Встроенные механизмы защиты информации в системах управления базами данных

Домашнее задание :

Копирование и перенос данных. Восстановление данных

План:

1.Создание резервных копий всей базы данных, журнала транзакций, а также одного или нескольких файлов или файловых групп.

2.Параллелизм операций модификации данных и копирования.

3.Типы резервного копирования. Использование зеркальных наборов носителей резервных копий. Управление резервными копиями.

4.Автоматизация процессов копирования. 5. Восстановление данных

Системы управления базами данных, в особенности реляционные СУБД, стали доминирующим инструментом хранения больших массивов информации. Сколько-нибудь развитые информационные приложения полагаются не на файловые структуры операционных систем, а на многопользовательские СУБД, выполненные в технологии клиент/сервер.

В этой связи обеспечение информационной безопасности СУБД, и в первую очередь их серверных компонентов, приобретает решающее значение для безопасности организации в целом.

Идентификация и проверка подлинности пользователей

SQL Server 2005 поддерживает два режима аутентификации:

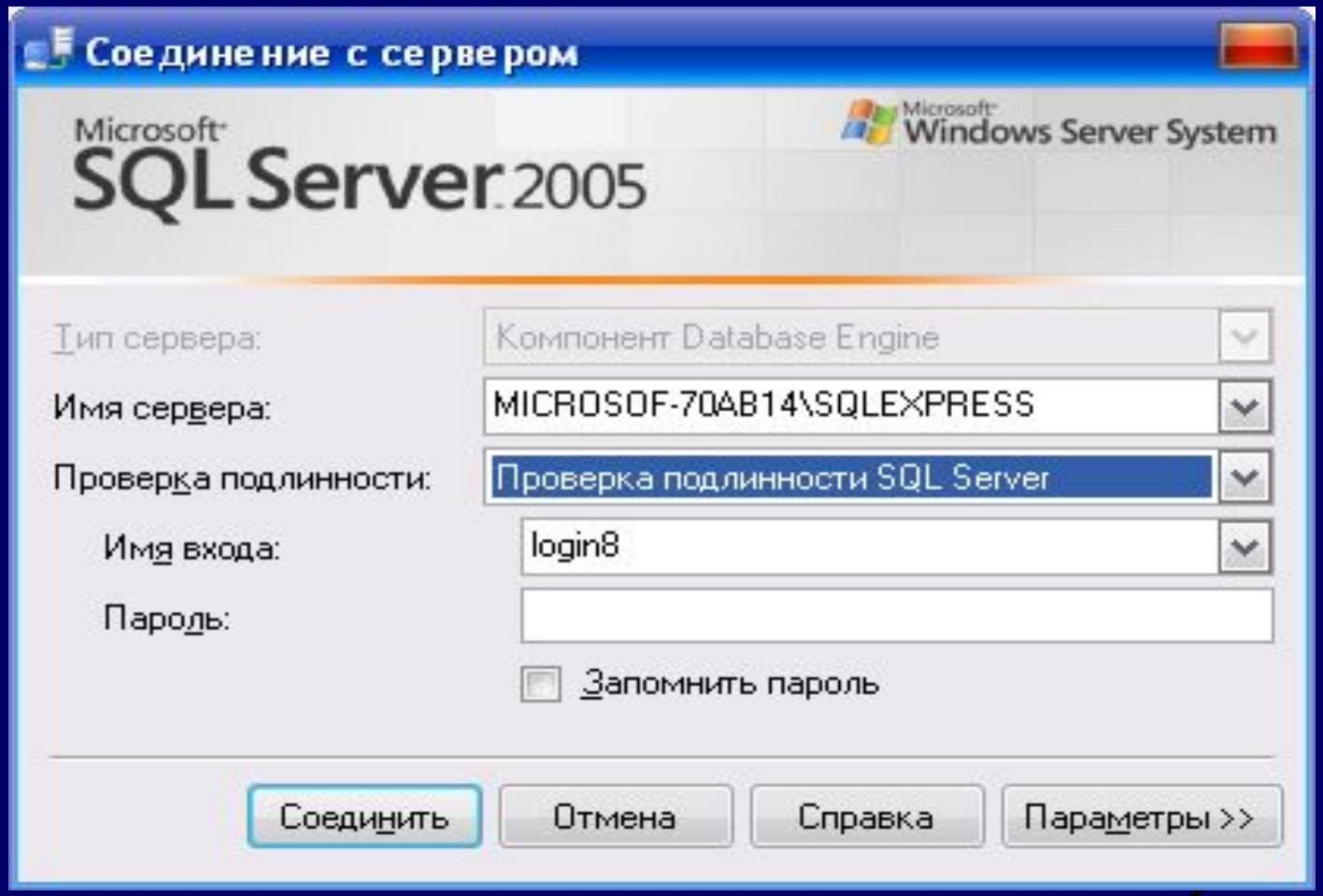
с помощью Windows и с помощью SQL Server.

