

The logo for SQL Server 2005 features a blue and green wave-like background. On the left, there are two spheres: a blue one and a green one. The text "SQL SERVER 2005" is displayed in a stylized, blue, serif font with a slight shadow effect.

SQL SERVER 2005

数据库系统概论

第四章 数据库的安全性

第4章 数据库安全性

- ❖ 第一节 计算机安全性概论(了解)
- ❖ 第二节 数据库安全性控制
- ❖ 第三节 视图机制
- ❖ 第四节 审计
- ❖ 第五节 数据加密
- ❖ 第六节 统计数据库安全性



教学目标

❖ 掌握

- 身份验证、存取控制、角色管理

❖ 了解

- 安全标准、强制存取控制、审计、数据加密

❖ 重点

- 存取控制、角色管理

❖ 难点

- 角色管理

本讲内容

❖ 第一节 计算机安全性概论(了解)

● 第二节 数据库安全性控制

❖ 第三节 视图机制

❖ 第四节 审计

❖ 第五节 数据加密

❖ 第六节 统计数据库安全性



数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

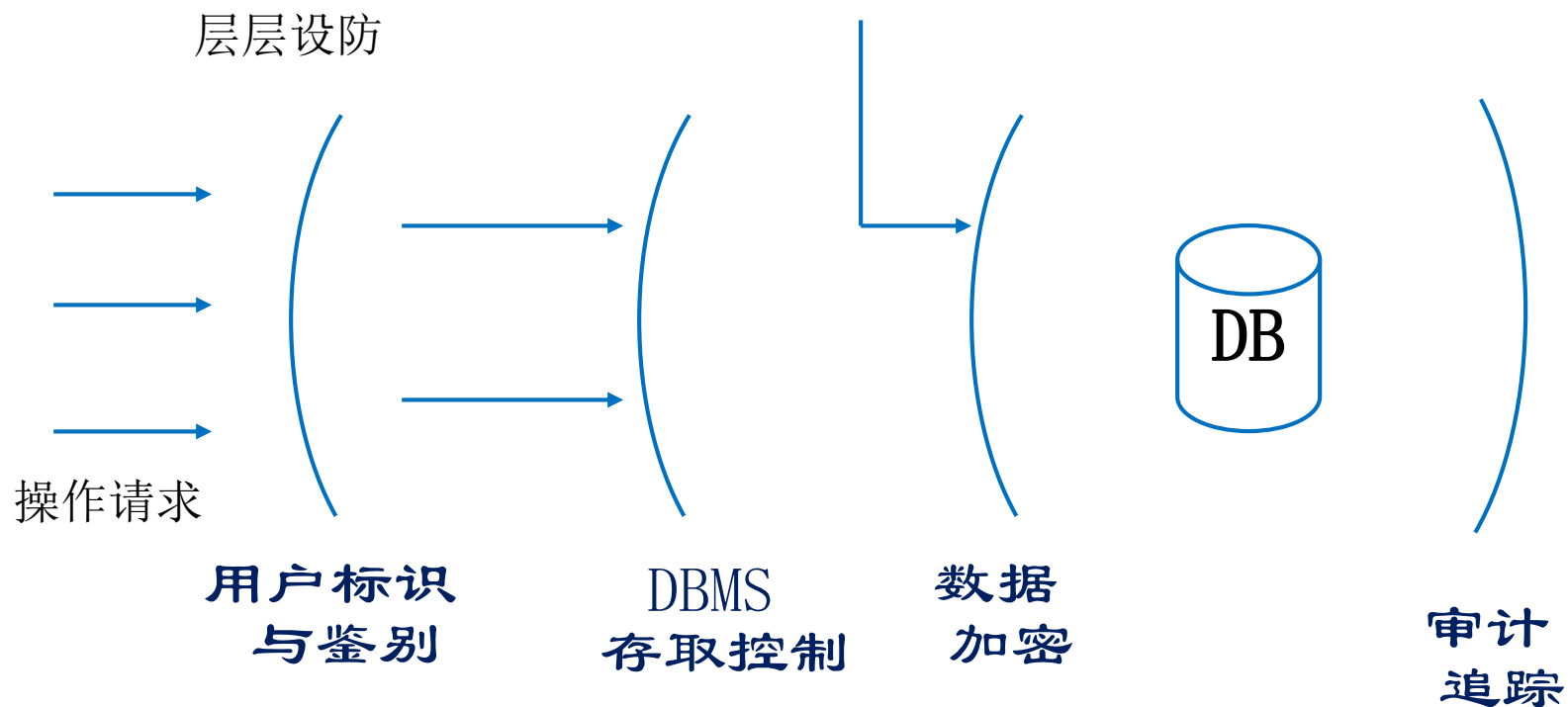
数据库安全性控制概述

❖ 非法使用数据库的情况

- 编写合法程序绕过DBMS及其授权机制
- 直接或编写应用程序执行非授权操作
- 通过多次合法查询数据库从中推导出一些保密数据
- 破坏安全性的行为可能是无意的，故意的，恶意的



