



西安交通大学  
XI'AN JIAOTONG UNIVERSITY

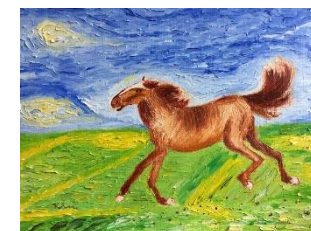
# 数据库系统

白秀秀

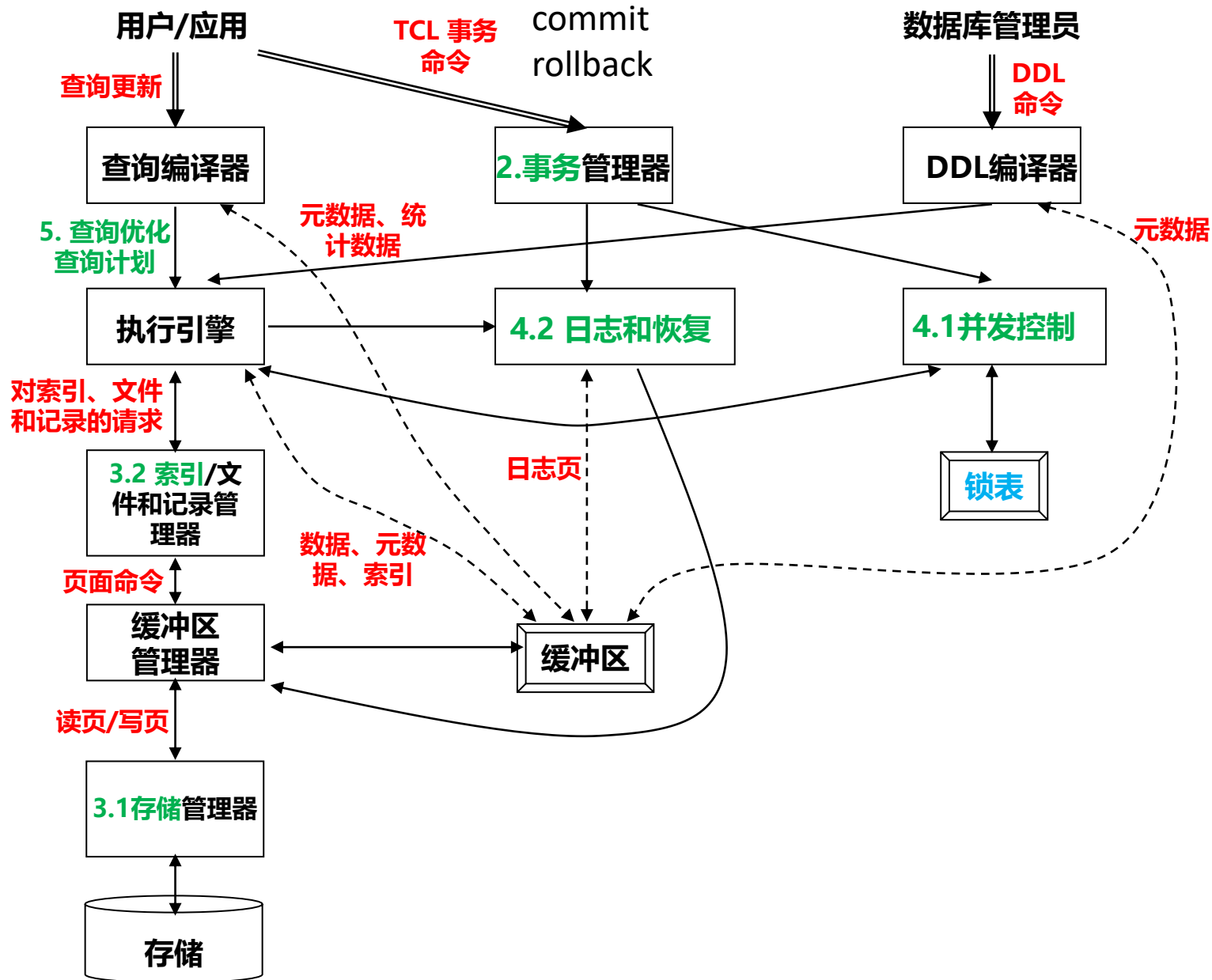
软件学院 副教授

西安交通大学

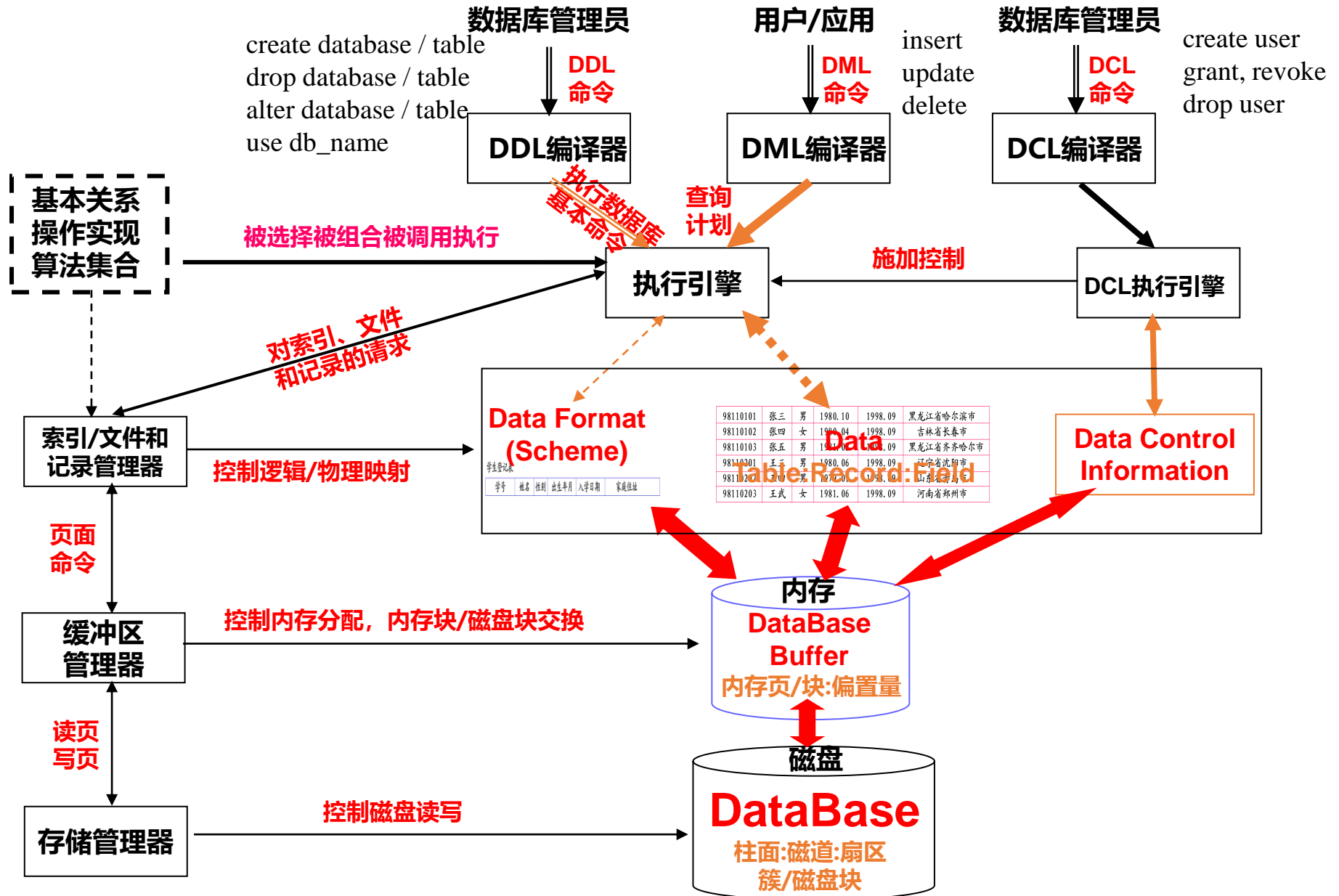
Email: [xiubai@xjtu.edu.cn](mailto:xiubai@xjtu.edu.cn)



