西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY 计算机学院 SCHOOL OF COMPUTER SCIENCE

西北工业大学 | Northwestern Polytechnical University

数据库系统原理 | Database System

第四章 数据库安全性

赵晓南

2024.10

课前

- 实验课（头歌平台 & 本地环境）
- OceanBase技术报告与大赛报名
- 在线NPU-SQL-OJ平台

https://open.oceanbase.com/competition?_gl=1*v5nggf*_ga*MTc1NDIyMzE3OS4xNjk0NzY2MzAy*_ga_T35KTM57DZ*MTY5NTYxNDE2OC4yLjEuMTY5NTYxNDE2OC42MC4wLjA.#train

https://www.oceanbase.com/video/9000654

课前复习

数据查询语言

```
Select [ ALL|DISTINCT ] <输出属性列表>
From        <一个或多个数据库表或视图>
[Where      <查询条件> ]
[Group By   <分组条件>[HAVING <条件表达式>] ]
[Order By   <结果排序> [ASC|DESC]
```

1. 单表查询
2. 连接查询
3. 嵌套查询
4. 集合查询
5. 基于派生表的查询

SQL语言所使用的动词

SQL功能	动词
数据定义DDL	CREATE, DROP, ALTER
数据查询DQL	SELECT
数据更新DML	INSERT, UPDATE, DELETE
数据控制DCL	GRANT, REVOKE

本章目录

4.1.计算机安全性概述

4.2.数据库安全性控制

4.3.视图机制

4.4.审计(Audit)

4.5.数据加密

4.6.其他安全性保护

- 数据库的安全性

指保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。

- 计算机系统安全性
  - 技术安全
  - 管理安全
  - 政策法律

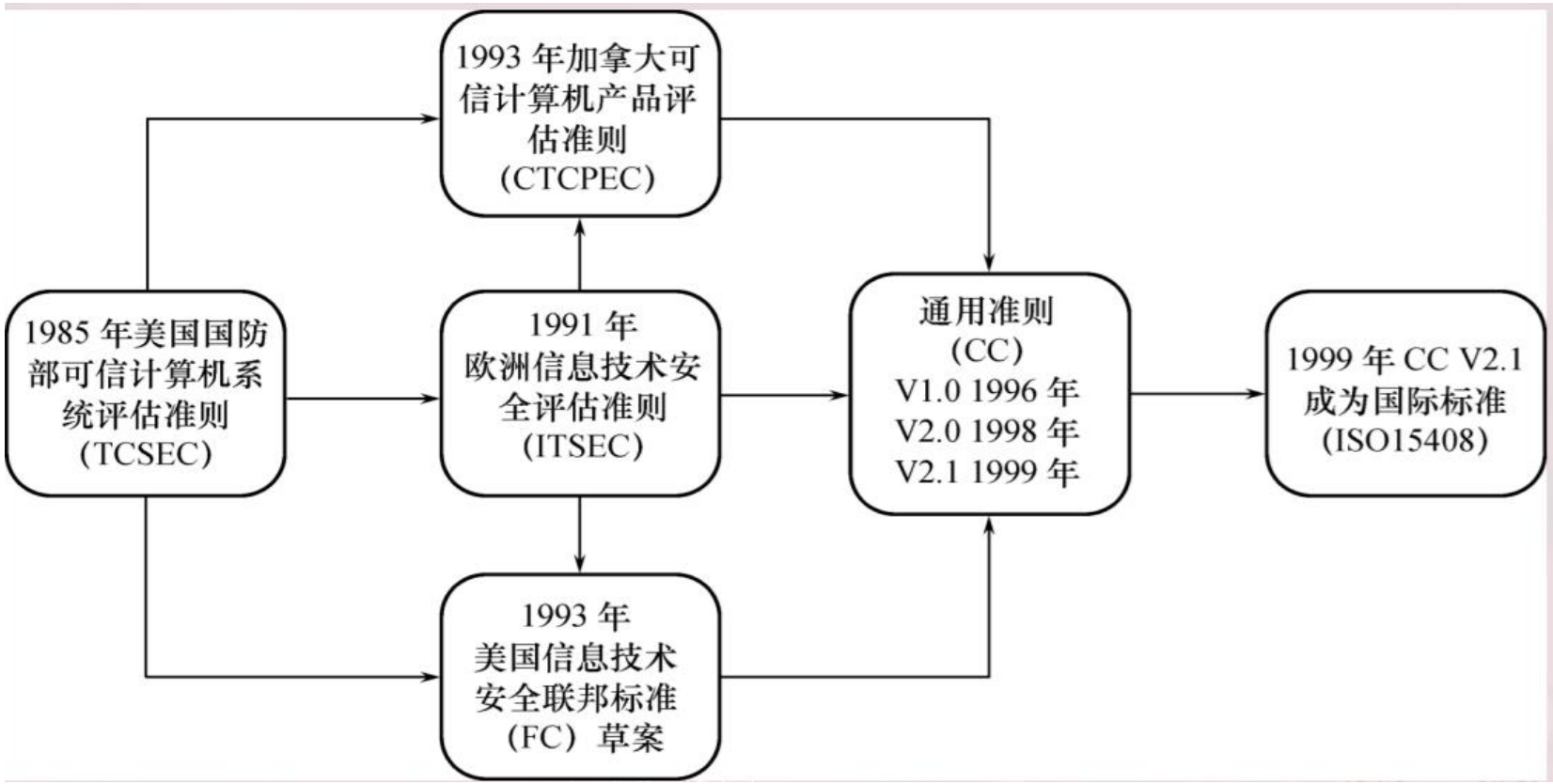
数据库安全与计算机系统的安全性，包括操作系统，网络系统的安全性紧密联系，相互支持

4.1 计算机安全性概述

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 数据的完整性和安全性是两个不同概念
  - 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
  - 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

4.1 计算机安全性概述



信息安全发展历程

Quiz Time

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

《中华人民共和国数据安全法》是为了规范数据处理活动，保障数据安全，促进数据开发利用，保护个人、组织的合法权益，维护国家主权、安全和发展利益，制定的法律。

《中华人民共和国数据安全法》自（　　）起施行。

1）2020/9/1

2）2021/9/1

3）2022/9/1

本法所称数据，是指任何以电子或者其他方式对信息的记录。数据处理，包括数据的收集、存储、使用、加工、传输、提供、公开等。数据安全，是指通过采取必要措施，确保数据处于有效保护和合法利用的状态，以及具备保障持续安全状态的能力。

安全标准简介

- TCSEC（Trusted Computer System Evaluation Criteria,桔皮书）
  TDI(Trusted Database Interpretation，紫皮书)
  安全性级别划分标准：安全策略，责任，保证和文档

TDI安全级别划分

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

TCSEC/TDI安全级别划分

安全指标	安全策略							责任			保证									文档			
	自主存取控制	客体重用	标记完整性	标记信息的扩散	主体敏感度标记	设备标记	强制存取控制	标识与鉴别	可信路径	审计	系统体系结构	系统完整性	屏蔽信道分析	可信设施管理	可信恢复	可安全测试	设计规范和验证	配置管理	可信分配	安全特性用户指南	可信设施手册	测试文档	设计文档
C1	■							■			■	■				■				■	■	■	■
C2	▨	■						▨		■	■	□				▨				□	▨	□	□
B1	□	□	■	■			■	▨		▨	▨	□				▨	■			□	▨	□	▨
B2	□	□	□	□	■	■	▨	□	■	▨	▨	□	■	■		▨	▨	■		□	▨	▨	▨
B3	▨	□	□	□	□	□	□	□	▨	▨	▨	□	▨	▨	■	▨	▨	□		□	▨	□	▨
A1	□	□	□	□	□	□	□	□	□	□	□	□	▨	□	□	▨	▨	▨	■	□	□	▨	▨

