2026年春季学期

# 数据库系统概论

# An Introduction to Database Systems

## 第四章 数据库安全性

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn


# 4.2 数据库安全性控制



4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法


# 4.2.2 存取控制



- 存取控制机制的组成
  - 定义用户权限
    - 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DBMS的安全子系统

# 存取控制（续）



- 常用存取控制方法
  - 自主存取控制（Discretionary Access Control，DAC）
    - C2级
    - 灵活：用户自主（用户）
      - 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
  - 强制存取控制（Mandatory Access Control，MAC）
    - B1级
    - 严格：系统强制（数据）
      - 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


# 4.2 数据库安全性控制



4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法


# 4.2.3 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（续）

- 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<br>ALL PRIVILEGES |

关系数据库系统中的存取权限

在对用户授权列**INSERT**权限时，一定要包含对主码的**INSERT**权限，否则用户的插入会因为控制而被拒绝。除了授权的列，其他列的值或者取空，或者取默认值。

在对用户授权列**UPDATE**一列的权限时，用户修改该列仍然要遵守创建时定义的主码和其他约束。


# 4.2 数据库安全性控制



4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法

# 4.2.4 授权与收回



## 一、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


# GRANT（续）



- 发出GRANT：
  - DBA(数据库管理员，mysql中的root)
  - 数据库对象创建者（即属主Owner）
  - 拥有该权限的用户

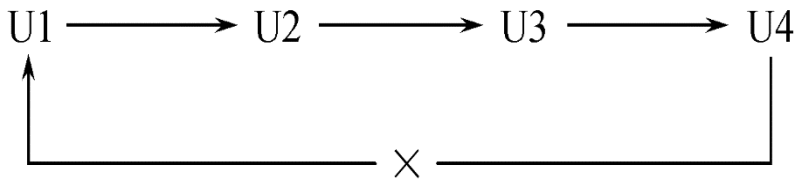
- 按受权限的用户
  - 一个或多个具体用户
  - PUBLIC（全体用户）

# WITH GRANT OPTION子句



- **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4. 1］把查询Student表权限授给用户U1

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

[例4.2] 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;
```

# 例题（续）



**[例4.3] 把对表SC的查询权限授予所有用户**

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

# 例题（续）



[例4.4] 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（续）



**[例4.5] 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4.5后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限：

[例4.6] GRANT INSERT ON TABLE SC TO U6
WITH GRANT OPTION;

同样，U6还可以将此权限授予U7：

[例4.7] GRANT INSERT ON TABLE SC TO U7;

但U7不能再传播此权限。

# 传播权限（续）



执行了［例4.1］到［例4.7］的语句后，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 |

# 授权与收回（续）



## 二、REVOKE

- 授予的权限可以由DBA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>]

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


# REVOKE（续）



**[例4.8] 把用户U4修改学生学号的权限收回**

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

**[例4.9] 收回所有用户对表SC的查询权限**

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

# REVOKE（续）



[例4.10] 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
  - U5—U6—U7，Ux—U6—U7

# REVOKE（续）

执行［例4.8］到［例4.10］的语句后，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



- 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

课后尝试MYSQL的授权方式

# 授权与收回（续）



*三、创建数据库的权限

- 创建数据库也需要进行安全控制，也是要授权的
- 但是在SQL标准中没有创建数据库的标准定义，因此各个数据库系统实现的语句不同

  下面仅以金仓数据库KingabseES为例做简要说明

  - CREATE USER语句创建超级用户，再由超级用户创建具有CREATE数据库权限的用户
  - CREATE USER语句格式

    CREATE USER <username> [WITH]

    [SUPERUSER | CREATEDB] | PASSWORD 'password';

# 授权与收回（续）



- **CREATE USER语句格式**
  - **具有SUPERUSER权限的用户是系统的超级用户，在系统中跳过权限检查，可以执行任何操作。**
  - **具有CREATEDB权限的用户可以创建数据库，成为数据库的属主，具有在数据库上创建模式的权限，并可以把这些权限授予其他用户**
  - **注意：超级用户是系统初始化时指定的。**
    - **例如，在安装系统时可以指定一个名称为system的超级用户。**

# 授权与收回（续）



**[例4.11] 创建超级用户system2**

**首先以超级用户system登录，然后创建system2:**

```
CREATE USER system2
WITH SUPERUSER PASSWORD '123456';
```

**[例4.12] 创建具有CREATEDB权限的用户U1和普通用户U2**

**以超级用户system登录，创建用户U1, U2如下：**

```
CREATE USER U1 WITH CREATEDB PASSWORD '123456'
/* U1 具有创建数据库的权限了 */
CREATE USER U2 PASSWORD '123456';    /* U2 是普通用户*/
```

# 授权与收回（续）



**[例4.13] 创建数据库U1DB**

**以U1用户登录，创建数据库U1DB:**