# DBMS的核心技术



# DBMS 框架



# 知识表示的演进

- 知识在计算机内的表示是人工智能的核心问题。
- 早期，知识以结构化的方式存储在数据库中，人类需要掌握机器语言（如SQL），才能调用这些知识；
- 后来，随着互联网的诞生，更多文本、图片、视频等非结构化知识存储在互联网中，人类通过关键词的方式调用搜索引擎获取知识；
- 现在，知识以参数的形式存储在大模型中（从 2018 年开始），ChatGPT 主要解决了用自然语言直接调用这些知识的问题，这也是人类获取知识最自然的方式。

表 1.1: 知识表示和调用方式的演进

知识表示方式	表示方式的精确度	知识调用方式	调用方式的自然度	研究领域	代表应用	代表公司
关系型数据库	高	SQL	低	数据库	DBMS	Oracle、Microsoft
互联网	中	Keywords	中	信息检索	搜索引擎	Google、Microsoft
大模型	低	自然语言	高	自然语言处理	ChatGPT	OpenAI、Microsoft、Google

SU

建立一个关系型金融数据库



好的，以下是一个建立关系型金融数据库的示例：



sql

Copy code

```
CREATE DATABASE financial_db;
```

这个语句将创建一个名为"financial\_db"的数据库。接下来，你可以使用以下语句创建金融数据表格并将其添加到这个数据库中：

sql

Copy code

```
USE financial_db;

CREATE TABLE financial_data (
  id INT PRIMARY KEY,
  date DATE,
  company VARCHAR(255),
  revenue DECIMAL(12, 2),
  expenses DECIMAL(12, 2),
  net_income DECIMAL(12, 2)
);
```

这个表格中包含了金融数据的基本信息，包括日期、公司、收入、支出和净收入。你可以根据实际需要修改表格的字段名称、类型和长度。在实际使用中，你还可以在这个表格上建立索引，以提高查询效率。

SU

你给上面这个数据库添加5组数据



好的，以下是一个向上面创建的金融数据表格中添加5组数据的示例：



sql

Copy code

```
USE financial_db;

INSERT INTO financial_data (id, date, company, revenue, expenses, net_income)
VALUES
(1, '2022-01-01', 'ABC Corp', 1000000.00, 500000.00, 500000.00),
(2, '2022-01-02', 'XYZ Inc', 750000.00, 400000.00, 350000.00),
(3, '2022-01-03', '123 Co', 600000.00, 300000.00, 300000.00),
(4, '2022-01-04', 'Acme Corp', 1250000.00, 700000.00, 550000.00),
(5, '2022-01-05', 'Big Corp', 2000000.00, 800000.00, 1200000.00);
```

这个示例添加了5组数据，包括日期、公司、收入、支出和净收入。你可以根据实际需要修改数据的字段值。注意，在添加数据时，要确保它们符合表格定义的数据类型和长度要求。

SU

再添加3组数据



好的，以下是向上面创建的金融数据表格中添加3组数据的示例：



sql

Copy code

```
USE financial_db;

INSERT INTO financial_data (id, date, company, revenue, expenses, net_income)
VALUES
(6, '2022-01-06', 'DEF Corp', 900000.00, 400000.00, 500000.00),
(7, '2022-01-07', 'GHI Inc', 1500000.00, 900000.00, 600000.00),
(8, '2022-01-08', '456 Co', 400000.00, 200000.00, 200000.00);
```

这个示例添加了3组数据，包括日期、公司、收入、支出和净收入。你可以根据实际需要修改数据的字段值。同样地，在添加数据时，要确保它们符合表格定义的数据类型和长度要求。

SU

删除第6组数据



好的，以下是删除第6组数据的示例：



sql

Copy code

```
USE financial_db;
```

```
DELETE FROM financial_data WHERE id = 6;
```

这个示例使用了`DELETE`语句来删除id等于6的数据行。你可以根据实际需要修改删除条件。如果你不知道要删除的数据的id，可以使用其他条件来限制删除，比如日期、公司等等。但要注意，使用不恰当的条件可能会导致误删数据，所以在删除数据时要格外小心。

SU

显式一下现在的数据库的数据情况



好的，以下是查询现在金融数据库表格中所有数据的示例：



sql

Copy code

```
USE financial_db;
```

```
SELECT * FROM financial_data;
```

这个示例使用了`SELECT`语句来查询金融数据表格中的所有数据。`\*`表示查询所有字段，你也可以根据需要查询指定字段。执行这个语句后，数据库会返回所有数据行的信息。如果你使用的是图形界面的数据库工具，可以在结果集中查看数据。如果你使用的是命令行工具，可以将结果导出到文件或屏幕上。

SU

给上述数据库生成一个触发器



好的，以下是一个向金融数据表格中添加触发器的示例：



sql

Copy code

```
USE financial_db;
```

```
CREATE TRIGGER update_net_income AFTER INSERT ON financial_data
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    UPDATE financial_data SET net_income = revenue - expenses WHERE id = NEW
```

```
END;
```

这个示例创建了一个名为`update\_net\_income`的触发器，当有新的数据插入到金融数据表格中时，它会自动计算并更新净收入字段的值。这个触发器在每行插入后执行，使用`NEW`关键字获取新插入的数据行的值。在这个示例中，我们使用`UPDATE`语句计算净收入，并将结果更新到表格中。你可以根据需要修改触发器的名称、触发时机、执行条件和动作等等。

SU

给上述数据库生成一个视图



好的，以下是一个用于创建视图的示例：



sql

Copy code

```
USE financial_db;
```

```
CREATE VIEW financial_summary AS
```

```
SELECT company, SUM(revenue) as total_revenue, SUM(expenses) as total_expenses
```

```
FROM financial_data
```

```
GROUP BY company;
```

这个示例创建了一个名为`financial\_summary`的视图，该视图将金融数据表格中的数据按公司进行汇总，并计算出每个公司的总收入、总支出和总净收入。视图本身不存储数据，而是基于数据表格动态生成结果集。每次查询视图时，数据库都会动态地从底层数据表格中提取数据，然后按照定义的逻辑进行汇总和计算。在这个示例中，我们使用`SELECT`语句和`GROUP BY`子句来定义视图的查询逻辑，然后使用`CREATE VIEW`语句创建视图。



# 第8章 数据库安全性

---

## 8.1 数据库安全性概述

## 8.2 数据库安全性控制

## 8.3 视图机制

## 8.4 审计(Audit)

## 8.5 数据加密

## 8.6 其他安全性保护



# 数据库安全性

---

- 2019年8月29日，世界最大的社交网络公司Facebook（Meta），被爆出数据丑闻。
- 其5000万用户数据资料，被第三方数据分析公司违规使用，股价本周一下跌6.8%，市值蒸发367亿美元。





# 数据库安全性

---

- 2018年2月，随着MongoDB, CouchDB和Cassandra服务器的沦陷，[MySQL](#)数据库成了攻击者的下一个目标。
- 数千个MySQL数据库被劫持，删除了存储数据，并留下勒索信息，要求支付0.2比特币的赎金。
- 攻击由2018年2月12日凌晨00:15发起，在30个小时内，攻击者拿下了成百上千个暴露在公网的MySQL服务器。
- 经调查，在此次勒索攻击中，所有的攻击皆来自相同的IP地址，109.236.88.20，属于荷兰的一家网络托管服务提供商WorldStream。



# 数据库安全性

---

- 攻击者（可能）利用了一台被盗的邮件服务器，该服务器同样可以提供HTTP和FTP服务器所提供的服务。
- 攻击以“root”密码暴力破解开始，一旦成功登陆，该黑客会获取已有MySQL数据库及其表的列表，
- 在已有的数据库中新建一个名为WARNING的表，插入信息包括一个邮箱地址、比特币地址和支付需求。



