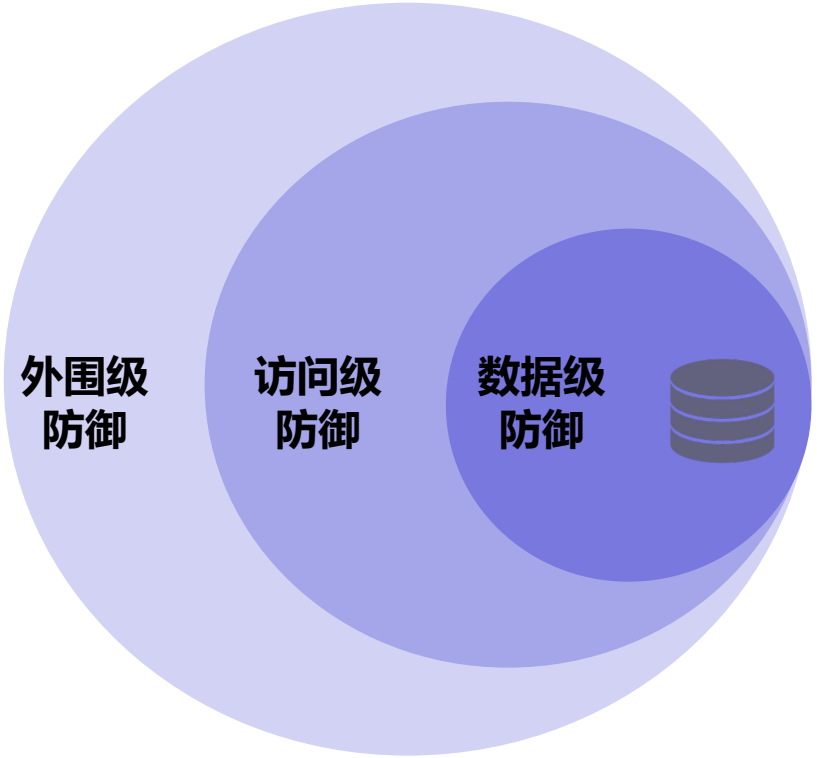
第13章 数据库安全

清华大学数据库组 李国良

liguoliang@tsinghua.edu.cn
http://dbgroup.cs.tsinghua.edu.cn/ligl

数据库安全威胁

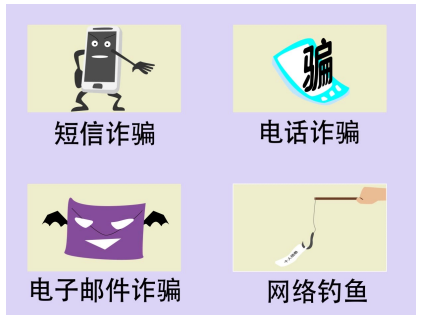
- **数据库面临的安全威胁**
  - 内部威胁：权限管理不当、弱密码、密码泄露、数据库软件漏洞
  - 外部威胁：SQL注入攻击、缓冲区溢出攻击、拒绝服务攻击
- **数据库安全**
  - 用于保护数据库管理系统免受恶意攻击和非法使用的各种措施
  - 保护数据和数据库系统本身
- **数据区安全层次**
  - 外围级防御
  - 访问级防御
  - 数据级防御



数据库安全威胁——内部威胁

- **使用不当**
  - 用户和管理人员忽略安全规则，不安全地配置和使用
  - 例如：弱密码，多人共用同一密码等
- **权限管理不当**
  - 为用户提供超出实际需要的权限或者用户滥用其权限
  - 缺乏对权限时效性的评估
- **数据库软件漏洞**
  - 用户很少及时安装补丁
- **社会工程学攻击**
  - 利用人类心理而非黑客技术来访问系统或数据的方法