表 4.2　不同安全级别对安全指标的支持情况

□ 不支持　▨ 支持较下一级有增/改　□ 同相邻低第一级　■ 新增支持

安全标准简介

- CC（Common Criteria）：安全要求：安全功能要求，安全保证要求

CC评估保证级划分

评估保证级	定　义	TDI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

本章目录

4.1.计算机安全性概述

4.2.数据库安全性控制

4.3.视图机制

4.4.审计(Audit)

4.5.数据加密

4.6.其他安全性保护

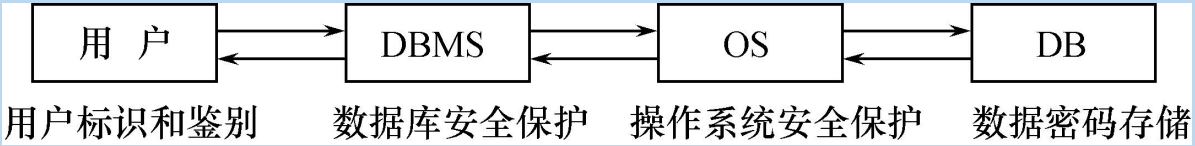
非法使用数据库的情况

- 编写一段合法的程序绕过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。

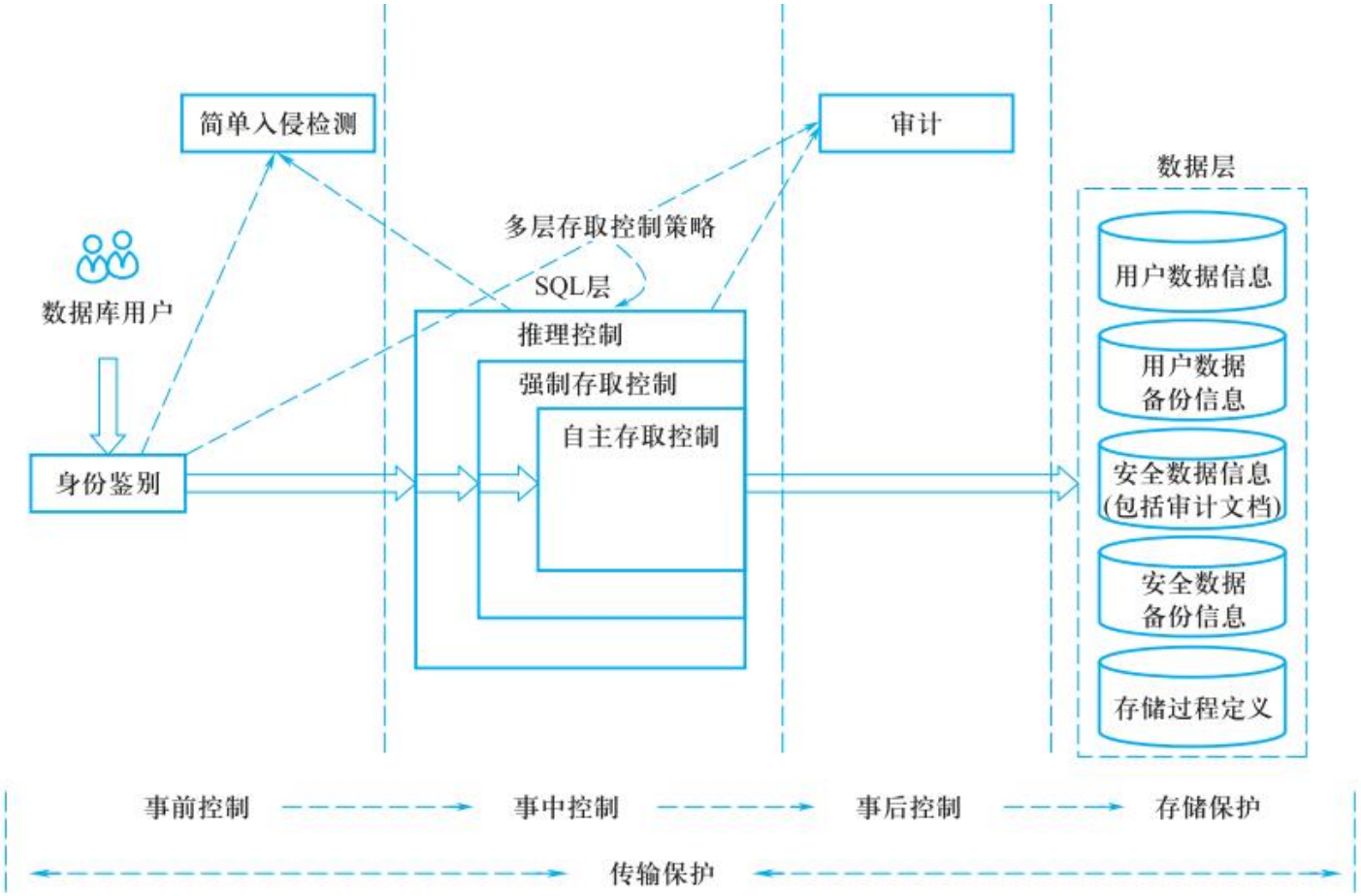
数据库安全性控制常用方法

- 用户标识和鉴定
- 存取控制
- 视图
- 审计
- 数据加密



计算机系统的安全模型

4.2 数据库安全性控制



数据库管理系统的安全性控制模型

Application Security公司提出的十大数据库漏洞：

1. 默认、空白及弱用户名/密码
2. SQL注入
3. 广泛的用户和组权限
4. 启用不必要的数据库功能
5. 失效的配置管理
6. 缓冲区溢出
7. 特权升级
8. 拒绝服务攻击
9. 数据库未打补丁
10. 敏感数据未加密

http://www.ok165.com/article/6670.html

4.2 数据库安全性控制 — SQL注入

- SQL Injection 的起因通常是因为程序采用了构造字符串的方式拼接SQL命令。
- 利用正常查询网站时，将攻击指令藏于查询指令中。
- 攻击者可穿透防火墙，绕过身份认证机制，取得资料库使用权限，进而窃取资料或窜改。破坏资料。

4.2 数据库安全性控制 — SQL注入

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 一般输入帐号密码的网站SQL语法：

```
select * from member where UID =’  ″& request(″ID″) &″  ’
                and Passwd =’ ″& request(″Pwd″) & ″ ’
```

✓ 如果正常使用者帐号A123456789 ， 密码1234

```
select * from member where UID =’A123456789’
    and  Passwd=’1234’
```

通常输入的帐号和密码等信息会取代Web程序中的变量，并由两个单引号(’ ’)所包住。

4.2 数据库安全性控制 — SQL注入

西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

若攻击者已知系统中已有一个Admin的管理者帐号，则输入

Admin ‘-- ，即可不须输入密码而进入资料库

select * from member where UID =’ Admin ’-- ’

And Passwd =’ ’

注：-- 符号后的任何叙述都会被当作注释

（以上面为例，And子句将被SQL视为注释语句）

```
mysql> select * from userinfo where id='Admin'-- and pwd='12345';
    -> ;
+-------+------+
| id    | pwd  |
+-------+------+
| Admin | 123  |
+-------+------+
1 row in set (0.00 sec)
```