# 数据库安全性

➤服务器被攻击破解，进行挖矿。

```
: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
: Failed password for invalid user ashil from 202.117.43.83 port 2438 ssh2
: Received disconnect from 202.117.43.83 port 2438:11: Normal Shutdown, Thank you for playing [preauth]
: Disconnected from 202.117.43.83 port 2438 [preauth]
Invalid user weiang from 202.117.43.83
input_userauth_request: invalid user weiang [preauth]
pam_unix(sshd:auth): check pass; user unknown
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
Failed password for invalid user weiang from 202.117.43.83 port 2439 ssh2
Received disconnect from 202.117.43.83 port 2439:11: Normal Shutdown, Thank you for playing [preauth]
Disconnected from 202.117.43.83 port 2439 [preauth]
Invalid user Muhammad from 202.117.43.83
input_userauth_request: invalid user Muhammad [preauth]
pam_unix(sshd:auth): check pass; user unknown
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
Failed password for invalid user Muhammad from 202.117.43.83 port 2440 ssh2
Received disconnect from 202.117.43.83 port 2440:11: Normal Shutdown, Thank you for playing [preauth]
Disconnected from 202.117.43.83 port 2440 [preauth]
Invalid user ymli from 202.117.43.83
input_userauth_request: invalid user ymli [preauth]
pam_unix(sshd:auth): check pass; user unknown
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
Failed password for invalid user ymli from 202.117.43.83 port 2441 ssh2
Received disconnect from 202.117.43.83 port 2441:11: Normal Shutdown, Thank you for playing [preauth]
Disconnected from 202.117.43.83 port 2441 [preauth]
Invalid user dcyan from 202.117.43.83
input_userauth_request: invalid user dcyan [preauth]
pam_unix(sshd:auth): check pass; user unknown
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
Failed password for invalid user dcyan from 202.117.43.83 port 2442 ssh2
Received disconnect from 202.117.43.83 port 2442:11: Normal Shutdown, Thank you for playing [preauth]
Disconnected from 202.117.43.83 port 2442 [preauth]
Invalid user zbcai from 202.117.43.83
input_userauth_request: invalid user zbcai [preauth]
pam_unix(sshd:auth): check pass; user unknown
pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=202.117.43.83
Failed password for invalid user zbcai from 202.117.43.83 port 2443 ssh2
Received disconnect from 202.117.43.83 port 2443:11: Normal Shutdown, Thank you for playing [preauth]
Disconnected from 202.117.43.83 port 2443 [preauth]
Invalid user muhammad from 202.117.43.83
```



# 数据库安全性

---

➤ 该怎么防范这种攻击？





# 数据库安全性

---

- 传统的数据库，需要一定的访问权限去管理，也就是说你必须依赖人为的管理。一旦访问权限被黑或者管理员作恶，数据就有很大的危险性，或者存在被人篡改的可能性。
  - 在现实世界中，银行和金融机构以及各种大公司通过一些方式积攒了一些信用。
  - 银行本身需要一批可信的数据库管理员，开发人员，以及一些安全的管理流程，依赖这些，来保证数据的安全性。
- 区块链，提供了一种方式去取代这些额外的成本，用数字加密学与节点共识的机制来保证数据的安全。





# 数据库安全性

---

## ➤ 问题的提出

- 数据库的一大特点是数据可以共享。
- 数据共享必然带来数据库的安全性问题。
- 数据库系统中的数据共享不能是无条件的共享。

例： 军事秘密、国家机密、新产品实验数据、  
市场需求分析、市场营销策略、销售计划、  
客户档案、医疗档案、银行储蓄数据



数据库安全性



# 数据库安全性

---

- 数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏。
- 系统安全保护措施是否有效是数据库系统主要的性能指标之一。



# 8.1 数据库安全性概述

---

数据库的不安全因素

安全标准简介





# 数据库的不安全因素

---

## 1.非授权用户对数据库的恶意存取和破坏

- 一些黑客(Hacker)和犯罪分子在用户存取数据库时猎取用户名和用户口令,然后假冒合法用户偷取、修改甚至破坏用户数据。
- 数据库管理系统提供的安全措施主要包括用户身份鉴别、存取控制和视图等技术。





---

## ➤ 2.数据库中重要或敏感的数据被泄露

- 黑客和敌对分子千方百计盗窃数据库中的重要数据，一些机密信息被暴露。
- 数据库管理系统提供的主要技术有强制存取控制、数据加密存储和加密传输等。
- 审计日志分析。



---

### ➤3.安全环境的脆弱性

- 数据库的安全性与计算机系统的安全性紧密联系

  - 计算机硬件、操作系统、网络系统等的安全性

- 需要建立一套可信(Trusted)计算机系统的概念和标准。



# 安全标准简介

---

- 1985年美国国防部(DoD)正式颁布《DoD可信计算机系统评估准则》(简称TCSEC或DoD85)。
- 不同国家建立在TCSEC概念上的评估准则
  - 欧洲的信息技术安全评估准则(ITSEC)
  - 加拿大的可信计算机产品评估准则(CTCPEC)
  - 美国的信息技术安全联邦标准(FC)