```
CREATE DATABASE U1DB;        / *  U1创建了数据库 * /
```

**U1成为数据库U1DB的属主，它可以在U1DB数据库上创建SCHEMA**


# 课后尝试MYSQL的权限管理



- **MySQL 默认有个root用户，权限太大，在管理数据库时候才用**
- **为 MySQL 创建一个新用户：**
  - **CREATE USER username IDENTIFIED BY 'password';**
- **为这个用户分配相应权限**
  - **GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost' IDENTIFIED BY 'password';**
- **授予它在某个数据库上的权限，切换到root 用户撤销刚才的权限，重新授权：**
  - EVOKE ALL PRIVILEGES ON *.* FROM 'username'@'localhost';
  - GRANT ALL PRIVILEGES ON wordpress.* TO 'username'@'localhost' IDENTIFIED BY 'password';
- **定该用户只能执行 select 和 update 命令**
  - GRANT SELECT, UPDATE ON wordpress.* TO 'username'@'localhost' IDENTIFIED BY 'password';

# 课后尝试MYSQL的权限管理



- 查询某个用户的权限
  - Show grants for USERNAME;
  - **select * from mysql.user where user=** USERNAME**;**
- 查询所有用户
  - **select * from mysql.user** # myysql数据库中的用户表
- 查询针对不同对象具有操作权限的用户
  - **数据库级别的权限信息是mysql.db表**
  - **表对象的授权信息记录是mysql.tables_priv表**
  - **列级权限记录在mysql.column_priv表**

# 查询权限

- 查询某个用户的权限

```
1 •  select * from mysql.user;
```

- 

- 

Result Grid | Filter Rows: | Edit: | Export/Import:

| Host | User | Select_priv | Insert_priv |
|---|---|---|---|
| localhost | mysql.infoschema | Y | N |
| localhost | mysql.session | N | N |
| localhost | mysql.sys | N | N |
| localhost | root | Y | Y |
| NULL | NULL | NULL | NULL |


# 4.2 数据库安全性控制



4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法


# 4.2.5 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



- 一、角色的创建

  **CREATE ROLE <角色名>**

- 二、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



- 三、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



- 四、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（续）



[例4.14]　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（续）



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（续）



**[例4.15]　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（续）



**[例4.16]**

**REVOKE SELECT**

**ON TABLE Student**

**FROM R1;**

使R1减少了SELECT权限


# 4.2 数据库安全性控制



4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法


# 存取控制



- 常用存取控制方法
  - 自主存取控制（Discretionary Access Control，简称DAC）
    - 用户可“自主”地决定将数据的存取权限授予何人、决定是否也将“授予”的权限授予别人
      - C2级
      - 灵活
  - 强制存取控制（Mandatory Access Control，简称MAC）
    - 系统“强制”地给用户和数据标记安全等级
      - B1级
      - 严格


# 自主存取控制缺点



- 可能存在数据的“无意泄露”
  - 用户可以授权，用户可以备份数据
- 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

# 4.2.6 强制存取控制方法



- 强制存取控制（MAC)
  - 保证更高程度的安全性
  - 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（续）



- 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

# 强制存取控制方法（续）



- 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）
    - 绝密（Top Secret, TS）
    - 机密（Secret, S）
    - 可信（Confidential, C）
    - 公开（Public, 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（续）



- 强制存取控制规则

  (1)仅当主体S的许可证级别大于或等于客体O的密级时，该主体才能读取相应的客体

  (2)仅当主体S的许可证级别小于或等于客体O的密级时，该主体才能写相应的客体

- 修正（即）规则
  - 主体的许可证级别 =客体的密级 → 主体能写客体

# 强制存取控制方法（续）

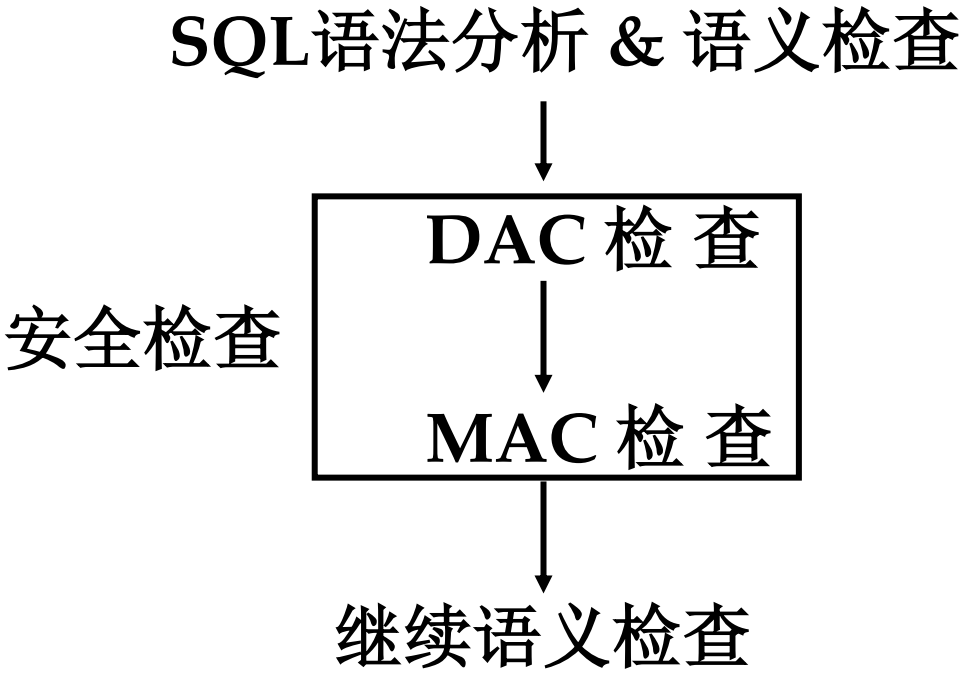


- 强制存取控制（MAC）是对数据本身进行密级标记，无论数据如何复制，标记与数据是一个不可分的整体，只有符合密级标记要求的用户才可以操纵数据。
- 实现MAC时要首先实现DAC
  - 原因：较高安全性级别提供的安全保护要包含较低级别的所有保护
- DAC与MAC共同构成数据库管理系统的安全机制

# 强制存取控制方法（续）



## DAC + MAC安全检查示意图




- 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。


# 第四章 数据库安全性


# 4.3 视图机制



- 视图机制与授权机制配合使用
- 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，无法完全满足要求
  - 间接实现了支持存取谓词的用户权限定义

# 视图机制（续）



**[例4.17]建立计算机系学生的视图，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**

**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';
WITH  CHECK OPTION;
```