数据库安全控制层次



❖ 数据库安全性控制的常用方法

- 用户标识和鉴定
- 存取控制
- 视图
- 审计
- 密码存储

数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

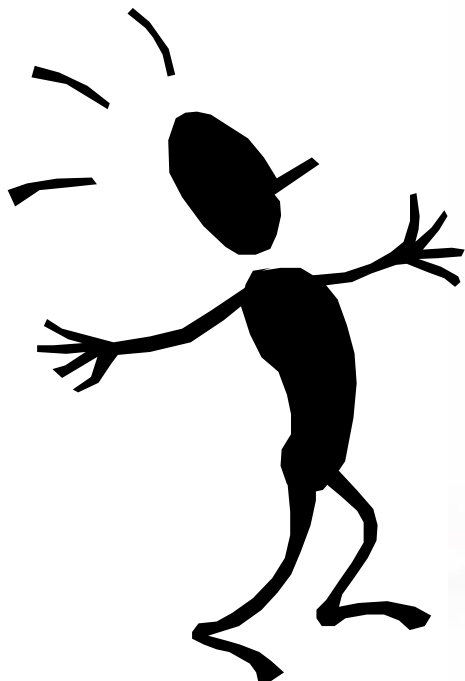
- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

用户标识与鉴别

❖ 用户标识与鉴别 (Identification & Authentication)

- 系统提供的最外层安全保护措施



用户标识自己的名字或身份

❖ 用户标识

❖ 口令

- 系统核对口令以鉴别用户身份

❖ 用户名和口令易被窃取

- 每个用户预先约定好一个计算过程或者函数

SQL Server 2008安全验证模式

当用户使用SQL SERVER2008时，需要经过两个安全性阶段，身份验证和权限认证阶段

❖ 身份验证阶段

- 用户在SQL SERVER2008上获得任何数据库访问权限之前，必须首先登录到SQL SERVER2008并且是合法的，否则服务器将拒绝用户登录

❖ 权限验证阶段

- 身份验证阶段只能验证用户是否具有连接到SQL SERVER2008的权限，通过身份验证后，需要验证用户是否具有访问服务器数据的权限，为此需要为每个数据库建立用户，并将账户映射到登录账户，并为用户分配对象的访问权限

- 1、**登录名**：服务器方的一个实体，使用一个登录名只能进入服务器，但是不能让用户访问服务器中的数据库资源。
- 2、**用户名**：一个或多个登录对象在数据库中的映射，可以对用户对象进行授权，以便为登录对象提供对数据库的访问权限。
- 3、**SQLSERVER把登录名与用户名的关系称为映射**。用登录名登录SQLSERVER后，在访问各个数据库时，SQLSERVER会自动查询此数据库中是否存在与此登录名关联的用户名，若存在就使用此用户的权限访问此数据库，若不存在就是用guest用户访问此数据库
- 4、**一个登录名可以被授权访问多个数据库，但一个登录名在每个数据库中只能映射一次**。即一个登录可对应多个用户，一个用户也可以被多个登录使用。好比SQLSERVER就象一栋大楼,里面的每个房间都是一个数据库.登录名只是进入大楼的钥匙,而用户名则是进入房间的钥匙.一个登录名可以有多个房间的钥匙，但一个登录名在一个房间只能拥有此房间的一把钥匙。



- 5、链接或登录Sql Server服务器时是用的登录名而非用户名登录的，程序里面的链接字符串中的用户名也是指登录名
- 6、我们常见的dbo（用户名）是指以sa(登录名)或windows administration(Windows集成验证登录方式)登录的用户,也就是说数据库管理员在SQLSERVER中的用户名就叫dbo,而不叫 sa
- 7、SQL Server中还有一个特殊的数据库角色public，它存在于每一个数据库中，包括系统数据库，如master、msdb、model和用户数据库，数据库的所有用户都属于public角色，并且不能从public角色中删除。

SQL Server 2008身份验证

❖ SQL Server 2008提供了两种确认用户对数据库引擎服务的验证模式：

- Windows身份验证

Windows验证模式允许SQL Server可以使用Windows的用户名和口令。在这种模式下，用户只需要通过Windows的验证，就可以连接到SQL Server，登录SQL Server时就不再需要输入帐户和密码了。