Первый режим позволяет реализовать решение, основанное на однократной регистрации пользователя и едином пароле при доступе к различным приложениям

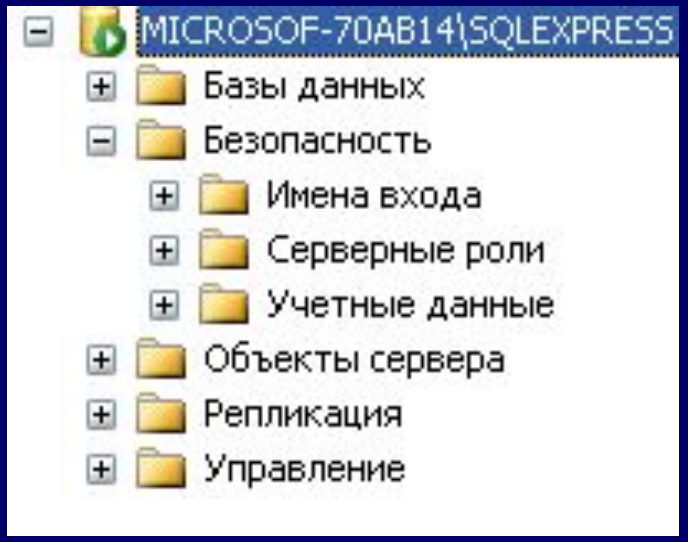
Идентификация и проверка подлинности пользователей

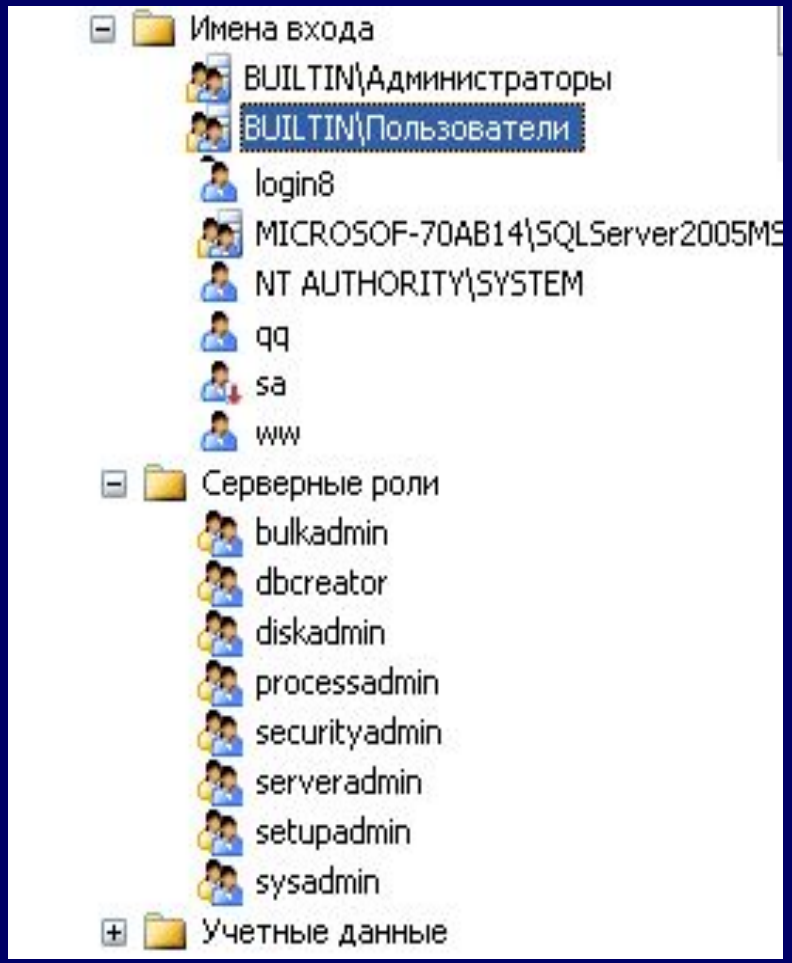
Аутентификация с помощью SQL Server предназначена главным образом для клиентских приложений, функционирующих на платформах, отличных от Windows. Этот способ считается менее безопасным