4.2 数据库安全性控制 — SQL注入


```
select * from table  where name= ‘ “+un+” ’
```

假设un输入：12345’ or 1=1

```
  select * from table    where name=’12345’ or 1=1
```

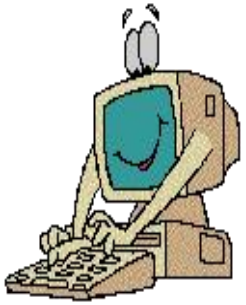
假设un输入:xxx; DROP TABLE SiteUsers WHERE 1=1

```
 select * from table
 where name=xxx; DROP TABLE SiteUsers WHERE 1=1
```

思考：如何进行SQL注入的预防？

用户标识与鉴定(Identification & Authentication)

- 静态口令
- 动态口令
- 生物特征（指纹、刷脸、虹膜等）
- 智能卡
- 入侵检测



存取控制

- 定义用户权限
- 合法权限检查

- 常用存取控制方

1）自主存取控制（Discretionary Access Control，简称DAC）
C2级，灵活

2）强制存取控制（Mandatory Access Control，简称 MAC）
B1级，严格

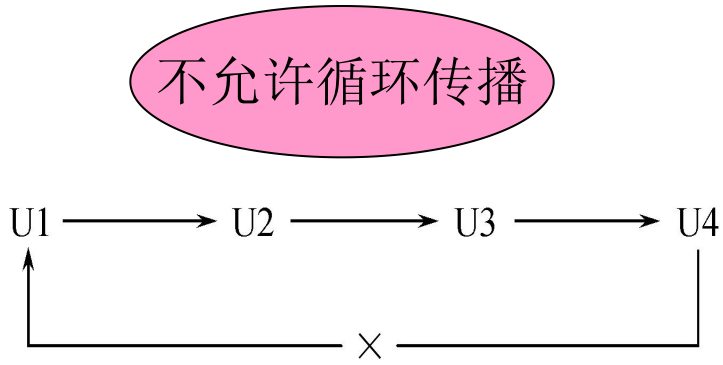
自主存取控制方法(DAC)

- 通过GRANT和REVOKE语句实现。
- 主要操作对象如下表：

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

4.2 数据库安全性控制 - DCL

- 授权
  - GRANT <权限>[，权限]...
    [ON <对象类型> <对象名>]
    TO <用户>[，<用户>]...
    [WITH GRANT OPTION]



1. DBA拥有数据库操作的所有权限，他可以将权限赋予其他用户。
2. 建立数据库对象用户称为该对象的属主(OWNER)，他拥有该对象的所有操作权限。
3. 接受权限的用户可以是一个或多个具体用户，也可以是全体用户(PUBLIC)。
4. WITH GRANT OPTION：决定是否有传播权限的权利

4.2 数据库安全性控制 - DCL


```
create user 'user1'@'localhost' identified by '123456';
```

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 PRIVILIGES
ON TABLE Student, Course
TO U2, U3
```

```
GRANT ALL
ON TABLE Course
TO U2, U3
```

```
GRANT ALL
ON TABLE Student
TO U2, U3
```

例3 把对表SC的查询权限授予所有用户

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

MySQL不支持public关键字
（MySQL如何实现同等功能？）

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          -->U5--> U6--> U7
```

例6 DBA把在数据库S_C中建立表的权限授予用户U8

```
GRANT CREATETABLE
ON DATABASE S_C
TO U8
```

- 收回权限

  - 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>]...
    [CASCADE|RESTRICT]

[例7] 把用户U4修改学生学号的权限收回。

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

- 收回权限

[例8] 收回所有用户对表SC的查询权限。

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

MySQL不支持public关键字

[例9] 把用户U5对SC表的INSERT权限收回

```
REVOKE    INSERT
ON        TABLE    SC
FROM      U5 ;
```

系统将收回直接或间接从U5处获得的对SC表的INSERT权限:

-->U5--> U6--> U7

收回U5、U6、U7获得的对SC表的INSERT权限:

<--U5<-- U6<-- U7

4.2 MySQL的权限控制

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

- MySQL中的权限相关概念
  - 用户名（user）：访问数据库用户名
  - 角色（role）：用户权限的组合 （主体集合）

相关SQL语句：

```
create user u1 identified by '123';
create role r1;
grant  select on student.* to 'r1'@'%';
grant  'r1'@'%' to 'u1'@'%';
flush privileges ;
```

1.MySQL设置权限后需要刷新：

```
flush privileges ;
```

2.ROLE需要激活后权限才生效：

SET DEFAULT ROLE命令

或者

```
SET global
activate_all_roles_on_login=ON
```

查看所有用户：select user,host from mysql.user;

查看某个用户权限：show grants for 'user1'@'localhost'

在MySQL中的应用：

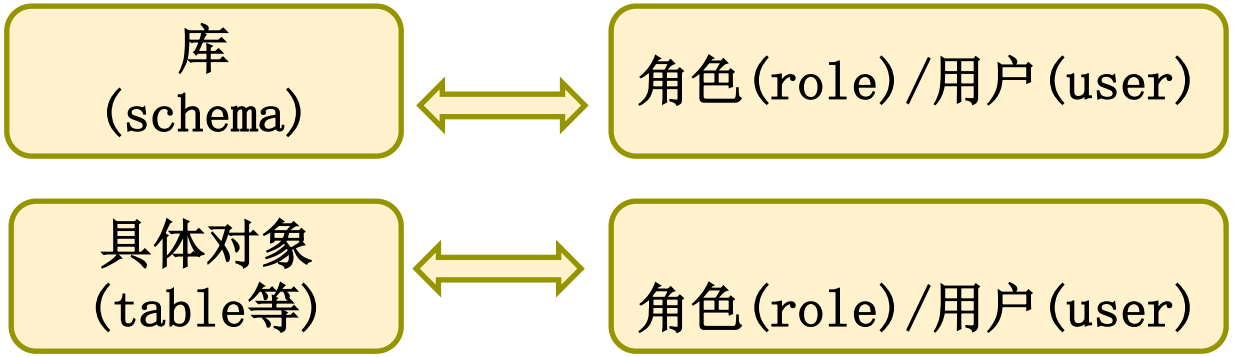
例1：对于表s：授予角色r2具有所有权限，u1具有查询、插入和修改学生系别的权限，u1的查询权限可以传播。
给角色r1授予对所有表的查询权限

```
grant all on s to r2;
grant update on s to 'u1'@'%' with grant option;
grant update on s to 'u1'@'localhost' with grant option;
grant update on s to 'u1'@'192.168.1.108' with grant option;
grant insert, update(sdept) on s to u1;
grant select on *.* to r1;
```

例2：对于表s：收回用户r2的所有权限，收回u1的查询权限。

```
revoke all on s from r2
revoke select on student from u1@%
```