- SQL Server身份验证

SQL Server身份验证模式要求用户在连接SQL Server时必须提供登录名和登录密码，与Windows的登录帐号无关。SQL Server自身执行认证处理。利用这种方式可以很方便地从网络上访问sql server服务器。

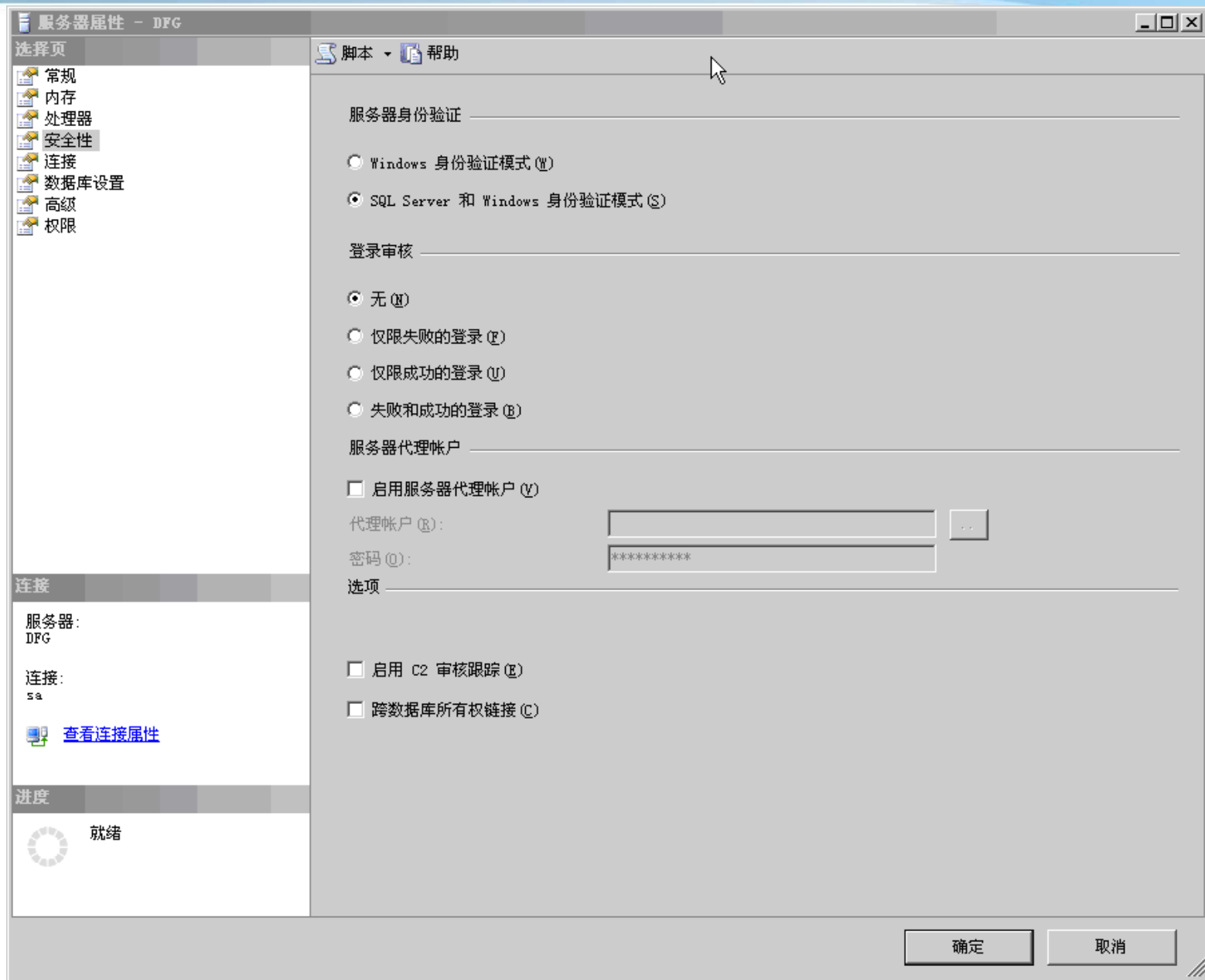


图1 SQL Server服务器属性标签

创建登录名

CREATE LOGIN <登录名>

[{

WITH PASSWORD = ' ' [HASHED][MUSTCHANGE]

,DEFAULT_DATABASE = <数据库>

} |

{

FROM

WINDOWS

[WITH DEFAULT_DATABASE = <数据库>]

|CERTIFICATE <证书名>

|ASYMMETRIC KEY <不对称密钥名>

}

]



create login dd

with password='123',default_database=SPJ;

create login [DESKTOP-U68UGF5\dinglei]