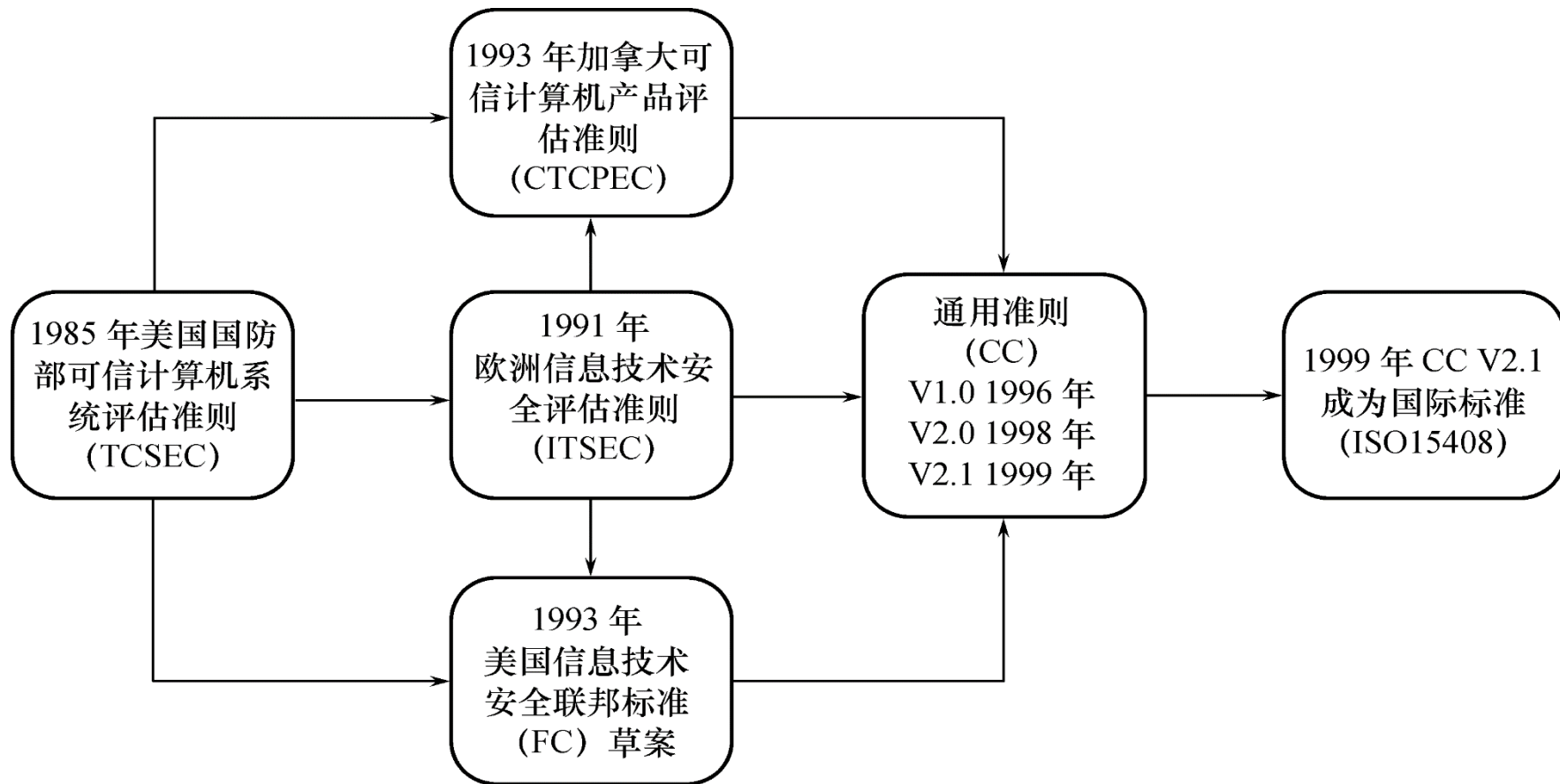
# 安全标准简介

---

- 1993年，CTCPEC、FC、TCSEC和ITSEC联合行动，解决原标准中概念和技术上的差异，称为CC(Common Criteria)项目。
- 1999年 CC V2.1版被ISO采用为国际标准
- 2001年 CC V2.1版被我国采用为国家标准
- 目前CC已基本取代了TCSEC，成为评估信息产品安全性的主要标准。



# 安全标准简介



信息安全标准的发展历史



# 安全标准简介

---

➤ TCSEC标准

➤ CC标准





## TCSEC标准

---

- 1991年4月美国NCSC(国家计算机安全中心)颁布了《可信计算机系统评估标准关于可信数据库系统的解释》(Trusted Database Interpretation 简称TDI)。
  - TDI又称紫皮书。它将TCSEC扩展到数据库管理系统。
  - TDI中定义了数据库管理系统的设计与实现中需满足和用以进行安全性级别评估的标准。



# TCSEC标准

---

## ➤ TCSEC/TDI标准的基本内容

➤ TCSEC/TDI，从四个方面来描述安全性级别划分的指标

- 策略
- 责任
- 保证
- 文档



# TCSEC/TDI安全级别划分

## ➤TCSEC/TDI安全级别划分

安全级别	定 义
<b>A1</b>	验证设计(Verified Design)
<b>B3</b>	安全域(Security Domains)
<b>B2</b>	结构化保护(Structural Protection)
<b>B1</b>	标记安全保护(Labeled Security Protection)
<b>C2</b>	受控的存取保护(Controlled Access Protection)
<b>C1</b>	自主安全保护(Discretionary Security Protection)
<b>D</b>	最小保护(Minimal Protection)



# TCSEC/TDI安全级别划分

---

## ➤四组(division)七个等级

- D
- C(C1, C2)
- B(B1, B2, B3)
- A(A1)

➤按系统可靠或可信程度逐渐增高。

➤各安全级别之间具有一种偏序向下兼容的关系，即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要求，同时提供更多或更完善的保护能力。



# TCSEC/TDI安全级别划分

---

## ➤D级

- 将一切不符合更高标准的系统均归于D组。
- 典型例子：DOS是安全标准为D的操作系统。
  - DOS在安全性方面几乎没有什么专门的机制来保障。



# TCSEC/TDI安全级别划分

---

## ➤C1级

- 非常初级的自主安全保护。
- 能够实现对用户和数据的分离，进行自主存取控制(DAC)，保护或限制用户权限的传播。
- 现有的商业系统稍作改进即可满足。



# TCSEC/TDI安全级别划分

---

## ➤C2级

- 安全产品的最低档次。
- 提供受控的存取保护，将C1级的DAC进一步细化，以个人身份注册负责，并实施审计和资源隔离。
- 达到C2级的产品在其名称中往往不突出“安全”(Security)这一特色。
- 典型例子
  - Windows 2000
  - Oracle 7



# TCSEC/TDI安全级别划分

---

## ➤B1级

➤标记安全保护。“安全”(Security)或“可信的”(Trusted)产品。

➤对系统的数据加以标记，对标记的主体和客体实施强制存取控制(MAC)、审计等安全机制

## ➤B1级典型例子

- 操作系统

- 惠普公司的HP-UX BLS release 9.09+

- 数据库

- Oracle公司的Trusted Oracle 7

- Sybase公司的Secure SQL Server version 11.0.6





# TCSEC/TDI安全级别划分

---

## ➤B2级

➤结构化保护。

➤建立形式化的安全策略模型并对系统内的所有主体和客体实施DAC和MAC。



# TCSEC/TDI安全级别划分

---

## ➤B3级

### ➤安全域

➤该级的TCB必须满足访问监控器的要求，审计跟踪能力更强，并提供系统恢复过程。

## ➤A1级

➤验证设计，即提供B3级保护的同时给出系统的形式化设计说明和验证以确信各安全保护真正实现。



# CC标准

---

## ➤ CC

➤提出国际公认的表述信息技术安全性的结构。

➤把信息产品的安全要求分为：

- 安全功能要求
- 安全保证要求



# CC

---

## ➤ CC文本组成

### ➤ 简介和一般模型

- 有关术语、基本概念和一般模型以及与评估有关的一些框架。

### ➤ 安全功能要求

- 列出了一系列类、子类和组件。

### ➤ 安全保证要求

- 列出了一系列保证类、子类和组件。
- 提出了评估保证级(Evaluation Assurance Level, EAL), 从EAL1至EAL7共分为七级。



# CC

- CC评估保证级(EAL)划分

评估保证级	定 义	TCSEC安全级别(近似相当)
EAL1	功能测试(functionally tested)	
EAL2	结构测试(structurally tested)	C1
EAL3	系统地测试和检查(methodically tested and checked)	C2
EAL4	系统地设计、测试和复查(methodically designed, tested, and reviewed)	B1
EAL5	半形式化设计和测试(semiformally designed and tested)	B2
EAL6	半形式化验证的设计和测试(semiformally verified design and tested)	B3
EAL7	形式化验证的设计和测试(formally verified design and tested)	A1



# 第8章 数据库安全性

---

8.1 数据库安全性概述

8.2 数据库安全性控制

8.3 视图机制

8.4 审计(Audit)

8.5 数据加密

8.6 其他安全性保护



## 8.2 数据库安全性控制

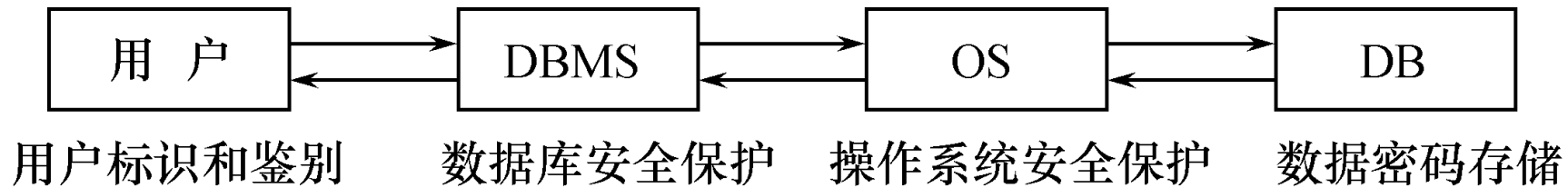
---

- 非法使用数据库的情况
  - 编写合法程序绕过数据库管理系统及其授权机制。
  - 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。



# 数据库安全性控制

- 计算机系统中，安全措施是一级一级层层设置



计算机系统的安全模型

- 系统根据用户标识鉴定用户身份，合法用户才准许进入计算机系统。
- 数据库管理系统还要进行存取控制，只允许用户执行合法操作。
- 操作系统有自己的保护措施。
- 数据以密码形式存储到数据库中。