- 权限的设定有多个入口



- 当在不同入口设定的权限不冲突时，最终权限为权限叠加
- 当在不同入口设定的权限冲突时，最终的权限为否定的优先

（如：对某张表，user1具有选择权限，而role1拒绝选择权限，user1属于role1，则user1最终为拒绝选择）

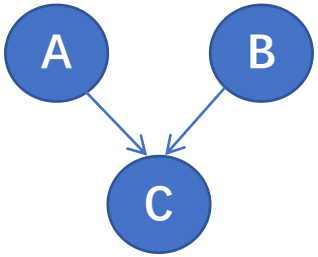
单选题 1分

设置

关于以下Revoke的操作结果是什么？

A授权限X给C，B授相同权限X给C；

当A从C回收权限X后，C是否还有权限X？



A C没有权限X

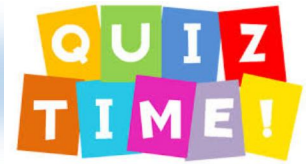
B C仍然具有B赋予的权限X

C 不确定

DBMS如何实现权限管理的功能呢？

提交

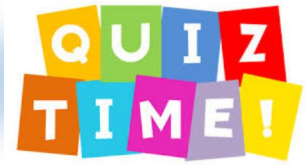
Quiz Time



西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

```
mysql> select * from columns_priv limit 1;
Empty set (0.00 sec)

mysql> grant select, update(grade) on student.sc to 'u2'@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> select * from tables_priv limit 1;
+------+---------+------+------------+---------------+---------------------+------------+-------------+
| Host | Db      | User | Table_name | Grantor       | Timestamp           | Table_priv | Column_priv |
+------+---------+------+------------+---------------+---------------------+------------+-------------+
| %    | student | u2   | sc         | root@localhost | 0000-00-00 00:00:00 | Select     | Update      |
+------+---------+------+------------+---------------+---------------------+------------+-------------+
1 row in set (0.00 sec)

mysql> select * from columns_priv limit 1;
+------+---------+------+------------+-------------+---------------------+-------------+
| Host | Db      | User | Table_name | Column_name | Timestamp           | Column_priv |
+------+---------+------+------------+-------------+---------------------+-------------+
| %    | student | u2   | sc         | grade       | 0000-00-00 00:00:00 | Update      |
+------+---------+------+------------+-------------+---------------------+-------------+
1 row in set (0.00 sec)

mysql> revoke select, update(grade) on student.sc  from  'u2'@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> select * from columns_priv limit 1;
Empty set (0.00 sec)

mysql> select * from tables_priv limit 1;
+-----------+-------+---------------+------------+---------+---------------------+------------+-------------+
| Host      | Db    | User          | Table_name | Grantor | Timestamp           | Table_priv | Column_priv |
+-----------+-------+---------------+------------+---------+---------------------+------------+-------------+
| localhost | mysql | mysql.session | user       | boot@   | 0000-00-00 00:00:00 | Select     |             |
+-----------+-------+---------------+------------+---------+---------------------+------------+-------------+
1 row in set (0.00 sec)
```

Quiz Time



OceanBase的Grant语法

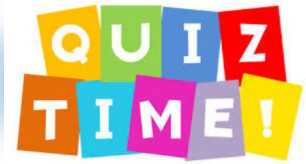
```
| __all_tenant_ols_user_level_history |
| __all_tenant_profile                |
| __all_tenant_profile_history        |
| __all_tenant_role_grantee_map       |
| __all_tenant_role_grantee_map_history |
| __all_tenant_scheduler_job          |
| __all_tenant_scheduler_job_run_detail |
| __all_tenant_scheduler_program      |
| __all_tenant_scheduler_program_argument |
| __all_tenant_security_audit         |
| __all_tenant_security_audit_history |
| __all_tenant_security_audit_record  |
| __all_tenant_sysauth                |
| __all_tenant_sysauth_history        |
| __all_tenant_tablespace             |
| __all_tenant_tablespace_history     |
| __all_tenant_time_zone              |
| __all_tenant_time_zone_name         |
| __all_tenant_time_zone_transition   |
| __all_tenant_time_zone_transition_type |
| __all_tenant_trigger                |
| __all_tenant_trigger_history        |
| __all_tenant_user_failed_login_stat |
| __all_type                          |
| __all_type_attr                     |
| __all_type_attr_history             |
| __all_type_history                  |
| __all_unit                          |
| __all_unit_config                   |
| __all_user                          |
| __all_user_history                  |
| __all_weak_read_service             |
| __all_zone                          |
| __all_zone_merge_info               |
| __tenant_parameter                  |
| t1                                  |
| t2                                  |
+-------------------------------------+
450 rows in set (0.003 sec)

MySQL [oceanbase]> show tables;
```

社区V4.0版：450个系统表（视图）

```
GRANT priv_type
    ON priv_level
    TO user_specification [, user_specification]...
     [WITH with_option ...]

priv_type:
      ALTER
    | CREATE
    | CREATE USER
    | CREATE VIEW
    | DELETE
    | DROP
    | GRANT OPTION
    | INDEX
    | INSERT
    | PROCESS
    | SELECT
    | SHOW DATABASES
    | SHOW VIEW
    | SUPER
    | UPDATE
    | USAGE

priv_level:
      *
    | *.*
    | database_name.*
    | database_name.table_name
    | table_name
    | database_name.rountine_name

user_specification:
user [IDENTIFIED BY [PASSWORD] 'password']

with_option:
 GRANT OPTION
```

https://www.oceanbase.com/docs/common-oceanbase-database-cn-10000000001579167

Quiz Time



西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

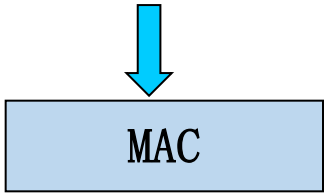
```
MySQL [oceanbase]>
MySQL [oceanbase]>
MySQL [oceanbase]> grant update on testln.sc to obu1;
Query OK, 0 rows affected (0.022 sec)

MySQL [oceanbase]> revoke update on testln.sc from obu1;
Query OK, 0 rows affected (0.019 sec)

MySQL [oceanbase]> grant update(grade) on testln.sc to obu1;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to yo
ar '(grade) on testln.sc to obu1' at line 1
MySQL [oceanbase]>
```

```
MySQL [oceanbase]> select * from __all_table_privilege\G
*************************** 1. row ***************************
       gmt_create: 2023-09-23 15:50:12.297285
     gmt_modified: 2023-09-23 15:50:12.297285
        tenant_id: 0
          user_id: 502929
    database_name: testln
       table_name: sc
       priv_alter: 0
      priv_create: 0
      priv_delete: 0
        priv_drop: 0
priv_grant_option: 0
      priv_insert: 0
      priv_update: 1
      priv_select: 1
       priv_index: 0
 priv_create_view: 0
   priv_show_view: 0
1 row in set (0.000 sec)
```

```
MySQL [oceanbase]> select user_id, user_name from __all_user;
+---------+------------+
| user_id | user_name  |
+---------+------------+
|  502929 | obu1       |
|  200005 | ORAAUDITOR |
|  200001 | root       |
+---------+------------+
3 rows in set (0.000 sec)

MySQL [oceanbase]>
```

OB如何实现对列控制权限呢？

- 自主存取控制(DAC)的缺点
  - 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(MAC)

1）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

2）客体是系统中的被动实体，是受主体操纵的
文件/ 基本表/ 索引/ 视图

4.2 数据库安全性控制 - MAC

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

- 强制存取控制(MAC)

(1)每一个数据对象被标以一定的密级
(2)每一个用户也被授予某一个级别的许可证
(3)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

- 敏感度标记（Level）
  - 绝密（Top Secret）
  - 机密（Secret）
  - 可信（Confidential）
  - 公开（Public）