--必须加中括号，计算机名+用户名

from windows;



[例1] 创建一个sql server验证模式的登录名

```
CREATE LOGIN 张三  
WITH PASSWORD = 'abc123!'
```

[例2] 创建一个windows验证模式的登录名

```
CREATE LOGIN [win2k3\Administrator]  
FROM WINDOWS
```

创建SQL Server 2008数据库用户

❖ 用户，也就是使用SQL SERVER的人，每个用来登录数据库的帐户都是一个用户。通过用户这个对象，可以设置数据库的使用权限。同一个数据库可以拥有多个用户，同一个用户也可以同时访问多个数据库。

■ 添加用户

选择页

- 常规
- 服务器角色
- 用户映射
- 安全对象
- 状态

连接

服务器:
B6AEEFA02E61474\SQLEXPRESS

连接:
B6AEEFA02E61474\Administrator

 [查看连接属性](#)

进度



就绪

 脚本  帮助

映射到此登录名的用户 (U):

映射	数据库	用户	默认架构
<input type="checkbox"/>	master		
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		
<input checked="" type="checkbox"/>	test	amei	...

☐ 已启用 Guest 帐户: test

数据库角色成员身份 (R): test

- ☐ db_accessadmin
- ☐ db_backupoperator
- ☐ db_datareader
- ☐ db_datawriter
- ☐ db_ddladmin
- ☐ db_denydatareader
- ☐ db_denydatawriter
- ☐ db_owner
- ☐ db_securityadmin
- ☒ public

确定

取消

创建用户

CREATE USER <用户名>

[{{FOR | FROM}

LOGIN <登录名>

| CERTIFICATE <证书名>

| ASYMMETRIC KEY <密钥名>

}

[WITHOUT LOGIN]

[WITH DEFAULT_SCHEMA = <架构名>]

]

[例3] 创建具有默认架构的数据库用户

```
CREATE USER 张三 FOR LOGIN 张三  
WITH DEFAULT_SCHEMA = student;
```

数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

存取控制

❖ 存取控制机制组成

- 定义用户权限
- 合法权限检查

❖ 用户权限定义和合法权检查机制一起组成了DBMS的安全子系统

❖ 常用存取控制方法

- 自主存取控制（Discretionary Access Control，简称DAC）
 - C2级
 - 灵活
- 强制存取控制（Mandatory Access Control，简称 MAC）
 - B1级
 - 严格

数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

■ 自主存取控制方法

■ 授权与回收

■ 数据库角色

❖ 强制存取控制方法

自主存取控制方法

❖ 通过 SQL 的 **GRANT** 语句和 **REVOKE** 语句实现

❖ 用户权限组成

- 数据对象

- 操作类型

❖ 定义用户存取权限：定义用户可以在哪些数据库对象上进行哪些类型的操作

❖ 定义存取权限称为**授权**

❖ 关系数据库系统中存取控制对象

对象类型	对象	操 作 类 型
数据库	模式	CREATE SCHEMA
	基本表	CREATE TABLE, ALTER TABLE
模式	视图	CREATE VIEW
	索引	CREATE INDEX
数据	基本表和视图	SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALL PRIVILEGES
数据	属性列	SELECT, INSERT, UPDATE, REFERENCES ALL PRIVILEGES