# 视图机制（续）



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;
```


# 回顾



- WITH CHECK OPTION
- WITH GRANT OPTION
- WITH ADMIN OPTION
- 各用在什么场景中？


# 4.2 数据库安全性控制

# 4.4 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


# 审计（续）



## 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（续）



- 审计功能
  - 基本功能
    - 提供多种审计查阅方式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（续）



- 审计分为
  - 用户级审计
    - 针对自己创建的数据库表或视图进行审计
    - 记录所有用户对这些表或视图的一切成功和（或）不成功的访问要求以及各种类型的SQL操作
  - 系统级审计
    - DBA设置
    - 监测成功或失败的登录要求
    - 监测GRANT和REVOKE操作以及其他数据库级权限下的操作


# 审计（续）



- AUDIT语句：设置审计功能
- NOAUDIT语句：取消审计功能

# 审计（续）



［例4.18］对修改SC表结构或修改SC表数据的操作进行审计

1.先显示当前审计开关状态

```
SHOW AUDIT_TRAIL;
```

2.打开审计开关

```
SET AUDIT_TRAIL TO ON;
```

3.对SC表设置审计

```
AUDIT ALTER, UPDATE
ON  SC BY ACCESS;
```


# 审计（续）



［例4.19］取消对SC表的ALTER和UPDATE操作审计

