и интерфейс пользователя, включающий экранные формы, меню, отчеты. При создании БД на базе СУБД FoxPro эти элементы (объекты) фиксируются в отдельных файлах, которые, в свою очередь, сосредоточиваются в одном файле, называемом проектом. После отработки БД проект преобразуется в приложение. В СУБД Access все созданные объекты размещаются в одном файле.

Существуют универсальные системы управления базами данных, используемые для различных приложений. При настройке универсальных СУБД для конкретных приложений они должны обладать соответствующими средствами. Процесс настройки СУБД на конкретную область применения называется генерацией системы. К универсальным СУБД относятся, например системы Microsoft Access, Microsoft Visual FoxPro, Borland dBase, Borland Paradox, Oracle.

Главная функция СУБД заключается в обеспечении пользователя информационной базы средствами для работы с данными в абстрактных термина, не связанных с особенностью хранения информации в памяти компьютера и для выполнения на этой основе операций по обработке данных для различных приложений.

К основным функциям СУБД относятся:

- Непосредственное управление данными во внешней и оперативной памяти и обеспечение эффективного доступа к данным в процессе решения задач.
- Поддержание целостности данных и управление транзакциями.
- Ведение системного журнала изменений в базе данных, что обеспечивает восстановление базы данных после технического или программного сбоя.
- Реализация поддержки языка описания данных и языка запросов к данным.
- Обеспечение безопасности данных.
- Обеспечение параллельного доступа к данным нескольких пользователей

Реляционная модель данных – логическая модель данных. В настоящее время эта модель является фактическим стандартом, на который ориентируются практически все современные коммерческие СУБД.

Реляционная модель является удобной и наиболее привычной формой представления данных в виде таблицы (отношения). Каждое отношение имеет имя и состоит из поименованных атрибутов (столбцов) данных. Одним из основных преимуществ реляционной модели является ее однородность. Все

данные хранятся в таблицах, в которых каждая строка имеет один и тот же формат. Каждая строка в таблице представляет некоторый объект реального мира или соотношение между объектами.

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие:

- реляционная БД – набор нормализованных отношений;
- отношение – файл, плоская таблица, состоящая из столбцов и строк; таблица, в которой каждое поле является атомарным;
- домен – совокупность допустимых значений, из которой берется значение соответствующего атрибута определенного отношения. С точки зрения программирования, *домен* – это тип данных;
- универсум – совокупность значений всех полей или совокупность доменов;
- кортеж – запись, строка таблицы;
- кардинальность - количество строк в таблице;
- атрибуты – поименованныеполя, столбцы таблицы;
- степень отношения - количество полей (столбцов);
- схема отношения – упорядоченный список имен атрибутов;
- схема реляционной БД – совокупность схем отношений;
- первичный ключ – уникальный идентификатор с неповторяющимися записями – столбец или некоторое подмножество столбцов, которые единственным образом определяют строки.

Первичный ключ, который включает более одного столбца, называется множественным, или комбинированным, или составным, или суперключом.

Правило целостности объектов утверждает, что первичный ключ не может быть полностью или частично пустым.

Структура реляционной модели данных:

-структурная;

-манипуляционная;

-целостная.

Структурная часть модели определяет то, что единственной структурой данных является нормализованное n-арное отношение. Отношения удобно представлять в форме таблиц, где каждая строка есть кортеж, а каждый столбец – атрибут, определенный на

некотором домене. Реляционная база данных представляет собой конечный набор таблиц.

Манипуляционная часть модели определяет два фундаментальных механизма манипулирования данными – реляционная алгебра и реляционное исчисление. Основной функцией манипуляционной части реляционной модели является обеспечение меры реляционности любого конкретного языка реляционных БД.

Язык называется реляционным, если он обладает не меньшей выразительностью и мощностью, чем реляционная алгебра или реляционное исчисление.

Целостная часть модели определяет требования *целостности сущностей* и *целостности ссылок*. Первое требование состоит в том, что любое отношение должно обладать первичным ключом. Требование целостности по ссылкам, или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать).

Структура реляционной модели данных

Можно провести аналогию между элементами реляционной модели данных и элементами модели "сущность-связь". Реляционные отношения соответствуют наборам сущностей, а кортежи – сущностям. Поэтому, также как и в модели "сущность-связь" столбцы в таблице, представляющей реляционное отношение, называют атрибутами.