数据库安全性控制

❖ 数据库安全性控制概述

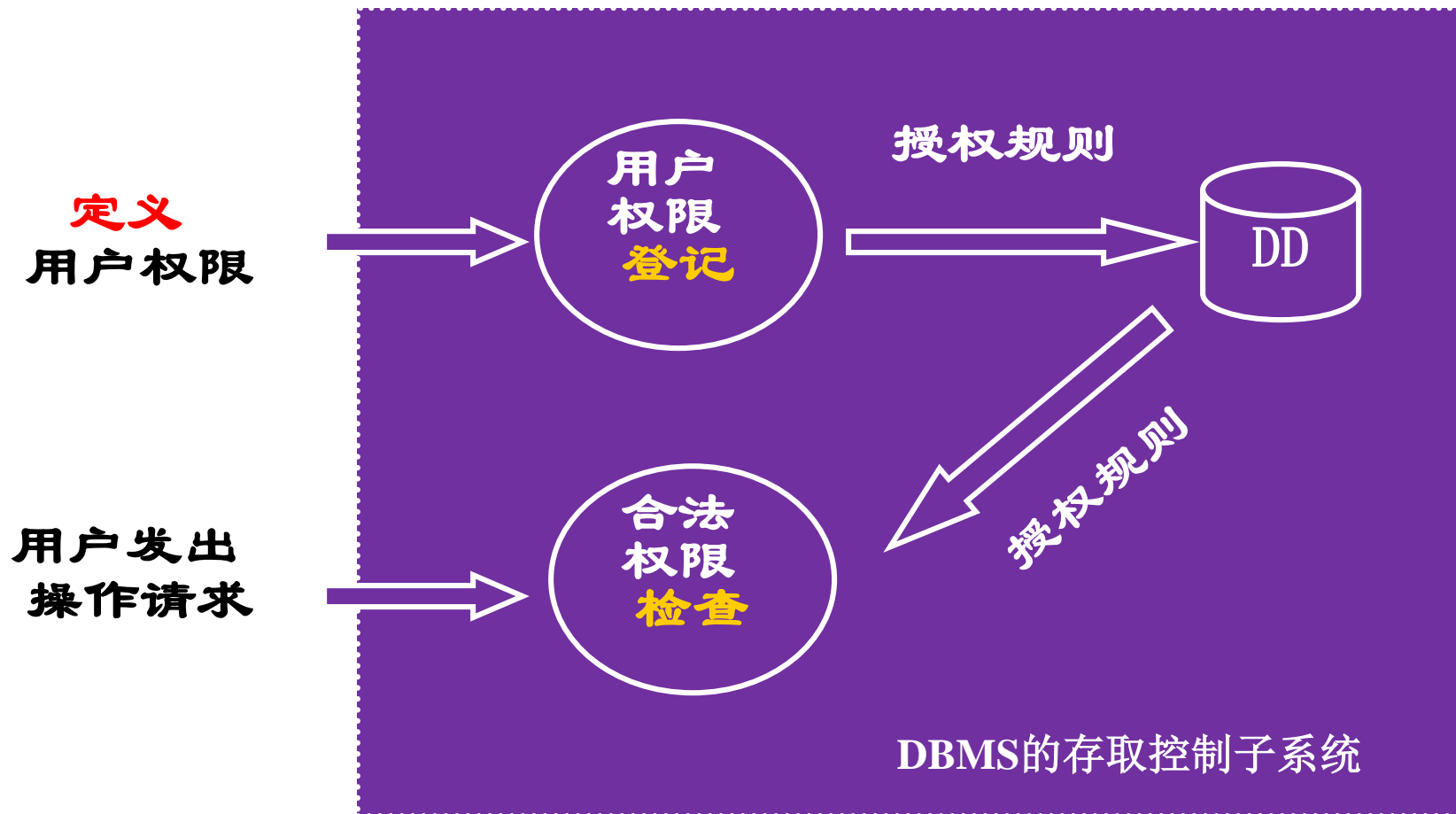
❖ 用户标识与鉴别

❖ 存取控制

- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

存取控制



GRANT

❖ GRANT语句的一般格式：

GRANT <权限>[,<权限>]...

[ON <对象类型> <对象名>]

TO <用户>[,<用户>]...

[WITH GRANT OPTION];

❖ 语义：将对指定操作对象的指定操作权限授予指定的
用户

❖ 发出GRANT

- DBA
- 数据库对象创建者（即属主Owner）
- 拥有该权限的用户

❖ 接受权限的用户

- 一个或多个具体用户
- PUBLIC（全体用户）

❖ WITH GRANT OPTION子句

- 指定：可以再授予
- 没有指定：不能传播
- 不允许循环授权



[例1] 把查询Student表权限授给用户U1。

```
GRANT SELECT ON TABLE Student  
TO U1;
```

[例2] 把对Student表和Course表的全部权限授予用户U2和U3。

```
GRANT ALL PRIVILEGES ON TABLE Student, Course  
TO U2, U3;
```

SQL不能同时对多个表授权

可以将多个权限授权给一个用户

[例3] 把对表SC的查询权限授予所有用户。

```
GRANT SELECT ON TABLE SC  
TO PUBLIC;
```

[例4] 把查询Student表和修改学生学号的权限授给用户U4。

```
GRANT UPDATE(Sno), SELECT ON TABLE Student  
TO U4;
```

[例5] 把对表SC的INSERT权限授予U5用户，并允许他
再将此权限授予其他用户。

```
GRANT INSERT ON TABLE SC  
TO U5
```

WITH GRANT OPTION;

执行例5后，U5不仅拥有了对表SC的
INSERT权限，还可以传播此权限：

[例6] GRANT INSERT ON TABLE SC
TO U6

WITH GRANT OPTION;



同样，U6还可以将此权限授予U7：

[例7] GRANT INSERT ON TABLE SC
TO U7;

但U7不能再传播此权限。



REVOKE

❖ REVOKE语句的一般格式为：

REVOKE <权限>[,<权限>]...