```
NOAUDIT ALTER，UPDATE
ON SC;
```


# 审计（续）



- 审计设置和审计日志一般存储在数据字典中。
  - 打开审计开关：系统参数AUDIT_TRAIL设为true
  - 可在系统表SYS_AUDITTRAIL中查看设计信息


# 4.2 数据库安全性控制


# 4.5 数据加密

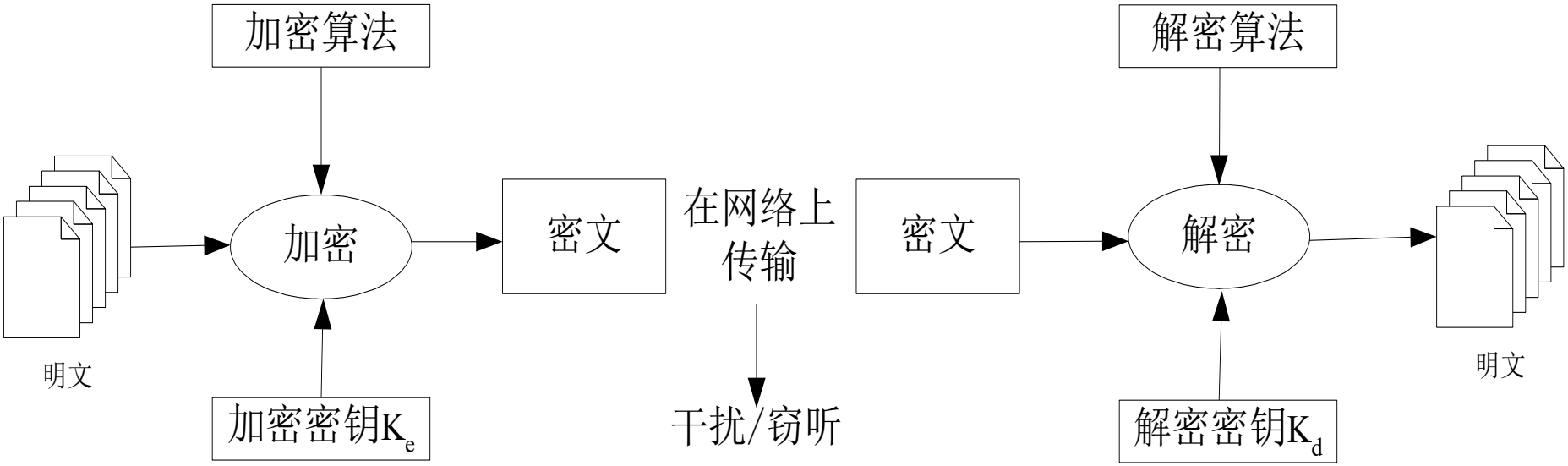


- 数据加密
  - 防止数据库中数据在存储和传输中失密的有效手段
- 加密的基本思想
  - 根据一定的算法将原始数据—明文（Plain text）变换为不可直接识别的格式—密文（Cipher text）
- 加密方法
  - 存储加密
  - 传输加密

# 4.5 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




# 案例一



我画兰江水悠悠，
爱晚亭上枫叶稠，
秋月融融照佛寺，
香烟袅袅绕轻楼.

《唐寅诗集》

# 案例

它是一种代换密码。据说凯撒是率先使用加密函的古代将领之一，因此这种加密方法被称为恺撒密码

密文: jrg oryhv shrsoh

算法: Ci=E(Pi)=Pi+3 恺撒密码

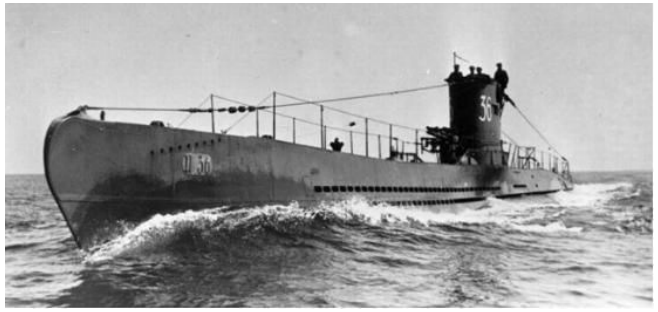
明文: GOD LOVES PEOPLE

**字母表：(密码本)**

ABCDEFGHIJKLMNOPQRSTUVWXYZ

defghijklmnopqrstuvwxyzabc

# 案例三




Alan M. Turing, codebreaker
and an Enigma Code Machine
30p
ST HELENA

恩尼格玛密码机


1940年 英国 布莱切利园
BLETCHLEY PARK, ENGLAND - 1940

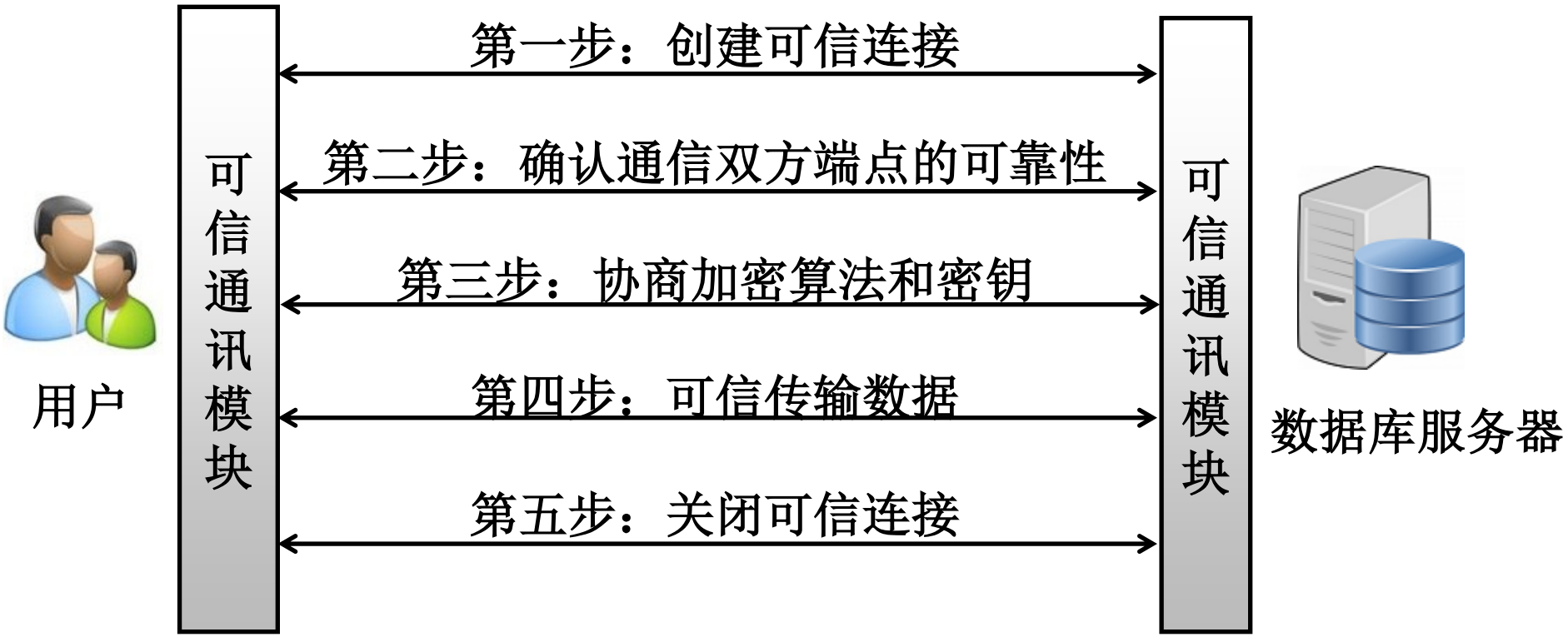
# 4.5 数据加密

- 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
      - 对加密数据进行增删改查时，DBMS自动加解密
    - 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