Идентификация и проверка подлинности пользователей

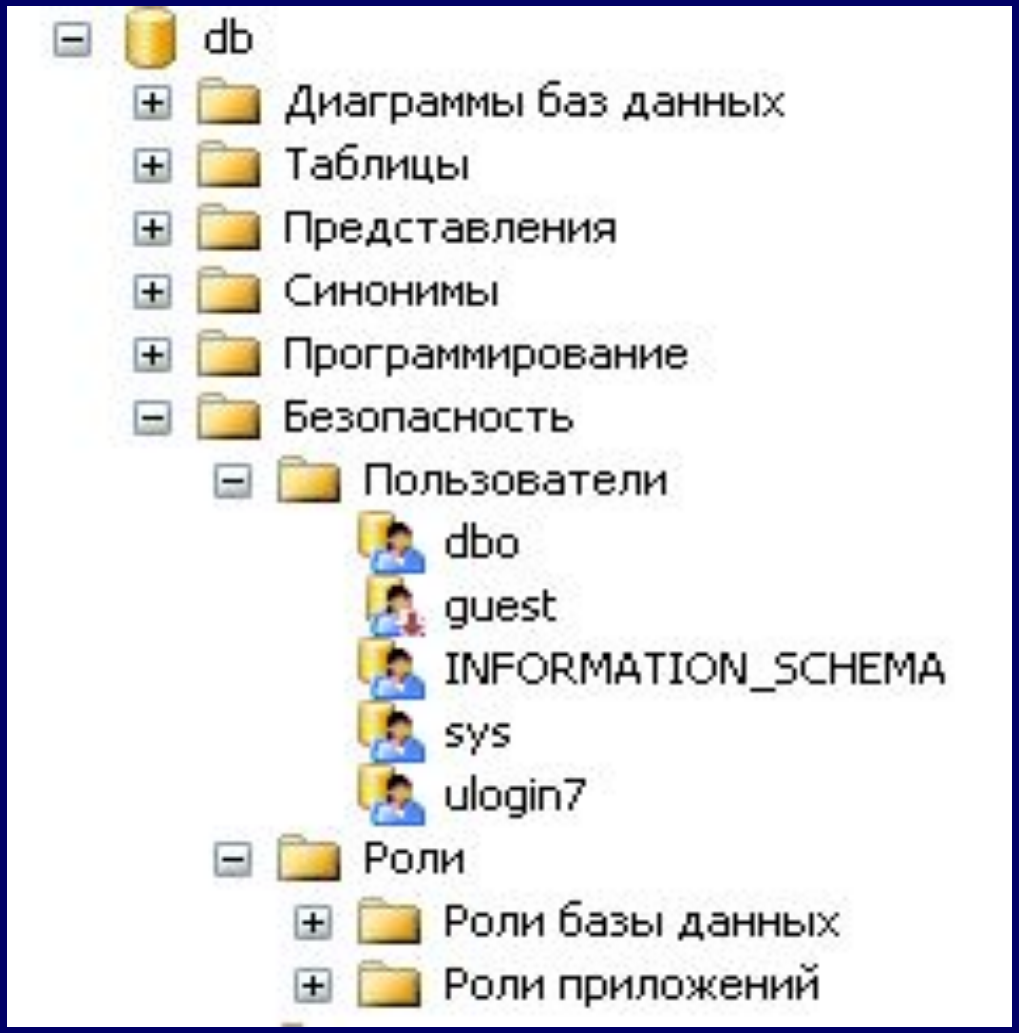


Политика безопасности





Политика безопасности



Идентификация и проверка подлинности пользователей

1. Установите выбранный режим проверки подлинности;
2. Перезапустите сервис MSSQLServer;
3. Заведите в вашем домене группу клиентов, которые будут работать с SQLS2005;
4. С помощью SQL SEM предоставьте пользователям или группам пользователей доступ к SQLS7;
5. Для не Windows клиентов создайте с помощью SQL SEM собственные логины, указав им язык по умолчанию.