[ON <对象类型> <对象名>]

FROM <用户>[,<用户>]...;

❖ 授予的权限可以由DBA或其他授权者用REVOKE
语句收回

[例8] 把用户U4修改学生学号的权限收回。

```
REVOKE UPDATE(Sno) ON TABLE Student  
FROM U4;
```

[例9] 收回所有用户对表SC的查询权限。

```
REVOKE SELECT ON TABLE SC  
FROM PUBLIC;
```

[例10] 把用户U5对SC表的INSERT权限收回。

REVOKE INSERT **ON** TABLE SC
FROM U5 CASCADE;

- 将用户U5的INSERT权限收回的时候必须级联（CASCADE）收回
- 系统只收回直接或间接从U5处获得的权限

小结:SQL灵活的授权机制

❖ DBA：拥有所有对象的所有权限

- 不同的权限授予不同的用户

❖ 用户：拥有自己建立的对象的全部的操作权限

- GRANT：授予其他用户

❖ 被授权的用户

- “继续授权”许可：再授予

❖ 所有授予出去的权力在必要时又都可用REVOKE

语句收回

创建数据库模式的权限

❖ 对数据库模式的授权由DBA在创建用户时实现

❖ CREATE USER语句格式

CREATE USER <username>

[WITH] [DBA | RESOURCE | CONNECT]

拥有的权限	可否执行的操作			
	CREATE USER	CREATE SCHEMA	CREATE TABLE	登录数据库 执行数据查询和操纵
DBA	可以	可以	可以	可以
RESOURCE	不可以	不可以	可以	不可以
CONNECT	不可以	不可以	不可以	可以，但必须拥有相应权限

数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

数据库角色

❖ **数据库角色**：被命名的一组与数据库操作相关的权限

- 角色是权限的集合
- 可以为一组具有相同权限的用户创建一个角色
- 简化授权的过程

❖ **角色的创建**

- CREATE ROLE <角色名>

❖ **角色的授权**

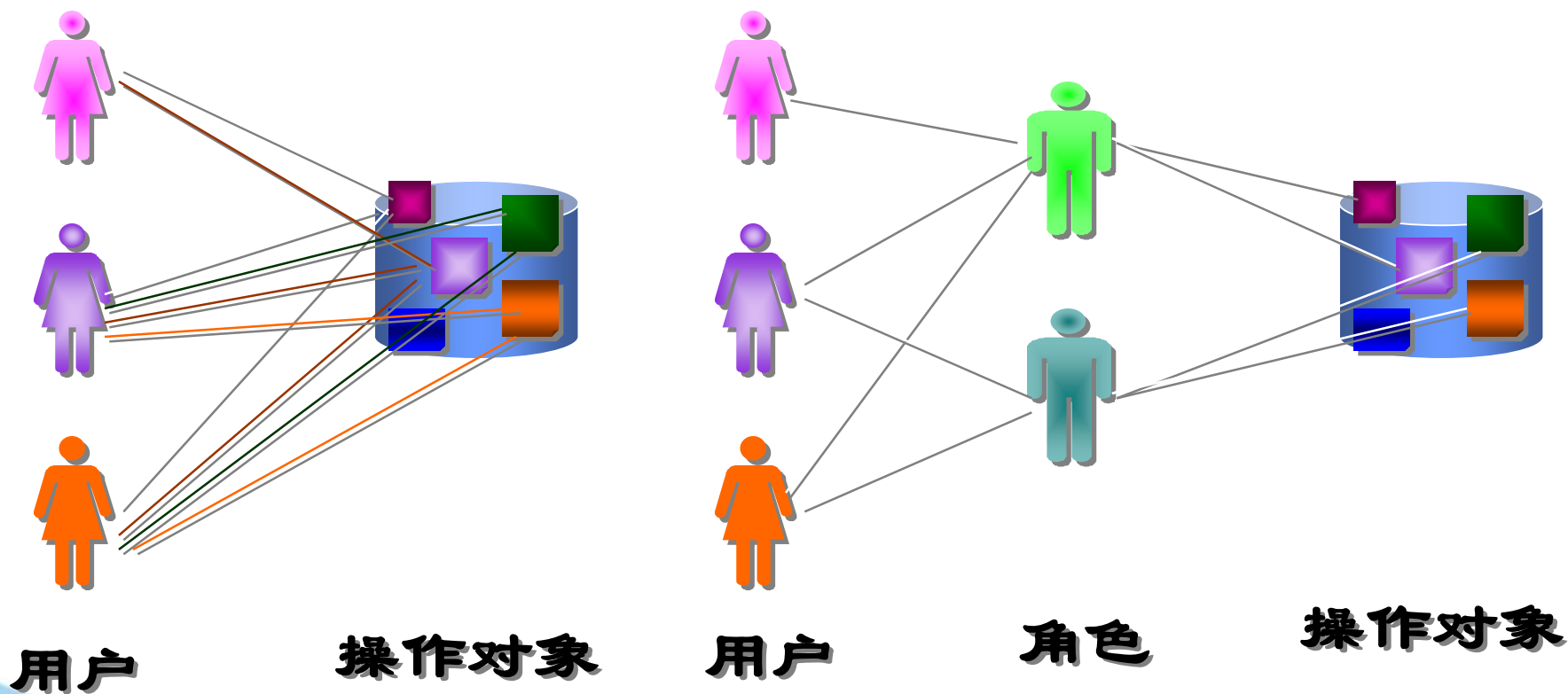
GRANT <权限> [, <权限>] ...