Пример базы данных, содержащей сведения о подразделениях предприятия и работающих в них сотрудниках, применительно к реляционной модели будет иметь вид:

Рис. 2. Построение диаграмм

База данных о подразделениях и сотрудниках предприятия

Например, связь между отношениями ОТДЕЛ и СОТРУДНИК создается путем копирования первичного ключа "Номер_отдела" из первого отношения во второе. Таким образом:

-для того, чтобы получить список работников данного подразделения, необходимо:

1. из таблицы ОТДЕЛ установить значение атрибута "Номер_отдела", соответствующее данному "Наименованию_отдела"
2. выбрать из таблицы СОТРУДНИК все записи, значение атрибута "Номер_отдела" которых равно полученному на предыдущем шаге

-для того, чтобы узнать в каком отделе работает сотрудник, нужно выполнить обратную операцию:

1. определяем "Номер_отдела" из таблицы СОТРУДНИК
2. по полученному значению находим запись в таблице ОТДЕЛ

Атрибуты, представляющие собой копии ключей других отношений, называются внешними ключами.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ №1

1. Изучить среду ERwin.
2. Выполнить практическое задание, соответствующее своему варианту, номер которого указан в таблице 1. Для выполнения этого задания необходимо:
   - разработать таблицы данных по заданной предметной области;
   - обозначить в каждой таблице, при необходимости, первичный ключ (primery key);
   - создать связи между таблицами.

Таблица 1

Вариант	**Предметная область**
1	Книги из домашней библиотеки
2	Продуктовый магазин
3	Домашняя видеотека(кинофильмы)
4	Домашняя фонотека (диски с музыкальными произведениями).
5	Кинологический клуб

Практическая работа №2

ОРГАНИЗАЦИЯ ЗАЩИТЫ РБД

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

За реализацию системы безопасности отвечает СУБД. Язык SQL является фундаментом системы безопасности реляционной СУБД: требования, предъявляемые к системе защиты информации в базе данных, формулируются с помощью инструкций SQL. С защитой данных в SQL связаны два фундаментальных принципа: проверка полномочий и проверка подлинности (аутентификация).

Проверка полномочий основана на том, что каждому пользователю или процессу информационной системы соответствует набор действий, которые он может выполнять по отношению к определенным объектам. Проверка подлинности означает достоверное подтверждение того, что пользователь или процесс, пытающийся выполнить санкционированное действие, действительно тот, за кого он себя выдает.

Система назначения полномочий имеет в некотором роде иерархический характер. Самыми высокими правами и полномочиями обладает системный администратор или администратор сервера БД. Традиционно только этот тип пользователей может создавать других пользователей и наделять их определенными полномочиями.

СУБД в своих системных каталогах хранит как описание самих пользователей, так и описание их привилегий по отношению ко всем объектам.

Далее схема предоставления полномочий строится по следующему принципу. Каждый объект в БД имеет владельца — пользователя, который создал данный объект. Владелец объекта обладает всеми правами-полномочиями на данный объект, в том числе он имеет право предоставлять другим пользователям полномочия по работе с данным объектом или забирать у пользователей ранее предоставленные полномочия.

В ряде СУБД вводится следующий уровень иерархии пользователей — это администратор БД. В этих СУБД один сервер может управлять множеством СУБД (например, MS SQL Server, Sybase).

В СУБД Oracle применяется однобазовая архитектура, поэтому там вводится понятие подсхемы — части общей схемы БД и вводится пользователь, имеющий доступ к подсхеме.

В стандарте SQL не определена команда создания пользователя, но практически во всех коммерческих СУБД создать пользователя можно не только в интерактивном режиме, но и программно с использованием специальных хранимых процедур. Однако для выполнения этой операции пользователь должен иметь право на запуск соответствующей системной процедуры.

В стандарте SQL определены два оператора: GRANT и REVOKE соответственно предоставления и отмены привилегий.

Оператор предоставления привилегий имеет следующий формат:

```
GRANT {<список действий>| ALL PRIVILEGES }
ON <имя_объекта> TO {<имя_пользователя> | PUBLIC }
[WITH GRANT OPTION ]
```

Здесь список действий определяет набор действий из общедопустимого перечня действий над объектом данного типа.

Параметр ALL PRIVILEGES указывает, что разрешены все действия из допустимых для объектов данного тина.

<имя_объекта> — задает имя конкретного объекта: таблицы, представления, хранимой процедуры, триггера.

<имя_пользователя> или PUBLIC определяет, кому предоставляются данные привилегии.

Параметр WITH GRANT OPTION является необязательным и определяет режим, при котором передаются не только права на указанные действия, но и право передавать эти права другим пользователям. Передавать права в этом случае пользователь может только в рамках разрешенных ему действий.

Рассмотрим пример, пусть у нас существуют три пользователя с абсолютно уникальными именами userl, user2 и userS. Все они являются пользователями одной БД.