Создание пользователей

В системе SQL-сервер организована двухуровневая настройка ограничения доступа к данным.

1. Создать учетную запись пользователя (login), что позволит подключиться к самому серверу, но не даст автоматического доступа к базам данных.

2. Для каждой базы данных (на основании учетной записи) создать запись пользователя

Создание пользователей

```
create login log8 with password = '1234', default_database = db
go
create user ulog8 for login log8
```

В разных БД login одного и того же пользователя может иметь одинаковые или разные имена user с разными правами доступа

Создание пользователей

Создание новой учетной записи может быть выполнено с помощью системной процедуры:

```
sp_addlogin
[@login=] 'учетная_запись'
[, [@password=] 'пароль']
[, [@defdb=] 'база_данных_по_умолчанию']
```

Создание пользователей

Доступ к конкретной базе банных - с помощью системной процедуры:

```
sp_adduser
[@loginame=] 'учетная_запись'
[, [@name_in_db=] 'имя_пользователя']
[, [@grpname=] 'имя_роли']
```

Создание пользователей

Отобразить учетную запись Windows NT в имя пользователя позволяет хранимая процедура

```
sp_grantdbaccess
[@login=] 'учетная_запись'
[, [@name_in_db=]'имя_пользователя']
```

Роли

Роль позволяет объединить в одну группу пользователей, выполняющих одинаковые функции.

В SQL Server реализовано два вида стандартных ролей: на уровне сервера и на уровне баз данных.

Например,

sysadmin с правом выполненя любых функций

db_owner с правом полного доступа к базе данных

Действия по отношению к роли

Создание новой роли

```
sp_addrole
[@rolename=] 'имя_роли'
[, [@ownername=] 'имя_владельца']
```

Добавление пользователя к роли

```
sp_addrolemember
[@rolename=] 'имя_роли',
[@membername=] 'имя_пользователя'
```

Действия по отношению к роли

Удаление пользователя из роли

```
sp_droprolemember
[@rolename=] 'имя_роли',
[@membername=] 'имя_пользователя'
```

Удаление роли

```
sp_droprole
[@rolename=] 'имя_роли'
```

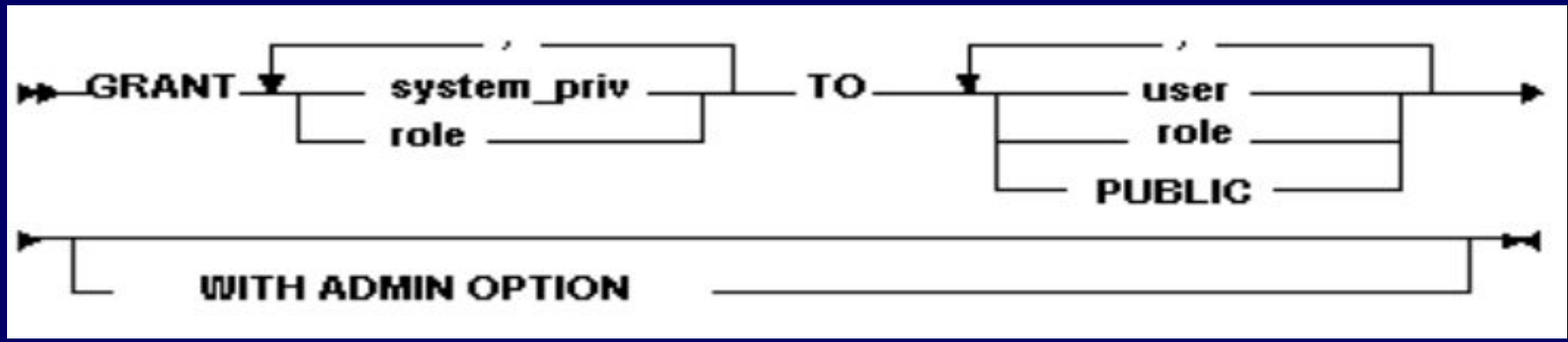
Команды управления данными (DCL).

С помощью них можно управлять доступом пользователей к базе данных.

Операторы управления данными включают:

- применяемые для предоставления и отмены полномочий команды **GRANT** и **REVOKE**;
- команду **SET ROLE**, которая разрешает или запрещает роли для текущего сеанса.