主体敏感度标记：许可证级别
客体敏感度标记：客体的密级

是对数据本身进行密级标记，无论数据如何复制，标记和数据是一个不可分的整体

目前：DB2，Oracle（基于安全标签的访问控制）

4.2 数据库安全性控制 - MAC

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

MAC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

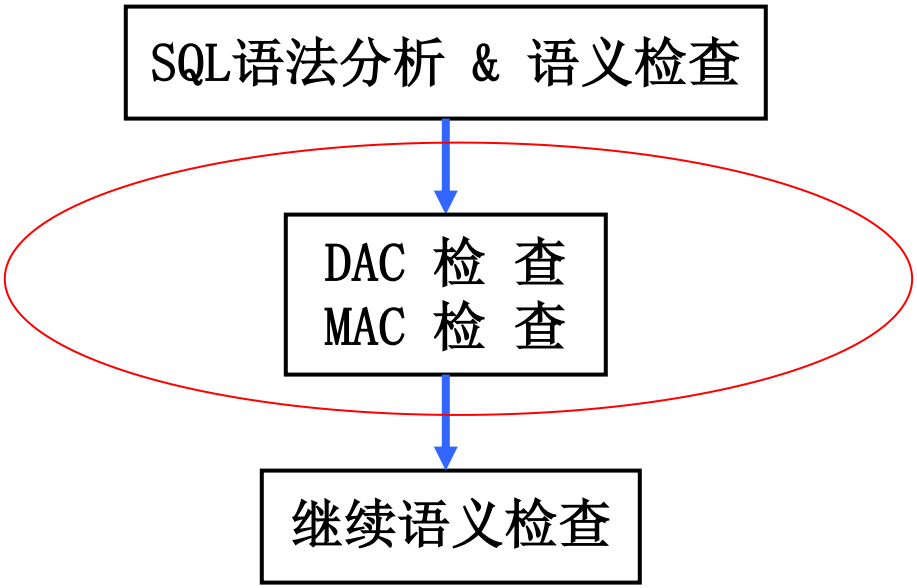
(2)仅当主体的许可证级别等于客体的密级时，该主体才能写相应的客体

- MAC（2）的修正规则

主体的许可证级别 <=客体的密级 → 主体能写客体

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

DBMS的安全机制：DAC+MAC



- 先进行DAC检查，通过DAC检查的数据对象再由系统进行MAC检查，只有通过MAC检查的数据对象方可存取。

本章目录

4.1.计算机安全性概述

4.2.数据库安全性控制

4.3.视图机制

4.4.审计(Audit)

4.5.数据加密

4.6.其他安全性保护

视图的安全保护

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。

- 主要功能是提供数据独立性，其安全保护功能不够精细，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]建立主修计算机专业学生的视图，把对该视图的SELECT权限授于王平，把该视图上的所有操作权限授于张明

1）先建立计算机专业学生的视图CS_Student

```
CREATE VIEW CS_Student AS
SELECT Sno, Sname
FROM   Student
WHERE  Smajor= ‘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;
```

本章目录

4.1.计算机安全性概述

4.2.数据库安全性控制

4.3.视图机制

4.4.审计(Audit)

4.5.数据加密

4.6.其他安全性保护

4.4 审计

- 什么是审计
  - 启用一个专用的审计日志（Audit Log）
    用途：将用户对数据库的**所有操作记录在上面**
  - DBA可以利用审计日志中的追踪信息找出非法存取数据的人
  - C2以上安全级别的DBMS必须具有审计功能

- 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。

- 审计功能的关键字
  - AUDIT/NOAUDIT（教材提及）
    - 对表 t1 的所有 INSERT、UPDATE 和 DELETE 操作进行审计。

```
1. obclient> AUDIT INSERT, UPDATE, DELETE on t1;
2. Query OK, 0 rows affected (31.62 sec)
```

4.4 审计

- MySQL中的审计日志：社区版没有审计功能

审计日志例子：

```
[root@smiletest mysql]# tail -f server_audit.log
20190903 09:24:18,smiletest,root,localhost,2,0,FAILED_CONNECT,,,1045
20190903 09:24:23,smiletest,root,localhost,3,0,CONNECT,,,0
20190903 09:24:23,smiletest,root,localhost,3,3,QUERY,,'select @@version_comment limit 1',0
20190903 09:24:25,smiletest,root,localhost,3,4,QUERY,,'SHOW VARIABLES LIKE \'%audit%\'',0
20190903 09:24:45,smiletest,root,localhost,3,5,QUERY,,'show databases',0
20190903 09:25:12,smiletest,root,localhost,3,0,DISCONNECT,,,0
```

MySQL中的其他日志（非审计日志）

1：重做日志（redo log）

2：回滚日志（undo log）

3：二进制日志（binlog）

4：错误日志（errorlog）

5：慢查询日志（slow query log）

6：一般查询日志（general log）

7：中继日志（relay log）。

本章目录

4.1.计算机安全性概述
4.2.数据库安全性控制
4.3.视图机制
4.4.审计(Audit)
4.5.数据加密
4.6.其他安全性保护

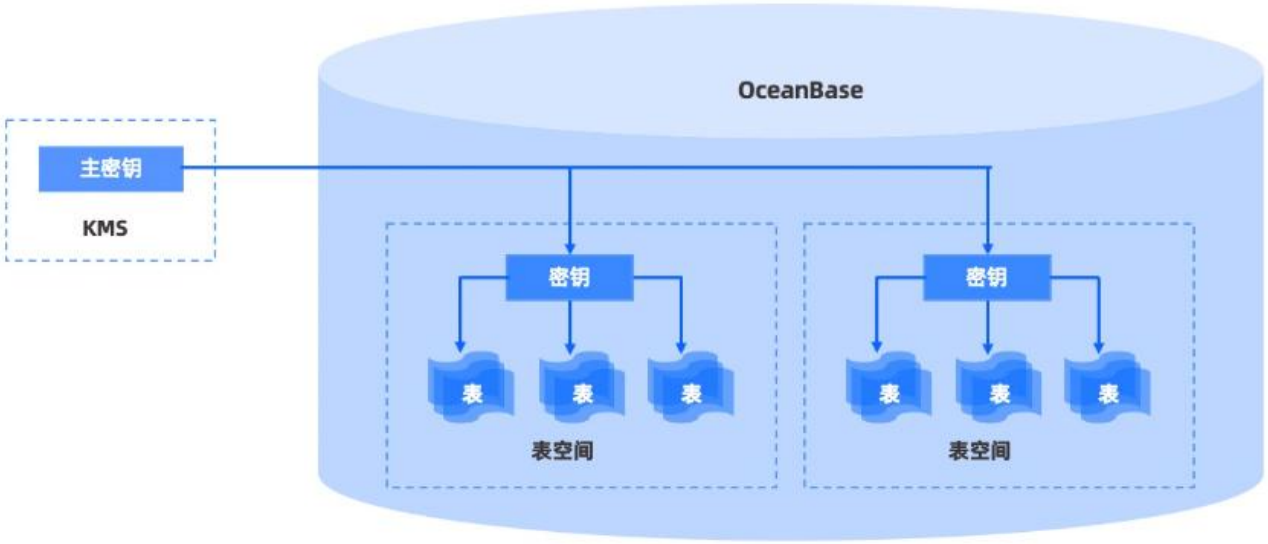
4.5 数据加密

- **数据加密（存储加密 + 传输加密）**
  - 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）
  - 不知道解密算法的人无法获知数据的内容。

- **加密的基本方法**
  - 替换方法：使用密钥（Encryption Key）将明文中的每一个字符转换为密文中的一个字符
  - 置换方法：将明文的字符按不同的顺序重新排列
  - 混合方法：美国1977年制定的官方加密标准：数据加密标准