User1 создал объект Tab1, он является владельцем этого объекта и может передать права на работу с эти объектом другим пользователям. Допустим, что пользователь user2 является оператором, который должен вводить данные в TaЫ (например, таблицу новых заказов), а пользователь user 3 является большим начальником (например, менеджером отдела), который должен регулярно просматривать введенные данные.

Для объекта типа таблица полным допустимым перечнем действий является набор из четырех операций: SELECT, INSERT, DELETE, UPDATE. При этом операция обновление может быть ограничена несколькими столбцами.

Общий формат оператора назначения привилегий для объекта типа таблица будет иметь следующий синтаксис:

```
    GRANT     {[SELECT][.INSERT][.DELETE][,UPDATE
(<список столбцов»)]}
    ON <имя таблицы»
    TO {<имя_пользователя> | PUBLIC } [WITH GRANT
OPTION ]
```

Тогда резонно будет выполнить следующие назначения:

```
GRANT INSERT
ON Tab1
TO user2
GRANT SELECT
ON Tabl
TO user3
```

Эти назначения означают, что пользователь user2 имеет право только вводить новые строки в отношение Tab1, а пользователь user3 имеет право просматривать все строки в таблице Tab1.

При назначении прав доступа на операцию модификации можно уточнить, значение каких столбцов может изменять пользователь. Допустим, что менеджер отдела имеет право изменять цену на предоставляемые услуги. Предположим, что цена

задается в столбце COST таблицы TaЫ. Тогда операция назначения привилегий пользователю user3 может измениться и выглядеть следующим образом:

```
GRANT SELECT. UPDATE (COST) ON Tabl TO user3
```

Если наш пользователь userl предполагает, что пользователь user4 может его замещать в случае его отсутствия, то он может предоставить этому пользователю все права по работе с созданной таблицей Tab1.

```
GRANT ALL PRIVILEGES
ON Tabl
TO user4 WITH GRANT OPTION
```

В этом случае пользователь user4 может сам назначать привилегии по работе с таблицей Tab1 в отсутствие владельца объекта пользователя user1. Поэтому в случае появления нового оператора пользователя user5 on может назначить ему права на ввод новых строк в таблицу командой

```
GRANT INSERT
ON Tab1
TO user5
```

Если при передаче полномочий набор операций над объектом ограничен, то пользователь, которому переданы эти полномочия, может передать другому пользователю только те полномочия, которые есть у него, или часть этих полномочий. Поэтому если пользователю user4 были делегированы следующие полномочия:

```
GRANT SELECT, UPDATE, DELETE
ON Tabl
TO user4
WITH GRANT OPTION
```

то пользователь user4 не сможет передать полномочия на ввод данных пользователю user5, потому что эта операция не входит в список разрешенных для него самого.

Кроме непосредственного назначения прав по работе с таблицами эффективным методом защиты данных может быть создание представлений, которые будут содержать только необходимые столбцы для работы конкретного пользователя и предоставление прав на работу с данным представлением пользователю.

Так как представления могут соответствовать итоговым запросам, то для этих представлений недопустимы операции изменения, и, следовательно, для таких представлений набор допустимых действий ограничивается операцией SELECT. Если же представления соответствуют выборке из базовой таблицы, то для такого представления допустимыми будут все 4 операции: SELECT, INSERT, UPDATE и DELETE.

Для отмены ранее назначенных привилегий в стандарте SQL определен оператор REVOKE. Оператор отмены привилегий имеет следующий синтаксис:

```
REVOKE {<список операций | ALL PRIVILEGES}
ON <имя_объекта>
FROM {<список пользователей | PUBLIC }
{CASCADE | RESTRICT }
```

Параметры CASCADE или RESTRICT определяют, каким образом должна производиться отмена привилегий. Параметр CASCADE отменяет привилегии не только пользователя, который непосредственно упоминался в операторе GRANT при предоставлении ему привилегий, но и всем пользователям, которым этот пользователь присвоил привилегии, воспользовавшись параметром WITH GRANT OPTION.

Например, при использовании операции:

```
REVOKE ALL PRIVILEGES
ON Tab1
TO user4 CASCADE
```

будут отменены привилегии и пользователя users, которому пользователь user4 успел присвоить привилегии.

Параметр RESTRICKT ограничивает отмену привилегий только пользователю, непосредственно упомянутому в операторе REVOKE. Но при наличии делегированных привилегий этот оператор не будет выполнен. Так, например, операция:

```
REVOKE ALL PRIVILEGES
ON Tabl
TO user4 RESTRICT
```

не будет выполнена, потому что пользователь user4 передал часть своих полномочий пользователю user5.