Синтаксис команды GRANT:



system_priv — системная привилегия.
role — роль: набор соответствующих полномочий, которые администратор может коллективно предоставлять пользователям и другим ролям.
user — пользователь.
PUBLIC — привилегия передается всем пользователям.
WITH ADMIN OPTION — если предоставлены системные полномочия или роли, то параметр позволяет пользователю передать полномочие или роль другим пользователям или ролям.

В системах **клиент/сервер** доступ к базе данных могут получить только пользователи, за**регистрированные** в системе.

Команда SQL **GRANT** используется для предоставления пользователю **роли** или **полномочия**.

Эту привилегию имеет только **администратор** базы данных.

Привилегии:

- **DBA** разрешает пользователю выполнять действия администратора базы данных. Имея привилегию **DBA,** пользователь может выполнять команду **SELECT** для любой таблицы и представления, создавать объекты для других пользователей, предоставлять другим пользователям различные привилегии, выполнять полный экспорт/импорт базы данных.
- **RESOURCE** разрешает пользователю создавать объекты базы данных, включая таблицы и индексы.
- **CONNECT** позволяет пользователю подключаться к базе данных и работать с объектами, к которым он имеет привилегии по доступу. Пользователь может создавать представления, синонимы и межтабличные связи.

Например:

GRANT DBA TO SYSADM IDENTIFIED BY SYSTEM

Можно использовать предложение **GRANT** для спецификации предоставляемого полномочия, роли или разделяемого запятыми списка полномочий либо ролей. Для указания целевого пользователя или роли применяется предложение **TO.** Например:

GRANT CREATE TABLE, CREATE VIEW Ivanov, Petrov
GRANT place1, update1, delete1 TO role77
GRANT SELECT ON Zakazchik to Sidorov

В **первом** примере системные полномочия **CREATE TABLE** и **CREATE VIEW** предоставляются пользователям **Ivanov** и **Petrov**.

Второй оператор предоставляет роли **place1, update1** и **delete1** другой роли -- **role77.**

Третий оператор предоставляет пользователю **Sidorov** полномочия **SELECT** на таблицу **Zakazchik**.

Если необходимо предоставить кому-то все полномочия на конкретный объект, используется ключевое слово **ALL:**

GRANT ALL ON Zakazchik TO Rodionov

Можно предоставлять системные полномочия или роли с параметром **ADMIN**, что позволяет пользователю передать полномочие или роль другим пользователям или ролям:

GRANT CREATE TABLE TO Medvedev WITH ADMIN OPTION

После этого **Medvedev** может предоставлять полномочия **CREATE TABLE** другим пользователям и ролям:

GRANT CREATE TABLE TO Krasnov

Если пользователю предоставляются полномочия на объект, то параметр **GRANT** позволяет передать эту возможность предоставления полномочий другим пользователями и ролям:

GRANT SELECT, INSERT, UPDATE, DELETE ON Zakazchik TO Dudkin WITH GRANT OPTION

Если таблица должна быть доступна всем пользователям, указывается вместо имени пользователя ключевое слово **PUBLIC.**

Отмена привилегий осуществляется с помощью команды **REVOKE,** синтаксис которой аналогичен синтаксису команды **GRANT.** Можно отменить общие привилегии пользователя или же привилегии на указанную таблицу. Например:

REVOKE CREATE TABLE FROM Ivanov

REVOKE place1 FROM role77

REVOKE SELECT ON Zakazchik FROM Sidirov

С помощью команды **REVOKE** можно задать отменяемое полномочие или роль. Предложение **FROM** позволяет указать пользователя или роль, для которых отменяются полномочия. Ключевое слово **ALL** позволяет отменить все объектные полномочия:

REVOKE ALL ON Zakazchik FROM Sidorov

Команда SQL **SET ROLE** разрешает или запрещает роли в текущем сеансе. С помощью ролей администратор может значительно упростить управление полномочиями.

Идентификация и проверка подлинности пользователей

Обычно в СУБД для идентификации и проверки подлинности пользователей применяются либо соответствующие механизмы операционной системы, либо SQL-оператор CONNECT. Например, в случае СУБД Oracle оператор CONNECT имеет следующий вид:

CONNECT пользователь[/пароль] [@база_данных];

Так или иначе, в момент начала сеанса работы с сервером баз данных, пользователь идентифицируется своим именем, а средством аутентификации служит пароль. Детали этого процесса определяются реализацией клиентской части приложения.