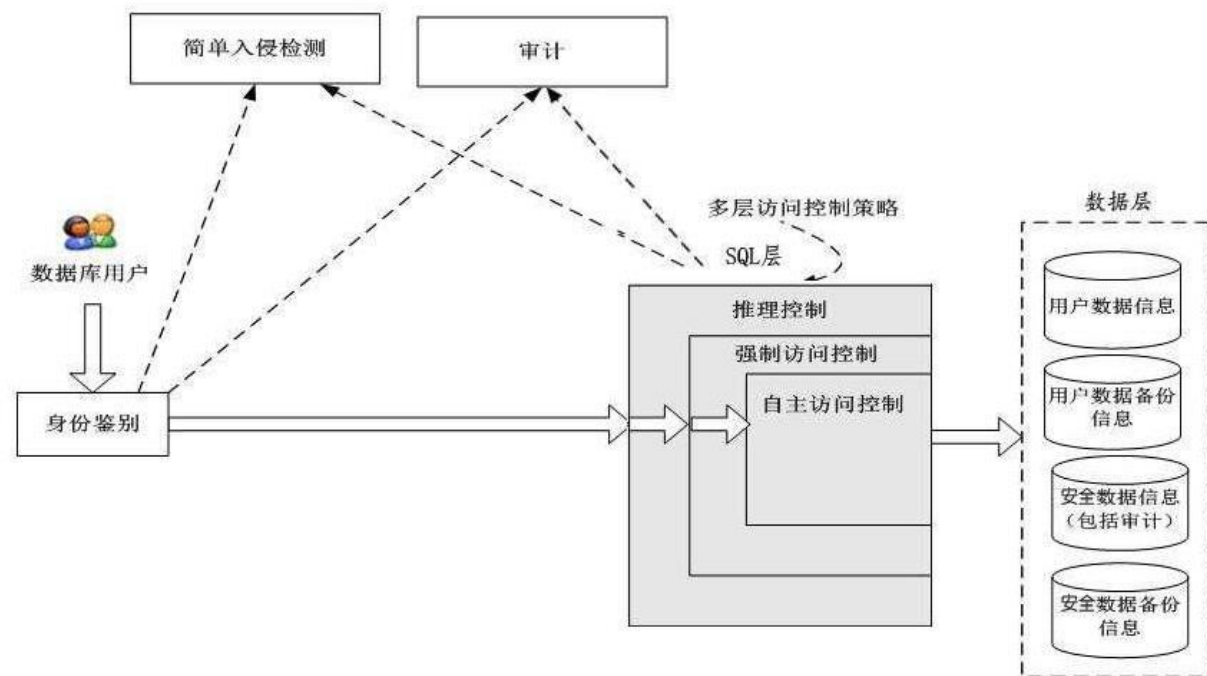




# 数据库安全性控制

## ➤ 存取控制流程

- 首先，数据库管理系统对提出SQL访问请求的数据库用户进行身份鉴别，防止不可信用户使用系统。
- 然后，在SQL处理层进行自主存取控制和强制存取控制，进一步可以进行推理控制。
- 还可以对用户访问行为和系统关键操作进行审计，对异常用户行为进行简单入侵检测。



图：数据库管理系统安全性控制模型



# 数据库安全性控制

---

## ➤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

- 数据加密



# 用户身份鉴别

---

## ➤ 用户身份鉴别 (Identification & Authentication)

➤ 系统提供的最外层安全保护措施

➤ 用户标识：由用户名和用户标识号组成

(用户标识号在系统整个生命周期内唯一)



# 用户身份鉴别

---

## ➤ 用户身份鉴别的方法

### ➤ 1. 静态口令鉴别

➤ 静态口令一般由用户自己设定，这些口令是静态不变的。

### ➤ 2. 动态口令鉴别

➤ 口令是动态变化的，每次鉴别时均需使用动态产生的新口令登录数据库管理系统，即采用一次一密的方法。例如U盾。

### ➤ 3. 生物特征鉴别

➤ 通过生物特征进行认证的技术，生物特征如指纹、虹膜和掌纹等。

### ➤ 4. 智能卡鉴别

➤ 智能卡是一种不可复制的硬件，内置集成电路的芯片，具有硬件加密功能。



# 存取控制

---

## ➤存取控制机制组成

➤定义用户权限，并将用户权限登记到数据字典中。

- 用户对某一数据对象的操作权力称为权限。
- DBMS提供适当的语言来定义用户权限，存放在数据字典中，称做安全规则或授权规则。

## ➤合法权限检查

- 用户发出存取数据库操作请求。
- DBMS查找数据字典，进行合法权限检查。

➤用户权限定义和合法权检查机制一起组成了数据库管理系统的存取控制子系统。



# 存取控制

## ➤ 常用存取控制方法

### ➤ 自主存取控制(Discretionary Access Control，简称DAC)

- C2级
- 用户对不同的数据对象有不同的存取权限
- 不同的用户对同一对象也有不同的权限
- 用户还可将其拥有的存取权限转授给其他用户

评估保证级	定 义	TCSEC安全级别
EAL1	功能测试(functionally tested)	
EAL2	结构测试(structurally tested)	C1
EAL3	系统地测试和检查 (methodically tested and checked)	C2
EAL4	系统地设计、测试和复查 (methodically designed, tested, and reviewed)	B1
EAL5	半形式化设计和测试 (semiformally designed and tested)	B2
EAL6	半形式化验证的设计和测试 (semiformally verified design and tested)	B3
EAL7	形式化验证的设计和测试 (formally verified design and tested)	A1



# 存取控制

---

## ➤ 常用存取控制方法

### ➤ 强制存取控制(Mandatory Access Control, 简称 MAC)

- B1级

- 每一个数据对象被标以一定的密级
- 每一个用户也被授予某一个级别的许可证
- 对于任意一个对象，只有具有合法许可证的用户才可以存取



# 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。





# 自主存取控制方法

- 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	模式	CREATE SCHEMA
	基本表	CREATE TABLE, ALTER TABLE
	视图	CREATE VIEW
	索引	CREATE INDEX
数据	基本表和视图	SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALL PRIVILEGES
	属性列	SELECT, INSERT, UPDATE, REFERENCES, ALL PRIVILEGES

关系数据库系统中的存取权限



# 授权：授予与回收

---

## 1. GRANT

➤ GRANT语句的一般格式：

GRANT <权限>[,<权限>]...

ON <对象类型> <对象名>[,<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



# GRANT

---

## ➤发出GRANT:

- 数据库管理员
- 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(即全体用户)



# WITH GRANT OPTION子句

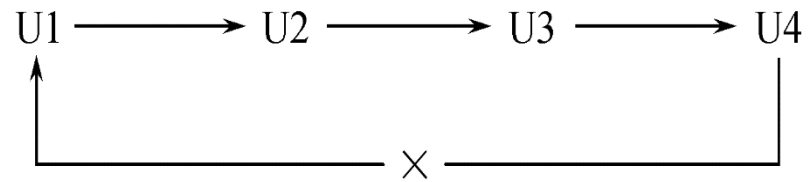
---

## ➤ 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





# 例题

---

[例8.1] 把查询Student表权限授给用户U1

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



## 例题

---

[例 8.2] 把对 Student 表和 Course 表的全部权限授予 Discretionary 用户 U2 和 U3

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



# 例题

---

[例8.3] 把对表SC的查询权限授予所有用户

GRANT SELECT

ON TABLE SC

TO PUBLIC;



# 例题

---

[例8.4] 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;
```

➤对属性列的授权时必须明确指出相应属性列名





# 例题

---

[例8.5] 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;



# 传播权限

---

执行例8.5后，U5不仅拥有了对表SC的INSERT权限，  
还可以传播此权限：

```
[例8.6] GRANT INSERT  
        ON TABLE SC  
        TO U6  
        WITH GRANT OPTION;
```

同样，U6还可以将此权限授予U7：