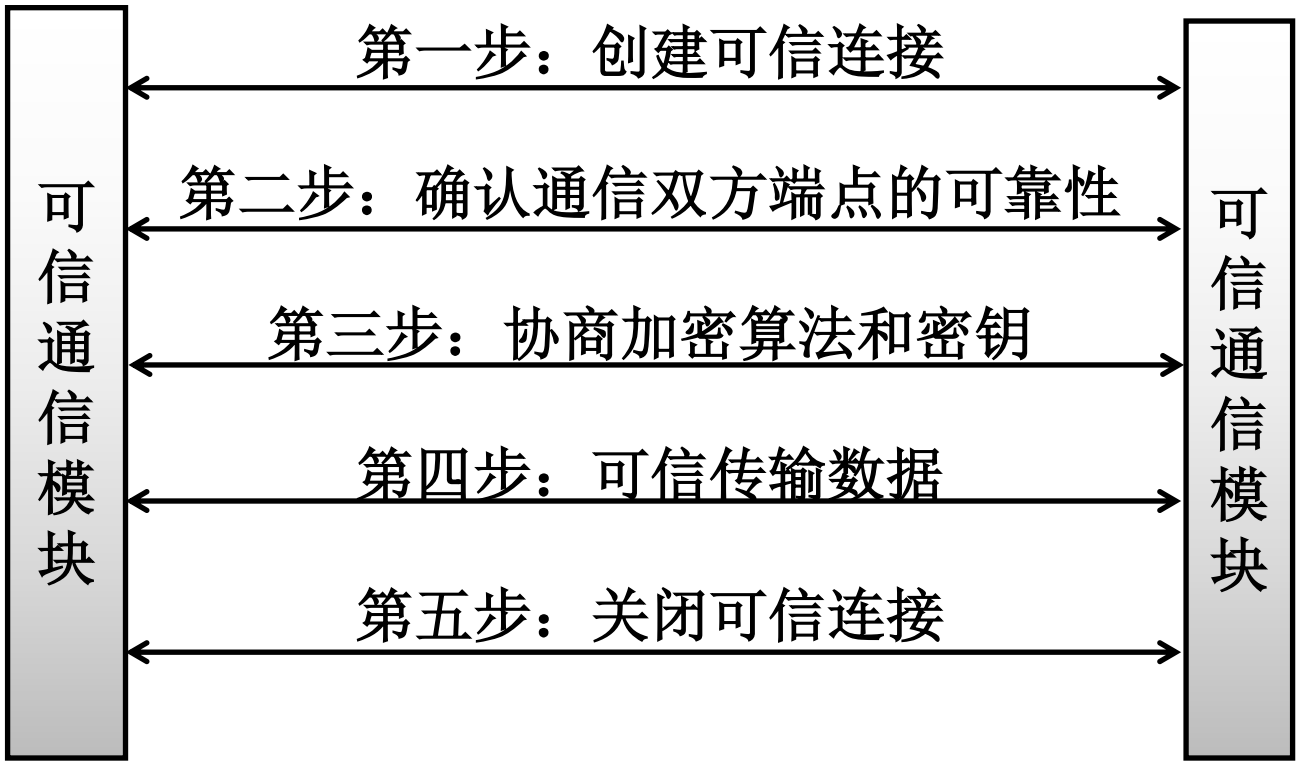
OceanBase中的数据加密 —— 存储加密



- 透明数据加密使用两级密钥体系实现加解密功能
- 开启加密的最小粒度为数据库中的一个表
- 需要开启加密的表需要放到一个加密的表空间（tablespace）中。

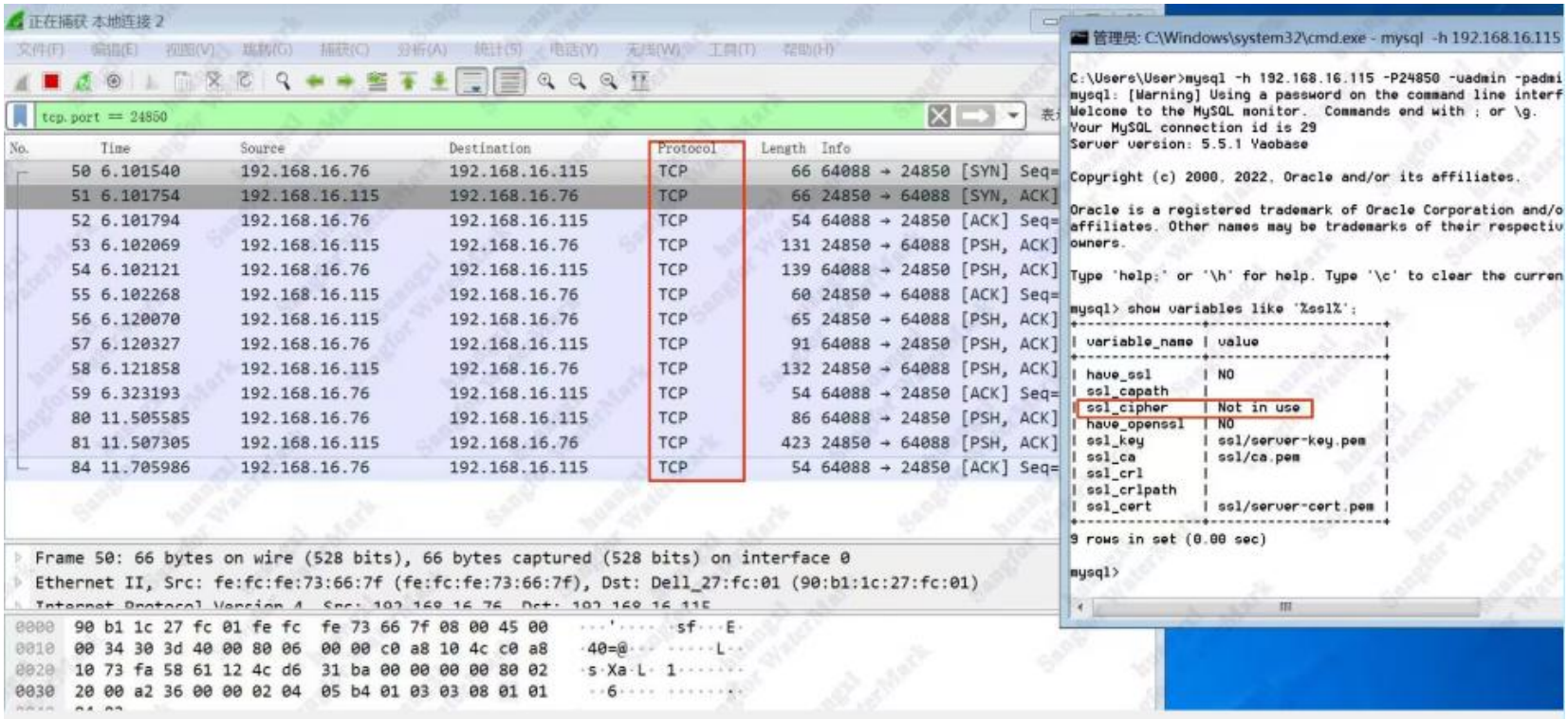
https://www.oceanbase.com/docs/common-oceanbase-database-cn-1000000000033718