ON <对象类型>对象名

TO <角色> [, <角色>] ...

存取控制

角色 —— 就是操作权限的集合



❖ 将一个角色授予其他的角色或用户

GRANT <角色1> [, <角色2>] ...

TO <角色3> [, <用户1>] ...

[**WITH ADMIN OPTION**]

exec sp_addrolemember r1,zhang

❖ 角色权限的收回

REVOKE <权限> [, <权限>] ...

ON <对象类型> <对象名>

FROM <角色> [, <角色>] ...

[例11] 通过角色来实现将一组权限授予一个用户。


❖ 步骤如下：

1. 首先创建一个角色 R1

CREATE ROLE R1;

2. 然后使用GRANT语句，使角色R1拥有Student表的SELECT、UPDATE、INSERT权限

GRANT SELET, UPDATE,INSERT **ON TABLE** Student
TO R1;

- 
3. 将这个角色授予王平，张明，赵玲。使他们具有角色R1所包含的全部权限

GRANT R1

TO 王平、张明，赵玲;

4. 可以一次性通过R1来回收王平的这3个权限

REVOKE R1

FROM 王平;

数据库安全性控制

❖ 数据库安全性控制概述

❖ 用户标识与鉴别

❖ 存取控制

- 自主存取控制方法
- 授权与回收
- 数据库角色

❖ 强制存取控制方法

自主存取控制缺点

❖ 可能存在数据的“无意泄露”

❖ 原因：这种机制仅仅通过对数据的存取权限来进行安全控制，而数据本身并无安全性标记

❖ 解决：对系统控制下的所有主客体实施强制存取控制策略



强制存取控制



❖ 强制存取控制 (MAC)

- 保证更高层次的安全性
- 用户不能直接感知或进行控制
- 适用于对数据有严格而固定密级分类的部门
 - 军事部门
 - 政府部门

❖ 主体与客体

- 在MAC中，DBMS所管理的全部实体被分为主体和客体两大类

❖ **主体**是系统中的活动实体

- DBMS所管理的实际用户
- 代表用户的各进程

❖ **客体**是系统中的被动实体，是受主体操纵的

- 文件
- 基表
- 索引
- 视图

❖ **敏感度标记** (Label)

- 对于主体和客体，DBMS为它们每个实例（值）指派一个敏感度标记（Label）

■ 敏感度标记分成若干级别

- 绝密（Top Secret）
- 机密（Secret）
- 可信（Confidential）
- 公开（Public）

❖ **主体的敏感度标记称为许可证级别（Clearance Level）**

❖ **客体的敏感度标记称为密级（Classification Level）**

❖ **MAC机制就是通过对比主体的Label和客体的Label，
最终确定主体是否能够存取客体**

❖ 强制存取控制规则

- 当某一用户（或某一主体）以标记label注册入系统时，系统要求他对任何客体的存取必须遵循下面两条规则：
 - （1）仅当主体的许可证级别**大于或等于**客体的密级时，该主体才能**读取**相应的客体；
 - （2）仅当主体的许可证级别**等于**客体的密级时，该主体才能**写**相应的客体。