```
[例8.7] GRANT INSERT  
        ON TABLE SC  
        TO U7;
```

但U7不能再传播此权限。



# 传播权限

执行了例8.1~例8.7语句后学生-课程数据库中的用户权限定义表

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系Student	SELECT	不能
DBA	U2	关系Student	ALL	不能
DBA	U2	关系Course	ALL	不能
DBA	U3	关系Student	ALL	不能
DBA	U3	关系Course	ALL	不能
DBA	PUBLIC	关系SC	SELECT	不能
DBA	U4	关系Student	SELECT	不能
DBA	U4	属性列Student.Sno	UPDATE	不能
DBA	U5	关系SC	INSERT	能
U5	U6	关系SC	INSERT	能
U6	U7	关系SC	INSERT	不能



# 授权：授予与回收

---

## 2.REVOKE

- 授予的权限可以由数据库管理员或其他授权者用REVOKE语句收回
- REVOKE语句的一般格式为：

REVOKE <权限>[,<权限>]...

ON <对象类型> <对象名>[,<对象类型><对象名>]...

FROM <用户>[,<用户>]...[CASCADE | RESTRICT];



# REVOKE

---

[例8.8] 把用户U4修改学生学号的权限收回

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



# REVOKE

---

[例8.9] 收回所有用户对表SC的查询权限

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



# REVOKE

---

[例8.10] 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权限收回的时候应该使用**CASCADE**，否则拒绝执行该语句。
- 如果U6或U7还从其他用户处获得对SC表的INSERT权限，则他们仍具有此权限，系统只收回直接或间接从U5处获得的权限。



# REVOKE

---

执行例8.8~8.10语句后学生-课程数据库中的用户权限定义表

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系Student	SELECT	不能
DBA	U2	关系Student	ALL	不能
DBA	U2	关系Course	ALL	不能
DBA	U3	关系Student	ALL	不能
DBA	U3	关系Course	ALL	不能
DBA	U4	关系Student	SELECT	不能





## 小结: SQL灵活的授权机制

---

- 数据库管理员:
  - 拥有所有对象的所有权限
  - 根据实际情况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
  - 可以使用**GRANT**，把权限授予其他用户
- 被授权的用户
  - 如果具有“继续授权”的许可，可以把获得的权限再授予其他用户
- 所有授予出去的权力在必要时又都可用**REVOKE**语句收回



# 授权：授予与回收

---

## 3.创建数据库模式的权限

- 数据库管理员在创建用户时实现
- CREATE USER语句格式

CREATE USER <username>

[WITH][DBA|RESOURCE|CONNECT];

注：

CREATE USER不是SQL标准，各个系统的实现相差甚远



# 授权：授予与回收

---

## ➤ CREATE USER语句格式说明

- 只有系统的超级用户才有权创建一个新的数据库用户
- 新创建的数据库用户有三种权限：CONNECT、RESOURCE和DBA
- 如没有指定创建的新用户的权限，默认该用户拥有CONNECT权限。
- 拥有CONNECT权限的用户不能创建新用户，不能创建模式，也不能创建基本表，只能登录数据库



# 授权：授予与回收

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	可以
CONNECT	不可以	不可以	不可以	可以，但必须拥有相应权限

**DBA权限，是系统中的超级用户**

权限与可执行的操作对照表



# 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



# 数据库角色

---

## 1.角色的创建

CREATE ROLE <角色名>

## 2.给角色授权

GRANT <权限>[,<权限>]...

ON <对象类型>对象名

TO <角色>[,<角色>]...



# 数据库角色

---

## 3. 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]

- 该语句把角色授予某用户，或授予另一个角色。
- 授予者是角色的创建者或拥有在这个角色上的ADMIN OPTION。
- 指定了WITH ADMIN OPTION则获得某种权限的角色或用户还可以把这种权限授予其他角色。
- 一个角色的权限：直接授予这个角色的全部权限加上其他角色。授予这个角色的全部权限。



# 数据库角色

---

## ➤ 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>]...

■ 用户可以回收角色的权限，从而修改角色拥有的权限

■ **REVOKE**执行者是

- 角色的创建者
- 拥有在这个(些)角色上的**ADMIN OPTION**





# 数据库角色

---

[例8.11] 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 SELECT, UPDATE, INSERT
```

```
ON TABLE Student
```

```
TO R1;
```



# 数据库角色

---

(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;



# 数据库角色

---

[例8.12] 角色的权限修改

GRANT DELETE

ON TABLE Student

TO R1;

使角色R1在原来的基础上增加了Student表的DELETE 权限



# 数据库角色

---

[例8.13]

REVOKE SELECT

ON TABLE Student

FROM R1;

使R1减少了SELECT权限



## 自主存取控制（DAC）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。



# 强制存取控制方法

---

## ➤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



# 强制存取控制方法

---

❖ 在强制存取控制中，数据库管理系统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

- 数据库管理系统所管理的实际用户。
- 代表用户的各进程。

➤ 客体是系统中的被动实体，受主体操纵

- 文件、基本表、索引、视图。



# 强制存取控制方法

---

- 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, TS)
  - 机密(Secret, S)
  - 可信(Confidential, C)
  - 公开(Public, P)
  - $TS \geq S \geq C \geq P$
- 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)。





# 强制存取控制方法

---

## ➤ 强制存取控制规则

- (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小于或等于客体的密级时，该主体才能写相应的客体。



# 强制存取控制方法

---

- 规则（2），用户可以**为写入的数据对象赋予高于自己的许可证级别的密级**。
  - 这样**一旦数据被写入，该用户自己也不能再读该数据对象了**。
- 如果违反了规则（2），就有可能把数据的密级从高流向低，造成数据的泄露。
- 例如，某个TS密级的主体把一个密级为TS的数据**恶意地降低为P，然后把它写回**。
  - 这样原来是TS密级的数据大家都可以读到了，造成了TS密级数据的泄露。



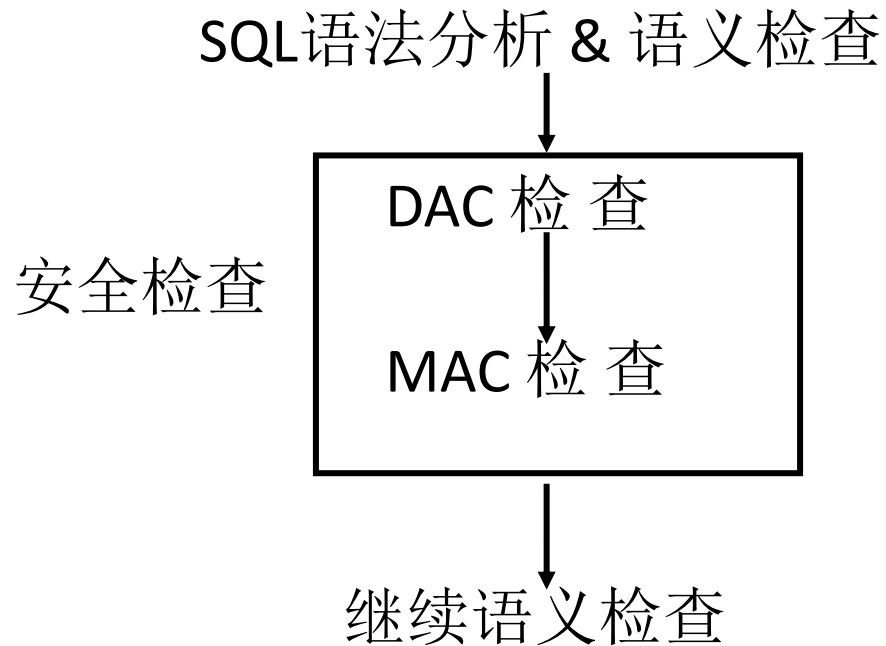
# 强制存取控制方法

---

- 强制存取控制(MAC)是**对数据本身进行密级标记**，无论数据如何复制，**标记与数据是一个不可分的整体**，只有符合密级标记要求的用户才可以操纵数据。
- 实现强制存取控制时要**首先实现自主存取控制**
  - **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
- 自主存取控制与强制存取控制共同构成**数据库管理系统的安全机制**