# 4.5 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

# 4.5 数据加密




数据库管理系统可信传输示意图

# 4.5 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CA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

# 4.5 数据加密

- 基于安全套接层协议SSL传输方案的实现思路：
  - （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
- 密码学
  - 对称加密
  - 非对称加密


# 第四章 数据库安全性


# 其它安全性保护



- 推理控制
  - 避免用户利用其能访问的数据推知更高密级的数据
- 隐蔽信道
- 数据隐私保护
  - 控制不愿被他人知道或他人不便知道的个人数据的能力
  - 存在于数据生命周期的各个阶段
  - 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
  - 加密
  - 算法是否可以保护数据隐私？

# 数据中的隐私泄露



- 大数据比赛
  - 在线电影推荐
  - 数据匿名化处理


# 其它安全性保护



- 大数据时代的数据隐私保护？
  - 差分隐私
  - 联邦学习
  - …

# 其它安全性保护



- 背景：
  - 用户隐私泄露事件多发
  - 隐私保护又愈发收到重视


## 2022年信息泄露事件盘点

| 序号 | 涉事国家/企业 | 事件回顾 | 时间 | 数据规模 |
|---|---|---|---|---|
| 1 | 红十字会总部 | 红十字国际委员会（ICRC）遭遇高级网络攻击，泄露了超过50万人的数据 | 2022.01 | 50万人的个人和机密数据 |
| 2 | 印尼央行 | 印尼央行遭Conti勒索软件袭击，内部网络十余个系统感染勒索病毒 | 2022.01 | 13GB内部文件 |
| 3 | 三星电子 | 三星电子遭黑客组织攻击，导致大量机密数据外泄 | 2022.03 | 190GB机密数据 |
| 4 | 俄罗斯 | 黑客组织Anonymous入侵了俄罗斯文化部，并通过DDoSecrets平台泄露数据 | 2022.04 | 446GB数据 |