Посредством оператора REVOKE можно отобрать все или только некоторые из ранее присвоенных привилегий по работе с конкретным объектом. При этом из описания синтаксиса оператора отмены привилегий видно, что можно отобрать привилегии одним оператором сразу у нескольких пользователей или у целой группы PUBLIC.

Поэтому корректным будет следующее использование оператора REVOKE:

```
REVOKE INSERT
ON Tab1
TO user2,user4 CASCADE
```

При работе с другими объектами изменяется список операций, которые используются в операторах GRANT и REVOKE.

По умолчанию действие, соответствующее запуску (исполнению) хранимой процедуры, назначается всем членам группы PUBLIC.

Если вы хотите изменить это условие, то после создания хранимой процедуры необходимо записать оператор REVOKE. REVOKE EXECUTE ON COUNT_EX TO PUBLIC CASCADE И теперь мы можем назначить новые права пользователю user4.

```
GRANT EXECUTE
ON COUNT_EX
TO user4
```

Системный администратор может разрешить некоторому пользователю создавать и изменять таблицы в некоторой БД. Тогда он может записать оператор предоставления прав следующим образом:

```
GRANT CREATE TABLE, ALTER TABLE.
DROP TABLE ON DB_LIB TO userl
```

В этом случае пользователь userl может создавать, изменять или удалять таблицы в БД DB_LIB, однако он не может разрешить создавать или изменять таблицы в этой БД другим пользователям, потому что ему дано разрешение без права делегирования своих возможностей.

В некоторых СУБД пользователь может получить права создавать БД. Например, в MS SQL Server системный администратор может предоставить пользователю main_user право на создание своей БД на данном сервере. Это может быть сделано следующей командой:

```
GRANT CREATE DATABASE
ON SERVER_0
TO main user
```

По принципу иерархии пользователь main_user, создав свою БД, теперь может предоставить права на создание или изменение любых объектов в этой БД другим пользователям.

В СУБД, которые поддерживают однобазовую архитектуру, такие разрешения недопустимы. Например, в СУБД Oracle на сервере создается только одна БД, но пользователи могут работать па уровне подсхемы (части таблиц БД и связанных с ними объектов). Поэтому там вводится понятие системных привилегий. Их очень много, 80 различных привилегий.

Для того чтобы разрешить пользователю создавать объекты внутри этой БД, используется понятие системной привилегии, которая может быть назначена одному или нескольким пользователям. Они выдаются только на действия и конкретный тип объекта. Поэтому, если вы, как системный администратор, предоставили пользователю право создания таблиц (CREATE TABLE), то для того чтобы он мог создать триггер для таблицы, ему необходимо предоставить еще одну системную привилегию CREATE TRIGGER. Система защиты в Oracle считается одной из самых мощных, по это имеет и обратную сторону — она весьма сложная. Поэтому задача администрирования в Oracle требует хорошего знания как семантики принципов поддержки прав доступа, так и физической реализации этих возможностей.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ №2

1. В условиях Практической работы №1организовать защиту БД по конкретной области на уровне администратора и пользователя.

Практическая работа №3

СТРУКТУРА И СИНТАКСИС ЗАПРОСОВ

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Создание запросов на языке SQL

Запрос — объект базы данных, используемый для выборки или модификации хранимых данных.

Запрос на выборку является наиболее часто используемым типом запроса. Запросы этого типа выбирают данные из одной или нескольких таблиц и отображают их в виде таблицы, записи в которой можно обновлять (с некоторыми ограничениями). Запросы на выборку можно также использовать для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений.

Для подготовки запросов используются:

QBE (Query By Example) — язык запросов по образцам,

SQL (Structured Query Language) — язык структурированных запросов.

Sql- язык структурированных запросов

Язык структурированных запросов SQL является наиболее распространенным языком управления базами данных клиент/сервер. содержит операторы:

-описания данных (DDL — Data Definition Language). Основные операторы DDL — Create Domain (Создание домена), Alter Domain (Изменение домена), Drop Domain (Уничтожение домена), Create Table, Alter Table, Table Drop;

-управления данными (DML — Data Manipulation Language). Основные операторы DML — Select (Выбор), Insert (Вставка), Update (Обновление), Delete (Удаление);

-формирования запросов.