# DAC + MAC安全检查



- ❖ 先进行自主存取控制（**DAC**）检查，通过**DAC**检查的数据对象再由系统进行强制存取控制检查，只有通过强制存取控制检查（**MAC**）的数据对象方可存取。
- ❖ **Discretionary Access Control, DAC**
- ❖ **Mandatory Access Control, MAC**



# 第8章 数据库安全性

---

8.1 数据库安全性概述

8.2 数据库安全性控制

8.3 视图机制

8.4 审计(Audit)

8.5 数据加密

8.6 其他安全性保护



## 8.3 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。
- 间接地实现支持存取谓词的用户权限定义。



# 视图机制

---

[例8.14] 建立软件学生的视图，把对该视图的SELECT权限授予王平，把该视图上的所有操作权限授予张明。

先建立软件学生的视图SE\_Student

```
CREATE VIEW SE_Student  
AS  
SELECT *  
FROM Student  
WHERE Sdept='SE';
```



# 视图机制

---

在视图上进一步定义存取权限

```
GRANT SELECT /*王平只能检索学生的信息 */  
ON SE_Student  
TO 王平;
```

```
GRANT ALL PRIVILIGES /*张明具有检索、增删改的所有权限 */  
ON SE_Student  
TO 张明;
```





# 第8章 数据库安全性

---

8.1 数据库安全性概述

8.2 数据库安全性控制

8.3 视图机制

8.4 审计(Audit)

8.5 数据加密

8.6 其他安全性保护



## 8.4 审计

---

### ➤什么是审计

➤启用一个专用的审计日志(Audit Log)

将用户对数据库的所有操作记录在上面。

➤审计员利用审计日志。

监控数据库中的各种行为，找出非法存取数据的人、时间和内容。

➤C2以上安全级别的DBMS必须具有审计功能。



# 审计

---

## ➤ 审计功能的可选性

➤ 审计很费时间和空间。

➤ DBA可以根据应用对安全性的要求，灵活地打开或关闭审计功能。

➤ 审计功能主要用于安全性要求较高的部门。



# 审计

---

## 1. 审计事件

### ➤ 服务器事件

- 审计数据库服务器发生的事件

### ➤ 系统权限

- 对系统拥有的结构或模式对象进行操作的审计
- 要求该操作的权限是通过系统权限获得的

### ➤ 语句事件

- 对SQL语句，如DDL、DML、DQL及DCL语句的审计

### ➤ 模式对象事件

- 对特定模式对象上进行的SELECT或DML操作的审计



# 审计功能

---

- 基本功能：提供多种审计查阅方式。
- 多套审计规则：一般在初始化设定。
- 提供审计分析和报表功能。
- 审计日志管理功能
  - 防止审计员误删审计记录，审计日志必须先转储后删除
  - 对转储的审计记录文件提供完整性和保密性保护
  - 只允许审计员查阅和转储审计记录，不允许任何用户新增和修改审计记录等
- 提供查询审计设置及审计记录信息的专门视图



# 审计

---

## 3. AUDIT语句和NOAUDIT语句

- AUDIT语句： 设置审计功能
- NOAUDIT语句： 取消审计功能



# 审计

---

## ➤用户级审计

- 任何用户可设置的审计。
- 主要是用户针对自己创建的数据库表和视图进行审计。

## ➤系统级审计

- 只能由数据库管理员设。
- 监测成功或失败的登录要求、监测授权和收回操作以及其他数据库级权限下的操作。



# 审计

---

[例8.15] 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8.16] 取消对SC表的一切审计

```
NOAUDIT ALTER,UPDATE
```

```
ON SC;
```





# 第8章 数据库安全性

---

8.1 数据库安全性概述

8.2 数据库安全性控制

8.3 视图机制

8.4 审计(Audit)

8.5 数据加密

8.6 其他安全性保护



## 8.5 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—明文(Plain text)变换为不可直接识别的格式—密文(Cipher text)。

### ➤ 加密方法

- 存储加密

- 传输加密



# 数据加密

---

## 存储加密

- 透明存储加密
  - 内核级加密保护方式，对用户完全透明。
  - 将数据在写到磁盘时对数据进行加密，授权用户读取数据时再对其进行解密。
  - 数据库的应用程序不需要做任何修改，只需在创建表语句中说明需加密的字段即可。
  - 内核级加密方法: 性能较好，安全完备性较高。
- 非透明存储加密
  - 通过多个加密函数实现。



# 数据加密

---

## ➤ 传输加密

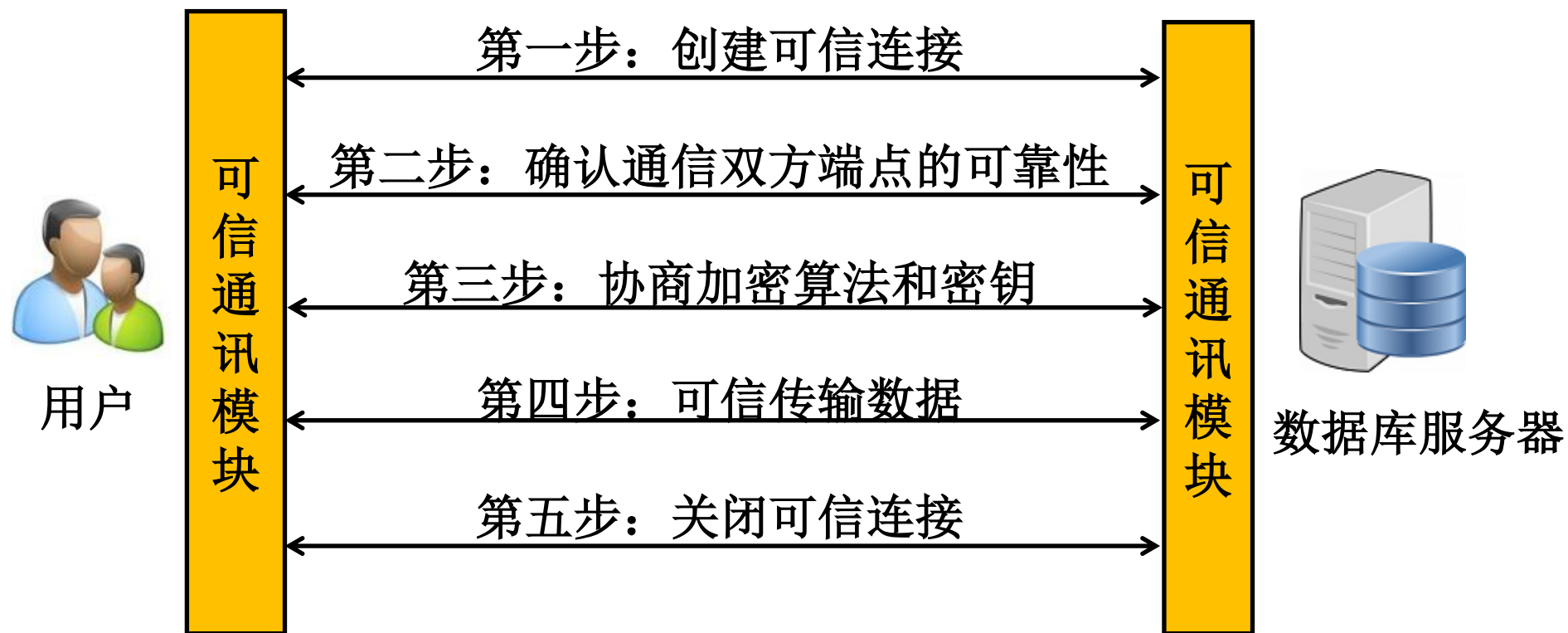
### ➤ 链路加密

- 在链路层进行加密
- 传输信息由报头和报文两部分组成
- 报文和报头均加密

### ➤ 端到端加密

- 在发送端加密，接收端解密
- 只加密报文不加密报头
- 所需密码设备数量相对较少，容易被非法监听者发现并从中获取敏感信息

# 数据加密



数据库管理系统可信传输示意图



# 数据加密

---

- 基于安全套接层协议SSL传输方案的实现思路：
  - (1)确认通信双方端点的可靠性
    - 采用基于数字证书的服务器和客户端认证方式。
    - 通信时均首先向对方提供己方证书，然后使用本地的CA 信任列表和证书撤销列表对接收到的对方证书进行验证。
  - (2)协商加密算法和密钥
    - 确认双方端点的可靠性后，通信双方协商本次会话的加密算法与密钥。