| 5 | 赛米控(Simikron) | 电子制造商赛米控近日披露遭到勒索软件攻击，部分公司网络被加密 | 2022.08 | 2TB |
| 6 | 思科 | 思科官方披露，内网遭到阎罗王勒索软件团伙入侵 | 2022.08 | 2.75GB机密数据 |
| 7 | 丰田汽车 | 丰田发现，T-Connect网站源代码的一部分被错误地发布在GitHub上，其中包含存储客户数据服务器的访问密钥 | 2022.10 | 30万客户数据 |


全国人民代表大会
The National People's Congress of the People's Republic of China

首页 | 宪法 | 人大机构 | 栗战书委员长 | 代表大会会议 | 常委会会议 | 委员长会议 | 权威发布 | 立法 | 监督 | 代表
对外交往 | 选举任免 | 法律研究 | 理论 | 机关工作 | 地方人大 | 图片 | 视频 | 直播 | 专题 | 资料库 | 国旗 | 国歌 | 国徽

当前位置： 首页

中华人民共和国个人信息保护法

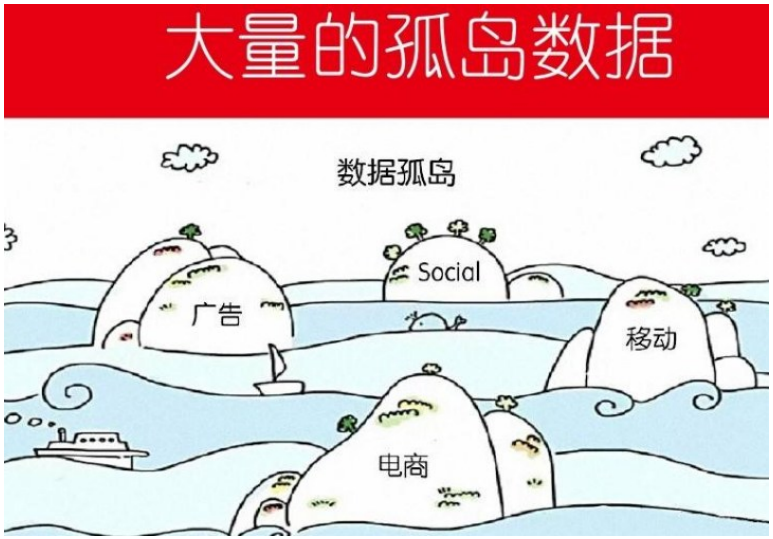
（2021年8月20日第十三届全国人民代表大会常务委员会第三十次会议通过）

来源： 中国人大网 浏览字号： 大 中 小 2021年08月20日 16:53:44

# 其它安全性保护



- **背景：**
  - **数据孤岛又越来越严重**
    - 每个事业部的数据就像一个个孤岛一样无法(或者极其困难)和企业其他数据进行连接互动




具体体现：
物理上：数据在不同部门独立存储，彼此独立

逻辑上：数据基于不同角度定义和理解，使得数据被赋予不同含义

# 其它安全性保护

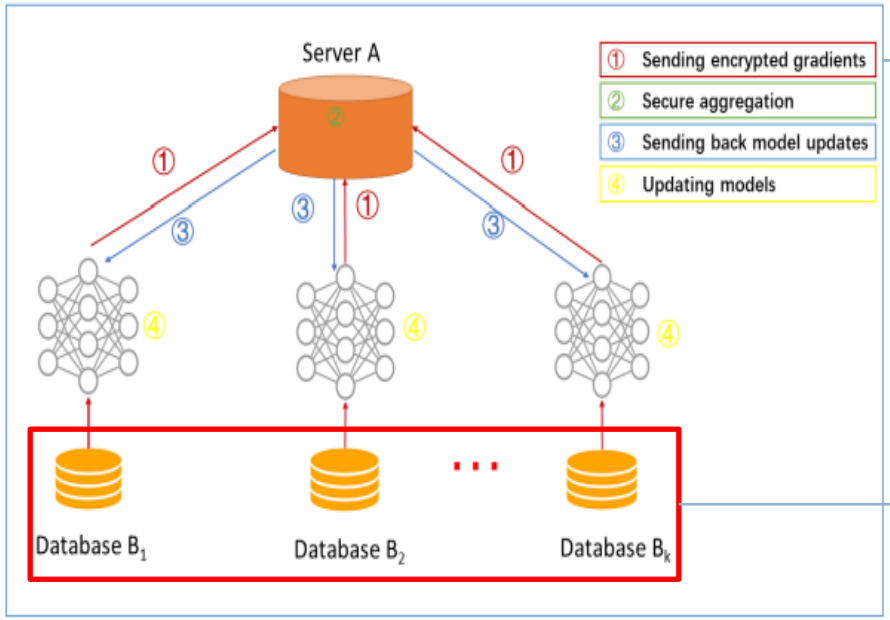


- 联邦学习
  - 提出：
    - 2016年为了解决手机终端本地更新模型问题谷歌提出联邦学习方法
  - 定义：2019年联邦学习技术与数据隐私保护大会
    - 联邦机器学习(Federated machine learning/Federated Learning)，又名联邦学习，联合学习，联盟学习。联邦机器学习是一个机器学习框架，能有效帮助多个机构在满足用户隐私保护、数据安全和政府法规的要求下，进行数据使用和机器学习建模

# 其它安全性保护



- 联邦学习与数据隐私保护