Ключевые слова SQL

1. Команды – представляют собой глаголы определяющего действия, которые необходимо выполнить (SELECT, ALTER, CREATE, DROP)

2. Условия (квалификатор)-ограничивают диапазон значений элементов, входящих в запрос (WHERE)
3. Модификаторы (предложения)-модифицируют выполнение инструкций (ORDER BY)
4. Предикаты – представляют собой выражения, такие как IN. Могут возвращать в качестве результата значения True, False, в некоторых случаях Null (неизвестный результат)
5. Операторы (=, <,>)- сравнивают значения или применяются для создания объединений в синтаксисе предложений WHERE. Эти операторы наз предикатами сравнения.
6. Статистические функции (агрегаты)- возвращают одно результирующее значение на основе набора данных (SUM (сумма), COUNT (количество), MIN (минимальное значение), MAX (максимальное значение) или AVG (среднее значение)
7. Функции преобразования типа данных – изменяют тип данных с одного на др.(CAST, CONVERT)
8. Другие ключевые слова (зарезервированные)-изменяющие действие команд или управляющие курсором (указателем текущей записи в наборе)

В синтаксических конструкциях используются следующие обозначения:

звездочка (*) для обозначения "все" - употребляется в обычном для программирования смысле, т.е. "все случаи, удовлетворяющие определению";

квадратные скобки ([]) – означают, что конструкции, заключенные в эти скобки, являются необязательными (т.е. могут быть опущены);

фигурные скобки ({}) – означают, что конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы, т.е. они позволяют уточнить порядок разбора синтаксических конструкций, заменяя обычные скобки, используемые в синтаксисе SQL;

многоточие (...) – указывает на то, что непосредственно предшествующая ему синтаксическая единица факультативно может повторяться один или более раз;

прямая черта (|) – означает наличие выбора из двух или более возможностей. Например обозначение ASC|DESC указывает,

можно выбрать один из терминов ASC или DESC; когда же один из элементов выбора заключен в квадратные скобки, то это означает, что он выбирается по умолчанию (так, [ASC]|DESC означает, что отсутствие всей этой конструкции будет восприниматься как выбор ASC);

точка с запятой (;) – завершающий элемент предложений SQL;

запятая (,) – используется для разделения элементов списков;

пробелы ( ) – могут вводиться для повышения наглядности между любыми синтаксическими конструкциями предложений SQL;

прописные жирные латинские буквы и символы – используются для написания конструкций языка SQL и должны (если это специально не оговорено) записываться в точности так, как показано;

строчные буквы – используются для написания конструкций, которые должны заменяться конкретными значениями, выбранными пользователем, причем для определенности отдельные слова этих конструкций связываются между собой символом подчеркивания (_);

Основой SQL является инструкция SELECT, используемая для создания запросов на выборку.

Синтаксис инструкции:

SELECT [ALL │ DISTINCT │DISTINCTROW] список_выбора

[AS псевдоним 1[, псевд 2 [,…]]]

FROM имена таблиц

[WHERE критерий поиска]

[GROUP BY имя столбца, имя столбца,…]

[HAVING условие поиска]

[ORDER BY критерий столбца [ASC │ DESC]];

SELECT — выбрать (директива) данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями

FROM — из (условие) перечисленных таблиц, в которых расположены эти столбцы

WHERE — где (условие) строки из указанных таблиц должны удовлетворять указанному перечню условий отбора строк

GROUP BY — группируя по (условие) указанному перечню столбцов с тем, чтобы получить для каждой группы единственное агрегированное значение, используя во фразе SELECT SQL – функции: SUM (сумма), COUNT (количество), MIN (минимум), MAX (максимум), AVG (среднее значение)

HAVING — имея в результате лишь те группы, которые удовлетворяют указанному перечню условий отбора групп (условие)

ORDER BY — спецификация сортировки (условие) определяет порядок сортировки: ASC – сортировка по возрастанию, DESC - сортировка по убыванию.

ПРЕДИКАТЫ :

1. Сравнения =, <>, >=, <, <=
2. В интервале - “между” BETWEEN a1 and a2
3. Входит в множество IN (=[Товар] IN (“Мука”,“Крупа”…..)
4. Подобие < имя > Like < образец >

(что) (с чем сравнивать)

Все MySQL запросы поделены на простые и сложные запросы.

Простые mysql запросы – запросы в которых участвует одна таблица базы данных.

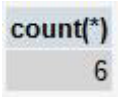
Сложные mysql запросы – запросы в которых могут участвовать две и более таблиц БД.

Простые mysql запросы

Select запросы

Слово select, говорит само за себя, и становится понятно, что пользуясь данными запросами, мы будем выбирать (читать) информацию из БД.