4.5 数据加密 - 传输加密



基于 SSL protol的数据库管理系统可信传输示意图

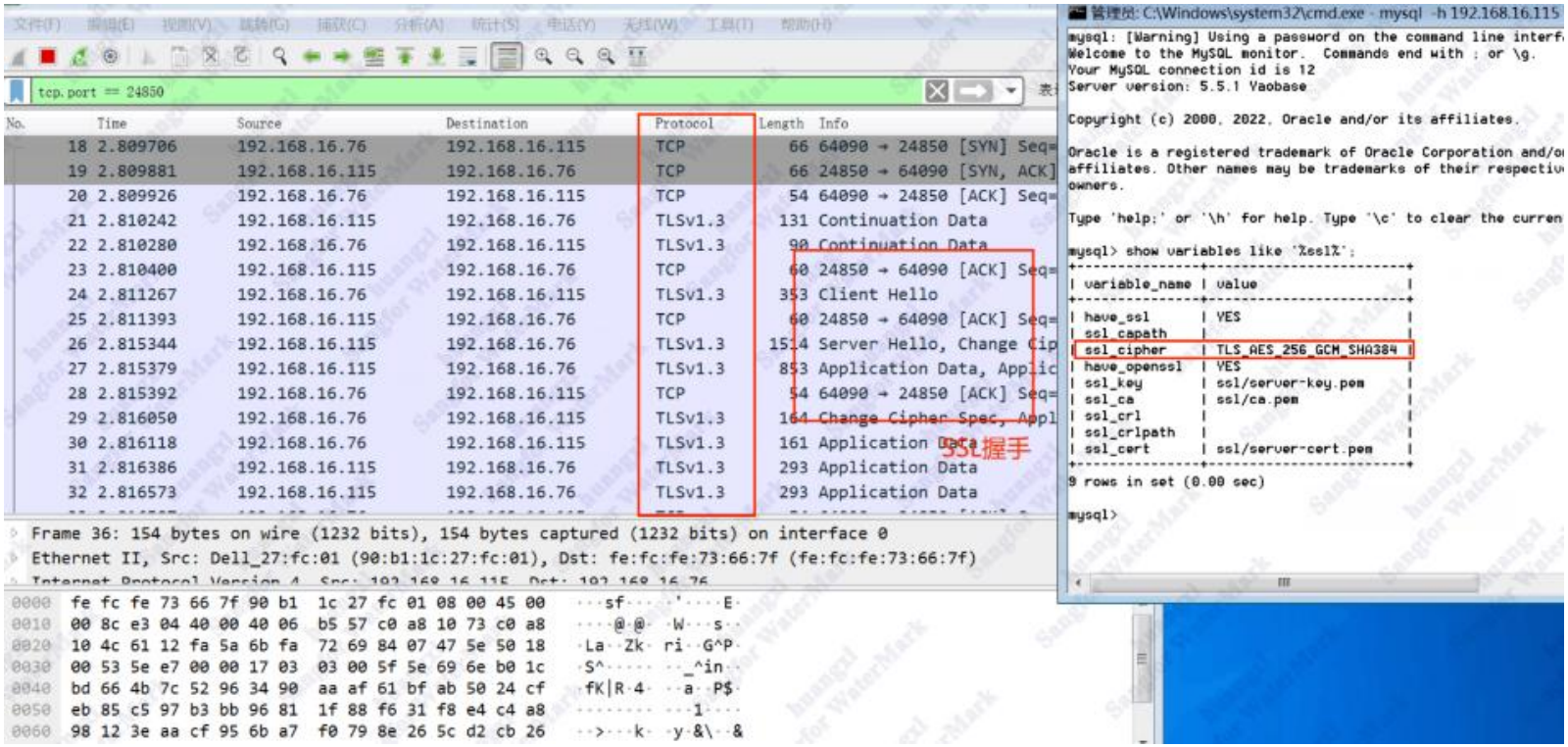
4.5 数据加密 - MYSQL传输加密



MYSQL 未开启SSL连接（普通连接模式）

4.5 数据加密 - MYSQL传输加密

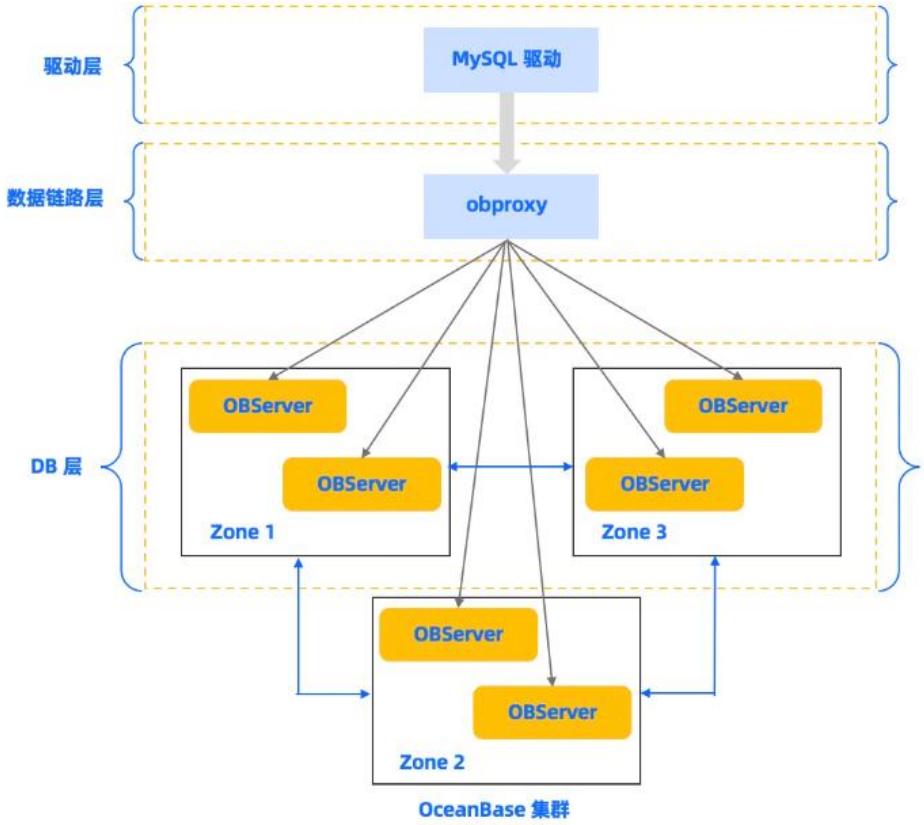

```
[root@k8s-master ca]# mysql -u usessl --ssl-ca=ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.31 MySQL Community Server - GPL

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

You are enforcing ssl connection via unix socket. Please consider
switching ssl off as it does not make connection via unix socket
any more secure.
mysql>
```