  - 实现方式：
    - 各方数据保留在客户本地数据库中，由服务器在保持数据保密性的前提下利用加权平均梯度或参数等方式汇总生成全局模型后再分发给各个客户端




加密上传模型参数/梯度

本地隐私数据

# 其他安全性保护



- 联邦学习和分布式机器学习

| | 联邦学习 | 分布式机器学习 |
|---|---|---|
| 控制权 | 计算节点数据不受中心节点控制 | 计算节点及其数据均受中心节点控制 |
| 数据分布 | Non-IID：非独立同分布 | IID |
| 数据量 | 计算节点数据量私有，不同节点间数据量可能不同 | 数据量统一分配，相对均衡 |
| 稳定性 | 计算节点可能随时退出 | 计算节点稳定 |

# 其它安全性保护



- **联邦学习的目标：**
  - 数据隔离：数据不会泄露
  - 无损：与将数据整合相比性能不明显下降
  - 对等：多个数据提供方地位平等
  - 共同获益：多个数据提供方共同享受收益
- **面临的挑战：**
  - 数据不平衡：数据量与特征不平衡
  - Non-IID：分布的数据子集非独立同分布
  - 大规模分布式数据：数据分布在大量客户端上难以聚合
  - 有限的沟通：各个分布的数据提供方提供行为自主、随机、不可控

# 其它安全性保护



- 研究层次：


联邦学习实际落地

联邦学习框架的优化与设计

如何更新模型，兼顾整体和个性化

降低频繁通信带来的资源开销

解决设备异质性问题


# 其它安全性保护

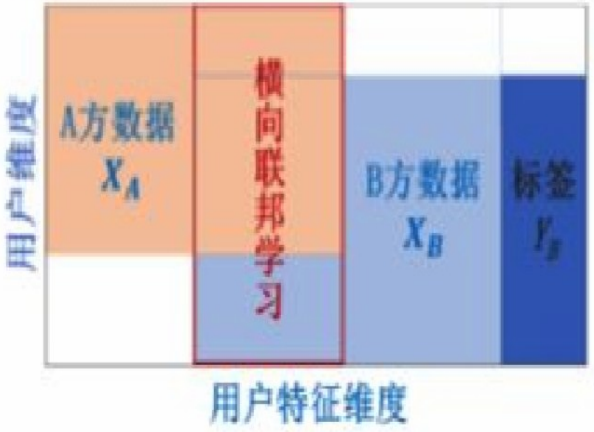


- 通信层：
  - 提高通信效率
    - 模型参数压缩
  - 保护隐私
    - 结合隐私加密技术
- 算法层：
  - 全局优化：
    - 注重从整体上拟合数据
  - 个性化优化：
    - 更好保留本地数据特征

# 其它安全性保护



- 系统层：依数据角度分类

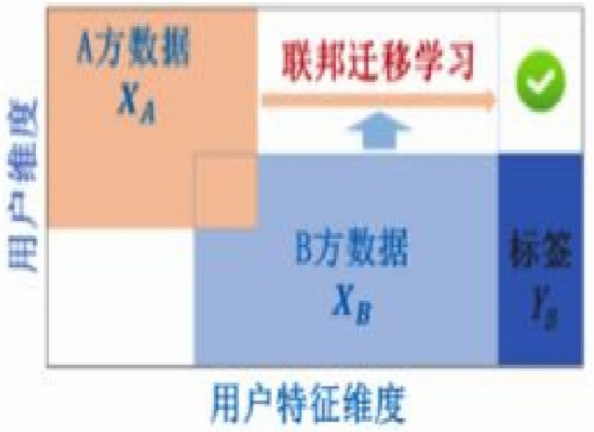



横向联邦学习

用户特征重叠较多，而用户重叠较少（最为普遍）


纵向联邦学习

用户重叠较多而用户特征重叠较少（需要进行加密样本对齐）




联邦迁移学习

数据集间用户与用户特征重叠部分都较小

# 其它安全性保护

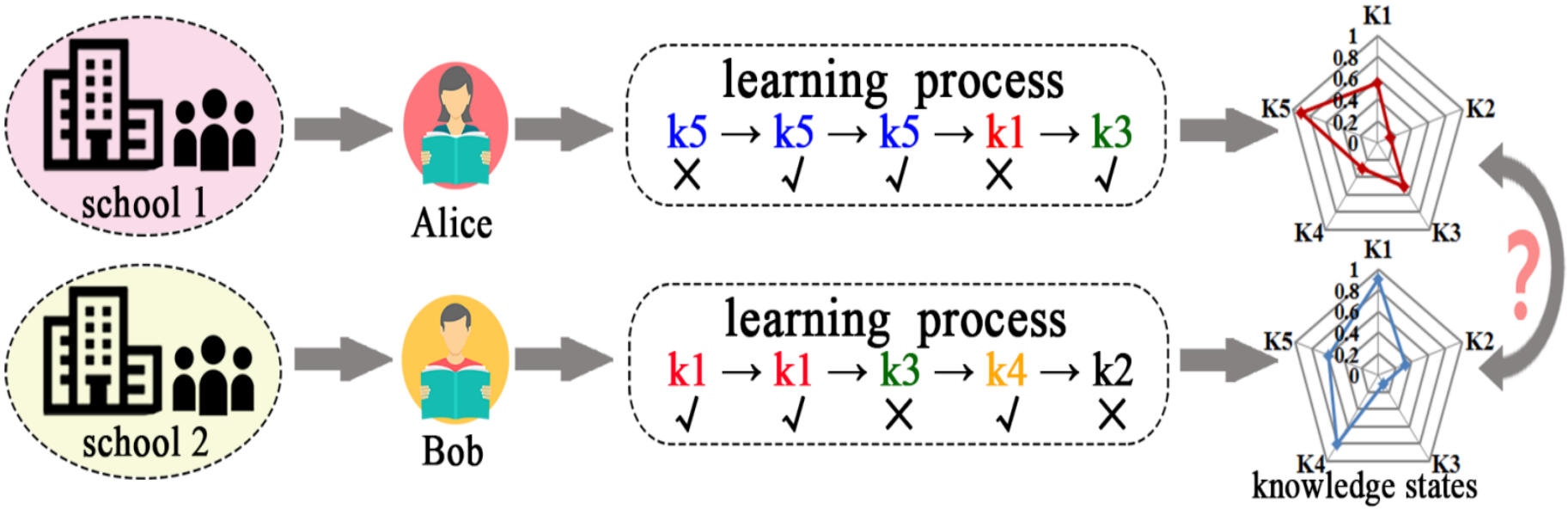