SELECT count(*) FROM table_name;



Выведет количество всех записей в таблице

SELECT * FROM table_name;

Выбирает все записи из таблицы БД

id	site	description
1	sitear.ru	SiteAR
2	sitear.ru	SiteAR
3	yaveterinar.ru	Ветеринария
4	wi-korporaciya.ru	О корпорации
5	sitear.ru	пример 1
6	sitear.ru	пример 2

SELECT * FROM person ORDER BY number;

Выберет все записи из таблицы person в порядке возрастания значений поля number.

number	name	last_name	age
1	Anna	Moroz	12
2	Anka	Moroz	15
3	Anna	Cool	16
4	Anko	Second	18
5	Polina	First	13
6	Polianna	Second	18
7	Vanna	Third	9
8	Anno	Wow	10

SELECT * FROM person WHERE name='Anna';

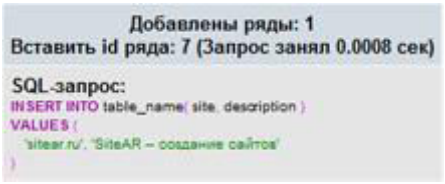
Выбирает все записи из таблицы person, где поле name соответствует значению Anna.

number	name	last_name	age
1	Anna	Moroz	12
3	Anna	Cool	16

Insert запросы

Данные запросы позволяют вставить запись в таблицу БД. Другими словами создать строку в таблице или добавить информацию в таблицу БД.

insert into table_name(site, description) values ('sitear.ru', 'SiteAR – создание сайтов')



Вставит в таблицу table_name, а точнее в поля site и description данной таблицы, соответствующие значения.

Update запросы

Направлены на изменение уже имеющихся данных в таблице БД.

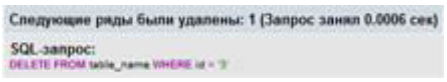
update table_name set site = 'domain.com' where id = '3'

Изменяет значение поля site на domain.com в таблице table_name где id равен 3.

Delete запросы

Удаляют записи из таблицы БД.

delete from table_name where id = '3'



Удаляет запись из table_name где id равен 3.

Сложные mysql запросы

Как уже упоминалось раньше, сложные mysql запросы, работают более, нежели с одной таблицей БД. Данные mysql запросы, мы будем рассматривать более в индивидуальном порядке, так как они сложные и их будет немного.

SELECT DISTINCT last_name FROM person, address WHERE person.adress_no = address.address_no AND city LIKE 'L%';

или

SELECT DISTINCT last_name FROM person p, address adr WHERE p.adress_no = adr.address_no AND city LIKE 'L%';

Выводит все уникальные фамилии людей (last_name), которые живут в городе с названием на букву L. (предполагаем, что в таблице address есть поля address_no, city).

Данные примеры сложных mysql запросов, выведут один и тот же результат. Запросы не очень то и сложные, нужно только указать имя таблицы БД, а потом, через точку указать поле таблицы. Или же можно, как во втором примере, дать короткие имена таблицам (p для person, adr для address). Результат запросов будет один и тот же.

SELECT heroes.char_name, heroes.count, char_templates.ClassName FROM char_templates, heroes WHERE char_templates.ClassId = heroes.class_id Order by char_templates.ClassName;

или

SELECT char_name, count, ClassName FROM heroes left join char_templates on heroes.class_id=char_templates.ClassId;

Берем из таблицы heroes поле char_name, из heroes поле count, из таблицы char_templates поле ClassName, где char_templates.ClassId и heroes.class_id имеют общий идентификатор и сортируем запрос по имени класса героев.

Таким же образом, можно подавать сложные mysql запросы с помощью update, insert, delete и др.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ №3

1. Заполнить данными таблицы БД.
2. Разработать 2-3 запроса на основании данных своей предметной области.

Практическая работа №4

РЕЗЕРВНОЕ КОПИРОВАНИЕ РБД

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Одна из главных обязанностей администратора баз данных – быть готовым к программным, аппаратным сбоям, нарушениям целостности данных, а также, уметь восстановить базы данных в случае аварии. В случае возникновения сбоя, главное – восстановить доступ пользователей к базе данных в течение приемлемого времени с минимальными потерями данных.

Чтобы избежать потери данных, можно реализовать для базы данных стратегию восстановления. Стратегию восстановления необходимо спланировать, реализовать и протестировать с учетом возможных неисправностей, с которыми можно встретиться в процессе работы системы, и необходимого уровня защиты данных. В витринах данных, то есть в случаях, когда данные можно восстановить из других систем, вероятно, нет необходимости создавать резервные копии каждой отдельной транзакции. Возможно, будет достаточно выполнять полное резервное копирование данных с регулярными временными интервалами. И наоборот, для базы данных, в которой хранятся транзакции интернет-магазина, возможно, будет необходимо сохранять резервные копии каждой отдельной транзакции. СУБД SQL Server предоставляет полный комплекс функций для реализации именно того вида резервного копирования, который вам необходим.