❖ 修正规则

- 主体的许可证级别 \leq 客体的密级 \rightarrow 主体能写客体

❖ 规则的共同点

- 禁止了拥有高许可证级别的主体更新低密级的数据对象

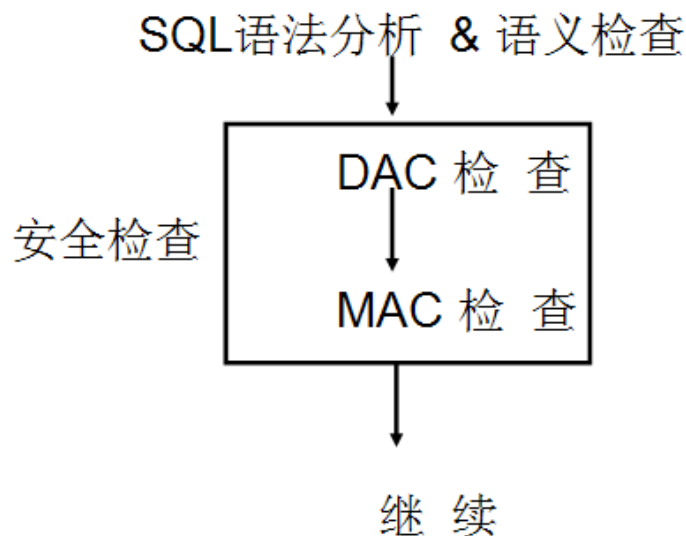
MAC与DAC

❖ DAC与MAC共同构成DBMS的安全机制

❖ 实现MAC时要首先实现DAC

- 原因：较高安全性级别提供的安全保护要包含较低级别的所有保护

❖ DAC + MAC



第4章 数据库安全性

❖ 第一节 计算机安全性概论(自学)

❖ 第二节 数据库安全性控制

● 第三节 视图机制

❖ 第四节 审计

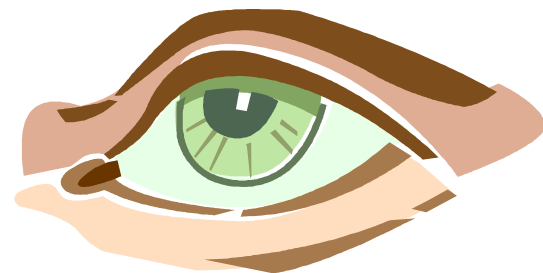
❖ 第五节 数据加密

❖ 第六节 统计数据库安全性(了解)



视图机制

- ❖ 视图机制把要保密的数据对无权存取这些数据的用户隐藏起来，对数据提供一定程度的安全保护
 - 视图机制更主要的功能在于提供数据独立性，其安全保护功能太不精细，往往远不能达到应用系统的要求。
 - 间接实现了支持存取谓词的用户权限定义



[例12] 建立计算机系学生的视图，把对该视图的
SELECT限授于王平，把该视图上的所有操作
权限授于张明。

1. 先建立计算机系学生的视图CS_Student

```
CREATE VIEW CS_Student
```

```
AS
```

```
SELECT *
```

```
FROM Student
```

```
WHERE Sdept='CS';
```

2. 在视图上进一步定义存取权限

GRANT SELECT ON CS_Student
TO 王平 ;

GRANT ALL PRIVILIGES ON CS_Student
TO 张明;

第4章 数据库安全性

❖ 第一节 计算机安全性概论(自学)

❖ 第二节 数据库安全性控制

❖ 第三节 视图机制

● 第四节 审计

❖ 第五节 数据加密

❖ 第六节 统计数据库安全性(了解)



审计

❖ 什么是审计

- 审计日志（Audit Log）

将用户对数据库的所有操作记录在上面

- DBA利用审计日志

找出非法存取数据的人、时间和内容

- C2以上安全级别的DBMS必须具有



❖ AUDIT语句：设置审计功能

❖ NOAUDIT语句：取消审计功能

[例13] 对修改SC表结构或修改SC表数据的操作进行审计。

AUDIT ALTER, UPDATE **ON** SC;

[例14] 取消对SC表的一切审计。

NOAUDIT ALTER, UPDATE **ON** SC;

第4章 数据库安全性

- ❖ 第一节 计算机安全性概论(自学)
- ❖ 第二节 数据库安全性控制
- ❖ 第三节 视图机制
- ❖ 第四节 审计
- 第五节 数据加密
- ❖ 第六节 统计数据库安全性(了解)



数据加密

❖ 数据加密

- 防止数据库中数据在存储和传输中失密的有效手段

❖ 加密的基本思想

- 根据一定的算法将原始数据（术语为明文，Plain text）变换为不可直接识别的格式（术语为密文，Cipher text）

❖ 加密方法

- 替换方法
- 置换方法
- 混合方法



这次课我们学到了...

- ❖ 用户的标识和鉴别，理解SQL SERVER的账户管理；
- ❖ 授权与回收，掌握GRANT和REVOKE用法，能够使用sql语句完成SQL SERVER授权和回收
- ❖ 掌握数据库角色的设置
- ❖ 理解视图机制和审计作用

作业