- 联邦学习应用：输入法

# 其它安全性保护



- **联邦学习应用：教育领域**
  - 保护不同学校学生的隐私
  - 精准评估不同学生的知识掌握水平

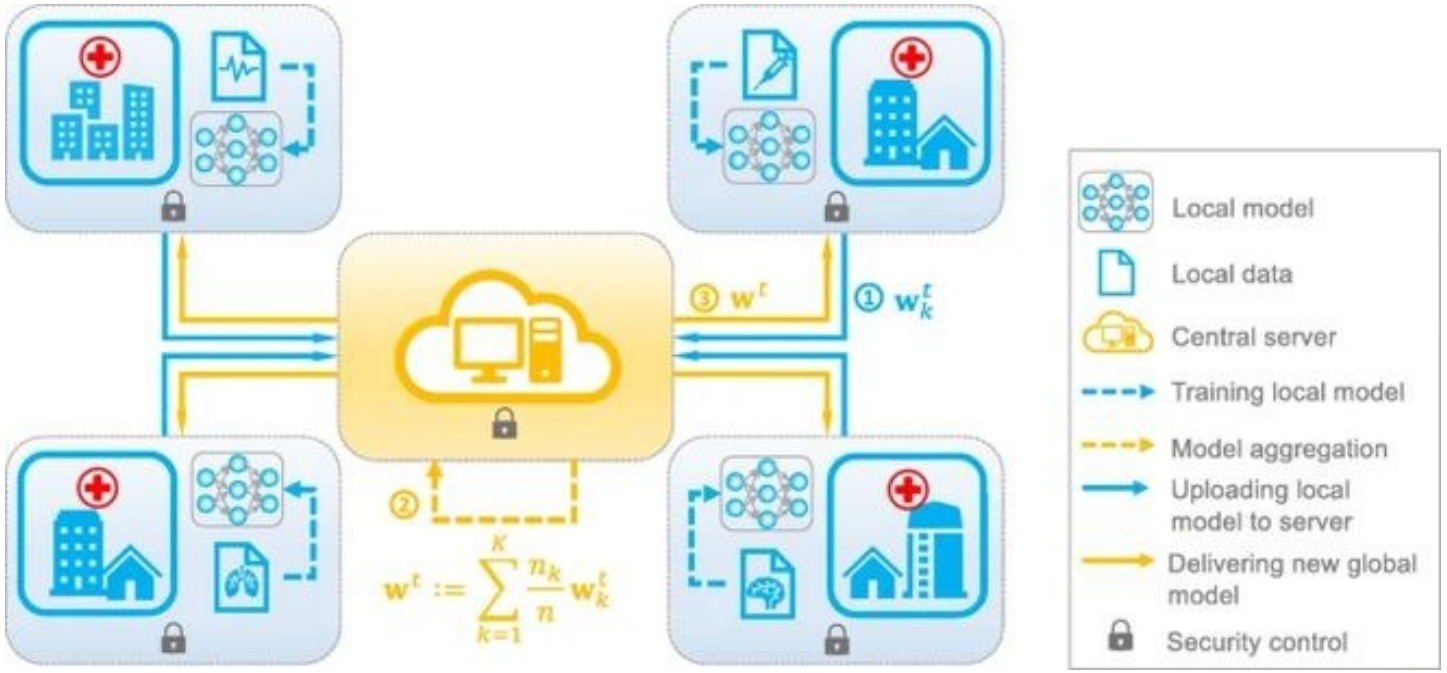



Jinze Wu, Zhenya Huang, Qi Liu, Defu Lian, Federated Deep Knowledge Tracing, WSDM'2021:

# 其它安全性保护



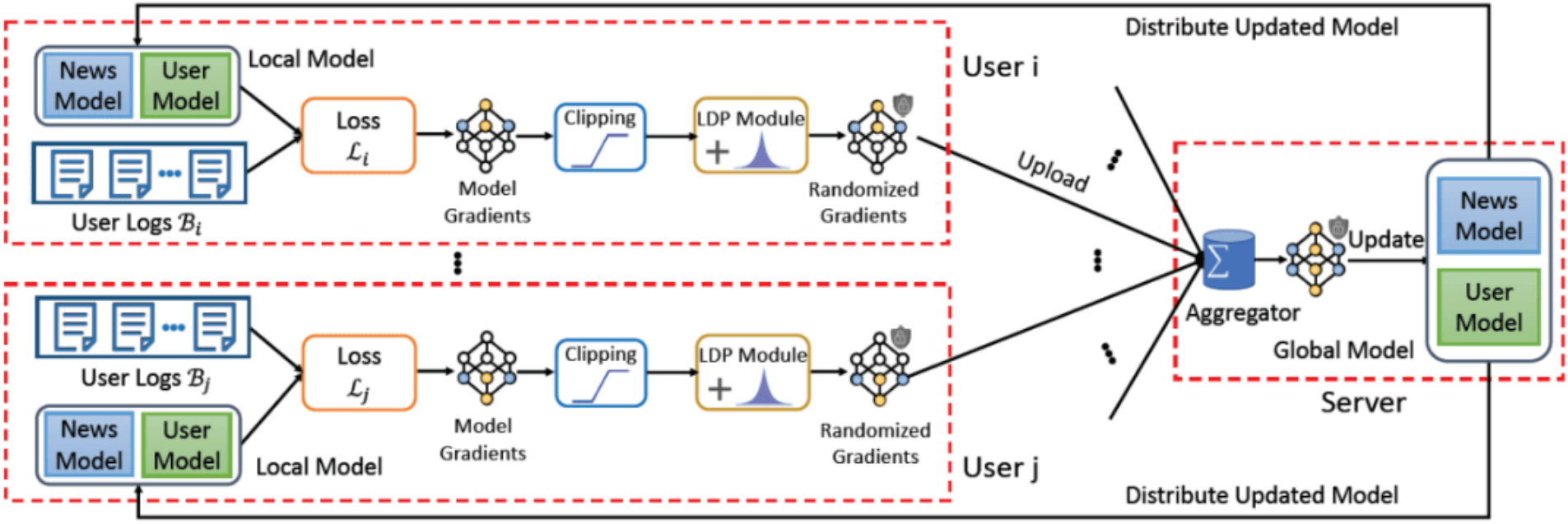
- **联邦学习应用：医疗领域**
  - 打破医疗数据（影像、ECG）等的数据孤岛



# 其它安全性保护



- **联邦学习应用：推荐系统**
  - 用户隐私保护
  - 提供适宜商品、服务



# 其它安全性保护



- **联邦学习应用：其他**
  - 自动驾驶
  - 信贷安全
  - “滴滴”
  - …


# 统计数据库安全性（续）



**规则1：任何查询至少要涉及N(N足够大)个以上的记录**

**规则2：任意两个查询的相交数据项不能超过M个**

**规则3：任一用户的查询次数不能超过1+(N-2)/M**

# 统计数据库安全性（续）



- 数据库安全机制的设计目标：

  试图破坏安全的人所花费的代价 >> 得到的利益


# 第四章 数据库安全性



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 其它安全性保护

4.7 小结

# 4.7 小结



- 数据的共享日益加强，数据的安全保密越来越重要
- DBMS是管理数据的核心，因而其自身必须具有一整套完整而有效的安全性机制
- 实现数据库系统安全性的技术和方法
  - 用户身份鉴别
  - 存取控制技术：自主存取控制和强制存取控制
  - 视图技术
  - 审计技术
  - 数据加密存储和加密传输

# 小结（续）



- 实现数据库系统安全性的技术和方法
  - 存取控制技术
  - 视图技术
  - 审计技术
- 自主存取控制功能
  - 通过SQL 的GRANT语句和REVOKE语句实现
- 角色
  - 使用角色来管理数据库权限可以简化授权过程
  - CREATE ROLE语句创建角色
  - GRANT 语句给角色授权