Полное резервное копирование базы данных

Самой распространенной стратегией резервного копирования является резервное копирование всей базы данных через заранее заданные промежутки времени (например, каждую ночь). Благодаря такой стратегии аварийного восстановления можно восстановить базу данных до состояния на момент выполнения последнего резервного копирования.

Полная резервная копия базы данных содержит все данные и метаинформацию базы данных, которые необходимы для восстановления базы данных полностью, включая полнотекстовые

каталоги. При восстановлении базы данных из полной резервной копии восстанавливаются все файлы базы данных, причем данные извлекаются в непротиворечивом состоянии на тот момент времени, в который выполнялось резервное копирование. Пока выполняется резервное копирование, база данных работает в рабочем режиме, и пользователь может выполнять транзакции, изменяя данные обычным путем. Термин "непротиворечивое состояние" означает, что все транзакции, которые были зафиксированы в процессе выполнения резервного копирования базы данных, применяются, а все транзакции, которые не были завершены, подвергаются откату. Для ситуаций, которые могли бы привести к нарушению непротиворечивости данных вследствие выполнения транзакций, изменяющих данные в процессе выполнения резервного копирования, в SQL Server есть особый процесс, который позволяет гарантировать непротиворечивость данных. Этот процесс выполняет запись на устройство резервного копирования как страниц данных, так и журнала транзакций.

Скорость резервного копирования определяется скоростью используемых устройств ввода/вывода (тех устройств ввода вывода, которые используются для сбора и хранения информации). Чтобы добиться наилучшей производительности, SQL Server считывает файлы последовательно. Если ваши устройства ввода/вывода способны одновременно обрабатывать данные ввода/вывода резервного копирования и данные ввода/вывода, поступающие в результате обычного использования системы, то создание резервной копии окажет на производительность системы незначительное воздействие. Тем не менее, лучше выполнять полное резервное копирование базы данных при отсутствии пиковых нагрузок.

Простая модель восстановления

Следует заранее уведомить SQL Server о том, какой тип резервного копирования вы намерены использовать, поэтому надо сконфигурировать базу данных так, чтобы настройки соответствовали выбранному вами типу резервного копирования. Такая настройка выполняется посредством выбора значения параметра "модель восстановления базы данных". Модель восстановления базы данных, которая используется по умолчанию,

является производным от модели восстановления модели базы данных, определенной при ее создании. Чтобы реализовать стратегию резервного копирования, которая будет включать только полные резервные копии, следует выбрать простую модель восстановления (SIMPLE).

Выбираем модель восстановления SIMPLE

1. В меню Start (Пуск) выберите All Programs,. Microsoft SQL Server 2005, SQL Server Management Studio (Все программы, Microsoft SQL Server 2005, Среда SQL Server Management Studio).
2. В диалоговом окне Connect To Server (Соединение с сервером) нажмите кнопку Connect (Соединить).
3. В панели инструментов Standard (Стандартная) нажмите кнопку New Query (Новый запрос), чтобы открыть окно New Query (Новый запрос).
4. Чтобы задать модель восстановления, можно использовать инструкцию ALTER DATABASE. Введите текст следующей инструкции и нажмите кнопку Execute (Выполнить).

```
USE master;
GO
ALTER DATABASE AdventureWorks
SET RECOVERY SIMPLE;
GO
```

Проверяем настройки модели аварийного восстановления

Чтобы просмотреть заданную для базы данных модель восстановления, можно использовать функцию DATABASEPROPERTYEX, которая извлекает параметры текущей базы даты или свойства указанной базы данных. Выполните инструкцию, приведенную ниже, чтобы извлечь информацию о модели восстановления базы данных AdventureWorks.

```
SELECT
DATABASEPROPERTYEX('AdventureWorks','Recovery')
```

Убедитесь, что в результатах запроса указана модель восстановления SIMPLE.

Закройте окно среды SQL Server Management Studio.

Устройства резервного копирования

До начала выполнения операций резервного копирования необходимо определить, где будут храниться резервные копии. Место хранения резервных копий называется устройством резервного копирования. Каждое устройство резервного копирования может хранить несколько резервных копий разных типов. Существует два разных вида устройств резервного копирования:

1. Ленточные устройства.Могут использоваться для хранения резервных копий на лентах. Ленточные устройства должны быть установлены локально. Резервная копия может занимать несколько лент, а на одной ленте могут находиться одновременно резервные копии SQL Server и Windows.
2. Дисковые устройства.Файлы на локальном или удаленном диске или дисковом накопителе. К этим файлам обращаются, указывая путь к файлу, в котором хранится резервная копия. Для обращения к удаленным хранилищам следует использовать путь в формате UNC.