MYSQL 开启SSL连接（安全加密模式）

OceanBase中的数据加密 —— 传输加密

- 安全传输层协议（TLS）用于在两个通信应用程序之间提供保密性和数据完整性。



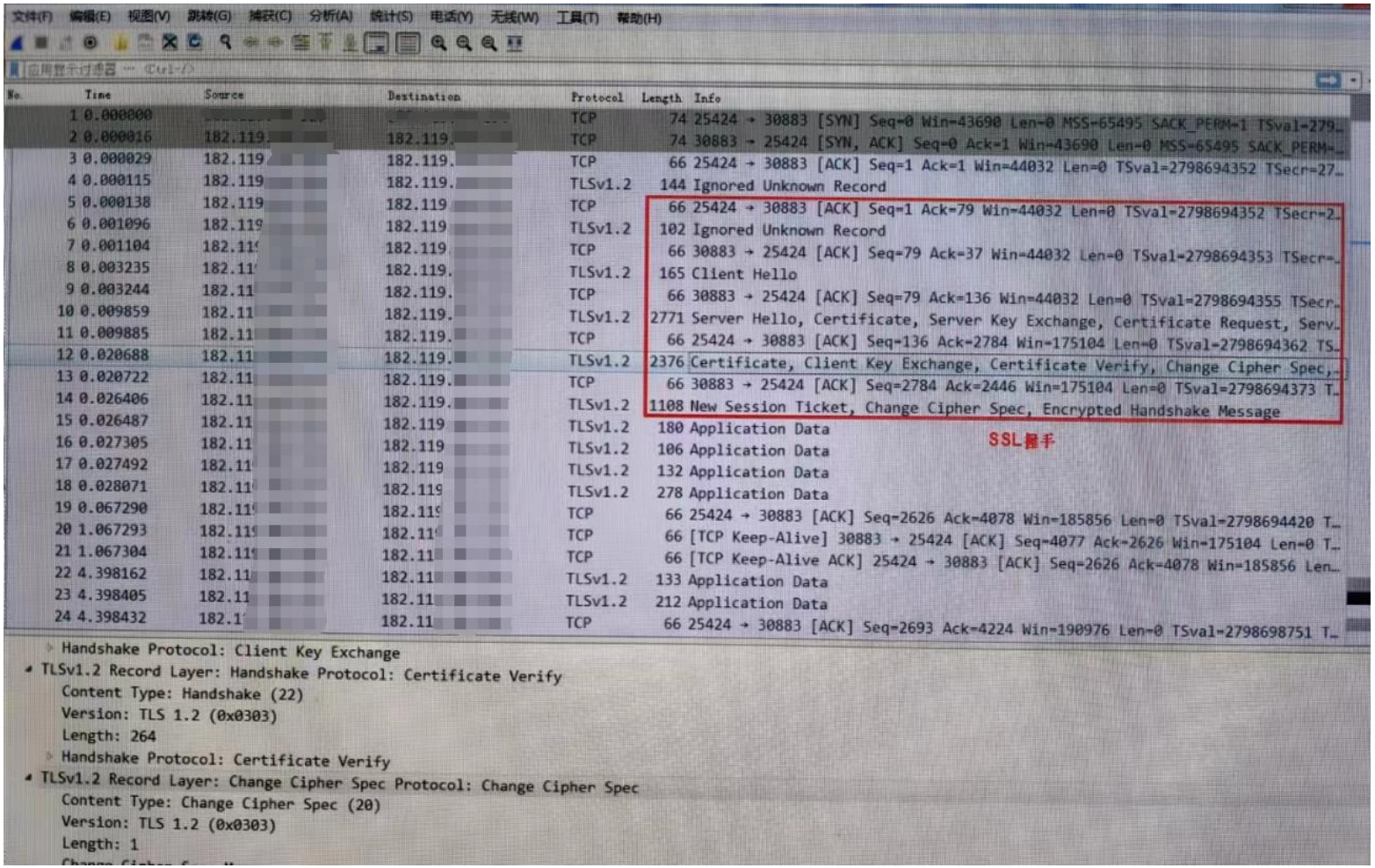
- MySQL 协议（扩展版）：
  - 驱动层 <=> 数据链路层
  - 数据链路层 <=> DB 层
- OB-RPC 协议（OB自有）：
  - 节点与节点之间
  - 节点与 liboblog、ob_admin 等之间

https://www.oceanbase.com/docs/common-oceanbase-database-cn-1000000000033720

4.5 数据加密 - 数据库OB内核

西北工业大学 NORTHWESTERN POLYTECHNICAL UNIVERSITY

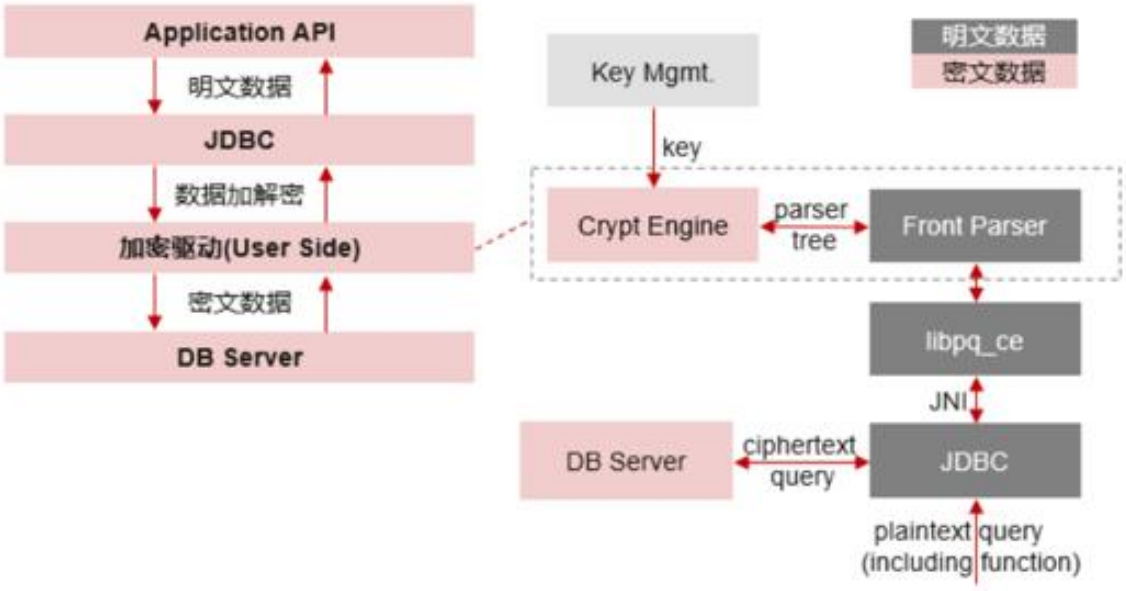
OceanBase中的数据加密 —— 传输加密



tcpdump工具抓包验证

4.5 数据加密 - 华为全密态数据库

openGauss全密态数据库



核心：客户端解析用户全部的输入输出语句，识别出已定义的敏感数据，并且进行自动化加解密。用户使用全密态数据库时，从输入语法，客户端发送到服务端，服务端执行并将结果返回给客户端的整个流程中，用户唯一能感知到的只有输入语法以及获得返回结果这两步。

https://blog.csdn.net/GaussDB/article/details/120843218?spm=1001.2101.3001.6650.7&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-7-120843218-blog-121101435.pc_relevant_multi_platform_whitelistv6&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-7-120843218-blog-121101435.pc_relevant_multi_platform_whitelistv6&utm_relevant_index=8

本章目录

4.1.计算机安全性概述
4.2.数据库安全性控制
4.3.视图机制
4.4.审计(Audit)
4.5.数据加密
4.6.其他安全性保护

4.6 其他安全性保护

- **推理控制机制（统计数据安全）**
  - 基于函数依赖的推理
  - 基于敏感关联的推理

- **隐蔽信道**

  经过隐蔽信息通道（如SQL执行后的反馈信息）获取信息

- **数据隐私(data privacy)**
  - 控制不愿或不便他人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阶段

- **“三权分立”的安全管理机制**

  解决数据库管理员权限过于集中的问题，遵照GB/T20273-2019，引进“三权分立”的安全管理机制

本章重点与难点

重点

- 数据库安全性实现的方法
- 自主存取控制（DAC）与强制存取控制（MAC）
- 授权（Authorization）与回收(GRANT&REVOKE)
- 数据库角色

难点

- 强制存取控制（MAC）的理解

作业： 实验预习：第4章：6,7,8。
(不用书面提交，实验预习)