# 数据加密

---

## ➤ (3)可信数据传输

- 业务数据在被发送之前将被用某一组特定的密钥进行加密和消息摘要计算，以密文形式在网络上传输。
- 当业务数据被接收的时候，需用相同一组特定的密钥进行解密和摘要计算。



# 第8章 数据库安全性

---

8.1 数据库安全性概述

8.2 数据库安全性控制

8.3 视图机制

8.4 审计(Audit)

8.5 数据加密

8.6 其他安全性保护





## 8.6 其他安全性保护

---

### ➤推理控制

- 处理强制存取控制未解决的问题。
- 避免用户利用能够访问的数据推知更高密级的数据。
- 常用方法
  - 基于函数依赖的推理控制
  - 基于敏感关联的推理控制

### ➤隐蔽信道

- 处理强制存取控制未解决的问题。



# 推理控制

- 用户利用其被允许的多次查询的结果，结合相关的领域背景知识以及数据之间的约束
  - 推导出其不能访问的数据。
- 例如，某个公司信息系统中假设姓名和职务属于低安全等级(如公开)信息，而工资属于高安全等级(如机密)信息。
- 用户A的安全等级较低，他通过授权可以查询自己的工资、姓名、职务，及其他用户的姓名和职务。
  - 由于工资是机密信息，因此用户A不应知道其他用户的工资。
- 若用户B的职务和用户A相同，则利用函数依赖关系职务->.工资，用户A可通过自己的工资信息(假设3000元)，推出B的工资也是3000元
  - 从而导致高安全等级的敏感信息泄露。



# 隐蔽信道

- 下面的例子就是利用未被强制存取控制的SQL执行后反馈的信息进行间接信息传递。
- 如果insert语句对unique属性列写入重复值，则系统会报错且操作失败。
  - 那么，针对unique约束列，高安全等级用户(发送者)可先向该列插入(或者不插入)数据，而低安全等级用户(接收者)向该列插入相同数据。
- 如果插入失败，则表明发送者已向该列插入数据，此时二者约定发送者传输信息位为0;
- 如果插入成功，则表明发送者未向该列插入数据，此时二者约定发送者传输信息位为1。
- 通过这种方式，高安全等级用户按事先约定方式主动向低安全等级用户传输信息，使得信息流从高安全等级向低安全等级流动，
  - 从而导致高安全等级敏感信息泄露。



# 数据隐私保护

---

## ➤数据隐私保护

- 描述个人控制其不愿他人知道或他人不便知道的个人数据的能力
- 范围很广：数据收集、数据存储、数据处理和数据发布等各个阶段

## ➤例如，在**数据存储阶段**应避免非授权的用户访问个人的隐私数据。

- 通常可以使用数据库安全技术实现这一阶段的隐私保护。
- 如使用自主访问控制、强制访问控制和基于角色的访问控制以及数据加密等。



# 数据隐私保护

---

- 在数据处理阶段，需要考虑数据推理带来的隐私数据泄露。
  - 非授权用户可能通过分析多次查询的结果，或者基于完整性约束信息，推导出其他用户的隐私数据。
- 在数据发布阶段，应使包含隐私的数据发布结果满足特定的安全性标准。
  - 如发布的关系数据表首先不能包含原有表的候选码，同时还要考虑准标识符的影响。



# 安全性

---

- 区块链，提供了一种方式去取代额外的成本，用数字加密学与节点共识的机制来保证数据的安全。
- 区块链的**优点**:
  - **高容错**，因为它内在的冗余机制。
  - 每个节点对于整个区块链网络而言，都不是必不可少的。
  - 节点之间互相通信，即使一些节点因为一些原因出错，**也不会导致整个网络垮掉**。
  - 同时失败的节点，**重启之后也总是能正确同步新的交易**。
- 传统数据库同样有冗余性，但是区块链却将它发挥到了一个新的高度。
  - **任何节点**都可以**自由加入**节点，并且不需要系统做任何的配置。
  - 任何节点都可以**自由离开**这个网络，而不会对网络造成影响。



# 安全性

---

- 区块链除了像传统数据库那样运行，还要做**额外的开销**：
- 签名验证
  - **每一个区块链的交易都需要签名验证**，同时这些签名验证在每个节点间都是要运行的，用于证明交易的合法性。
  - 传统数据库只在第一次通信做一些签名验证，后续请求就不需要。
- 共识机制
  - 节点间要达成共识，需要复杂的通信，并且需要处理分叉。
- 冗余性
  - 传统数据库，主节点更新数据，然后同步数据到从数据库。
  - 区块链**每个节点都需要保存交易数据**。



# 数据库安全性与完整性区别

---

- 数据的完整性和安全性是两个不同概念
- 数据的安全性
  - 保护数据库防止恶意的破坏和非法的存取。
  - 防范对象：非法用户和非法操作。
- 数据的完整性
  - 防止数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据。
  - 防范对象：不合语义的、不正确的数据。





# 课堂练习

---

➤4.6. 对下列两个关系模式:

学生(学号, 姓名, 年龄, 性别, 家庭住址, 班级号)

班级(班级号, 班级名, 班主任, 班长)

使用**GRANT**语句完成下列授权功能:

- (1)授予用户U1对两个表的所有权限, 并可给其他用户授权。
- (2)授予用户U2对学生表具有查看权限, 对家庭住址具有更新权限。
- (3)将对班级表查看权限授予所有用户。
- (4)将对学生表的查询、更新权限授予角色R1。
- (5)将角色R1授予用户U1, 并且U1可继续授权给其他角色。



## 思维简史

1. 经验，直立人，能人，智人，，，，工具，语言，文字
2. 定性，亚里士多德
3. 定量，伽利略，哥白尼，开普勒
4. 推理，亚里士多德，柏拉图，伽利略
5. 动态，瞬时动态，长期累积效应之间的关系，牛顿
6. 极限，理想情况，微积分，交流；牛顿
7. 周期表，化学版的牛顿定律，门捷列夫
8. 随机性，生命世界，进化，达尔文
9. 量子，量化，不连续，原子结构； 爱因斯坦，波尔，普朗克
10. 矩阵表示，（矩阵力学，波动方程，路径积分等价），海森堡，薛定谔，费曼
11. 不确定性，
12. 波粒二象性，薛定谔
13. 组合性，指数次扩增，DNA
14. 异构，不同生物组织结构，计算机系统结构发展
15. 。。。



## 笛卡尔 《方法论》

### 科学研究和发明创造的普适方法：

1. 不盲从，不接受任何自己不清楚的真理。对一个命题要根据自己的判断，确定有无可疑之处。
  1. 不管什么权威的结论，只要没有经过自己的研究，都可以怀疑。
  2. 例如亚里士多德曾说，重的物体比轻的物体下落速度快，但事实并非如此。
2. 对于复杂问题，尽量分解为多个简单的小问题来研究，一个一个地分开解决。化繁为简、化整为零。
3. 解决小问题时，应该按照先易后难的次序，逐步解决。
4. 解决每个小问题之后，再综合起来，看看是否彻底解决了原来的问题。
5. 推广到极限情况，会是什么情况呢？（牛顿）
6. 因果图（）
7. 。。。