Выполнение полного резервного копирования базы данных

После того, как вы установили модель резервного копирования на SIMPLE и решили, на каком устройстве резервного копирования сохранять резервные копии, можно приступить к выполнению резервного копирования. Полное резервное копирование базы данных выполняется с помощью довольно простой инструкции BACKUP DATABASE. В простейшей форме нужно только указать системе, резервную копию какой базы данных создать и на каком устройстве ее сохранить. Чтобы создать резервную копию базы данных AdventureWorks на предварительно выбранном логическом устройстве, используется следующая инструкция T-SQL:

```
USE master;
GO
BACKUP DATABASE AdventureWorks
TO Adv_FullDb_Dev;
```

Если надо выполнить полное резервное копирование базы данных на физическое устройство, необходимо указать тип устройства и место размещения резервной копии в инструкции BACKUP DATABASE. Чтобы создать резервную копию базы данных в папке t:\adv.*bak*, используйте следующую инструкцию:

```
USE master;
GO
BACKUP DATABASE AdventureWorks
TO DISK='t:\adv.bak';
```

Как уже говорилось, на любом устройстве резервного копирования может храниться больше одной резервной копии. В аргументе инструкции BACKUP DATABASE можно указать, следует ли SQL Server перезаписать существующую резервную копию на этом устройстве или дописать ее к существующим резервным копиям. Для этого используются ключевые слова INIT или NOINIT. Если указать INIT, то перед запуском резервного копирования устройство резервного копирования форматируется, при этом удаляются все резервные копии, которые имеются на этом устройстве. NOINIT, которое используется по умолчанию, если не указано иное, разрешает SQL Server дописать резервную копию на существующее устройство резервного копирования с сохранением всех существующих на нем резервных копий. Эти опции устанавливаются при помощи блока WITH в конце инструкции BACKUP DATABASE.

Если нужно создать такую же резервную копию, как в предыдущем примере, но при этом указать SQL Server сначала очистить устройство, используйте следующую инструкцию:

```
USE master;
GO
BACKUP DATABASE AdventureWorks
TO DISK='t:\adv.bak'
WITH INIT;
```

Как видите, выполнение полного резервного копирования базы данных не отличается сложностью. В следующем разделе вы увидите, что полная резервная копия - это основной тип резервного копирования, на котором строятся все остальные типы резервного копирования. Остальные типы резервного копирования зависят от наличия полной резервной копии, потому что им необходима восстановленная база данных в качестве исходной точки. Эти типы резервного копирования, в том числе, разностное резервное копирование, сохраняют изменения, которые были внесены в базу данных после создания полной резервной копии. Таким образом, мы видим, что полные резервные копии важны не только в стратегии восстановления, при которой выполняется только полное резервное копирование, но и в других стратегиях резервного копирования, о которых речь пойдет ниже.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ №4

1. Осуществить полное резервное копирование базы данных.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Уроки SQL и баз данных/ URL: http://www.site-do.ru/db/db.php

2. Как создать базу данных в MS SQL Server. URL: http://npk-kaluga.ru/CreateBase_MSSQL.htm

3. Как создать новую таблицу в базе данных MS SQL Server. URL: http://npk-kaluga.ru/CreateTable_MSSQL.htm

4. Создание новой диаграммы базы данных (визуальные инструменты для баз данных). URL: http://msdn.microsoft.com/ru-ru/library/ms189078.aspx

5. Создание базы данных. URL: http://technet.microsoft.com/ru- ru/library/ms186312(v=sql.110).aspx

6. [Электронный ресурс]. URL: http://www.osp.ru/win2000/2013/05/13035359/#list2

7. [Электронный ресурс]. URL: http://dbasimple.blogspot.ru/2013/08/ms-sql-server.html

8. [Электронный ресурс]. URL: http://rutube.ru/video/fa041b99e956c1534a6a424e2a9aac57/

9. [Электронный ресурс]. URL: http://msdn.microsoft.com/ru-ru/library/bb510663.aspx

10. Все о SQL и клиент/серверных технологиях [электронныйресурс]. URL: http://sql.ru

11. Администрирование баз данных [электронный ресурс] URL: http://www.firebirdsql.org/manual/ru/migration-mssql-db-admin-ru.html

12. Прозрачное шифрование баз данных в Microsoft SQL Server 2008 [электронный ресурс]. URL: http://rsdn.ru/article/db/liberman.xml

13. Шифрование в базах данных SQL Server [электронныйресурс].URL: http://www.itshop.ru/Shifrovanie-v-bazah-dannyh-SQL- Server/l9i36233