数据库安全威胁——外部威胁

注入攻击

*SELECT * FROM user WHERE name = '' AND pwd = ''*

SQL注入

*SELECT * FROM user WHERE name = '' AND pwd = '1' or 1 != ''*

缓冲区溢出攻击

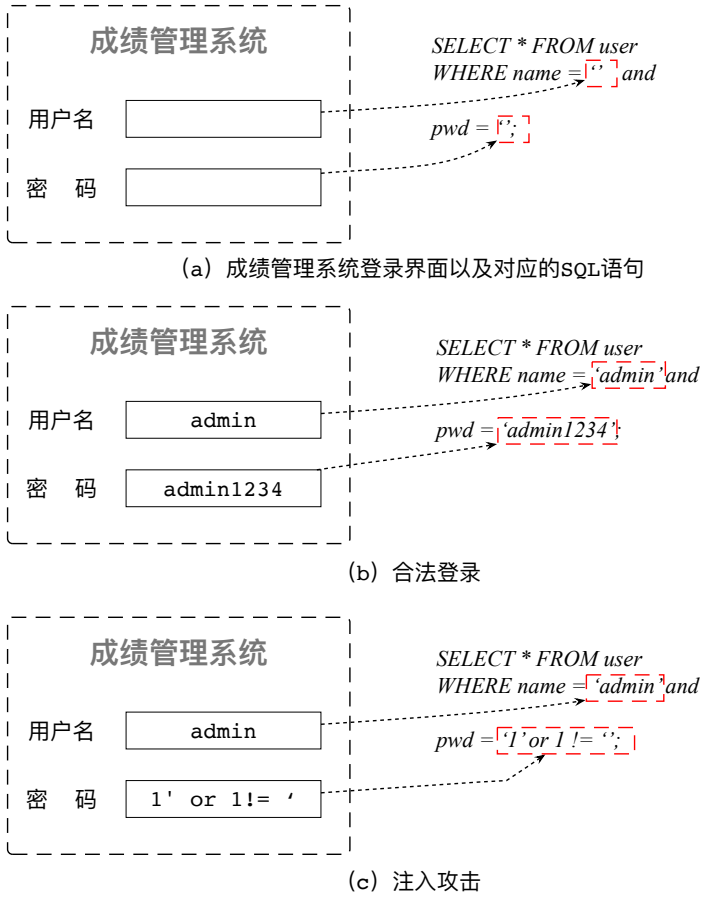
- 将大量数据写入固定大小内存块，发生缓冲区溢出
- 利用缓冲区溢出漏洞来破坏或控制数据库系统

拒绝服务攻击

- 用大量虚假请求压倒数据库服务器，导致数据库崩溃

恶意软件

- 为利用数据库系统漏洞或对损害数据库而编写的软件



数据库安全威胁——外部威胁

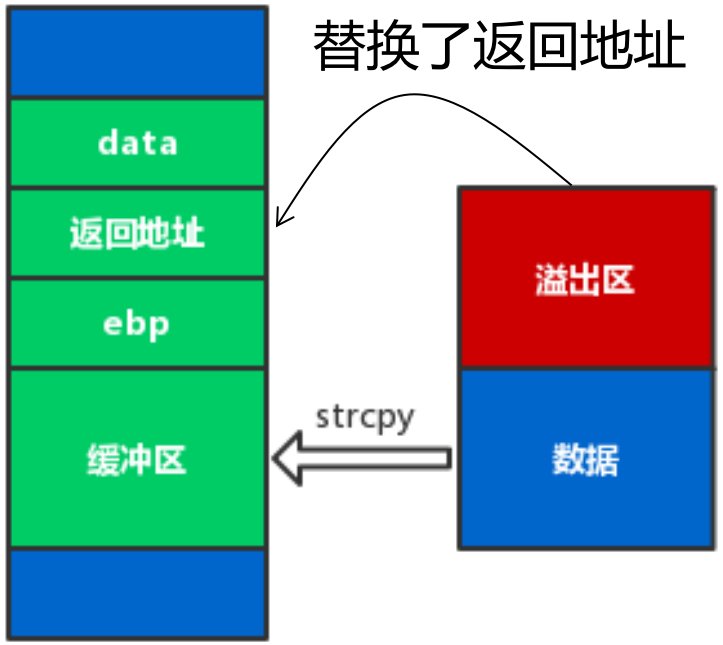
- **注入攻击**

*SELECT * FROM user WHERE name = '' AND pwd = ''*

SQL注入

*SELECT * FROM user WHERE name = '' AND pwd = '1' or 1 != ''*

- **缓冲区溢出攻击**
  - 将大量数据写入固定大小内存块，发生缓冲区溢出
  - 利用缓冲区溢出漏洞来破坏或控制数据库系统
- **拒绝服务攻击**
  - 用大量虚假请求压倒数据库服务器，导致数据库崩溃
- **恶意软件**
  - 为利用数据库系统漏洞或对损害数据库而编写的软件



数据库安全威胁——外部威胁

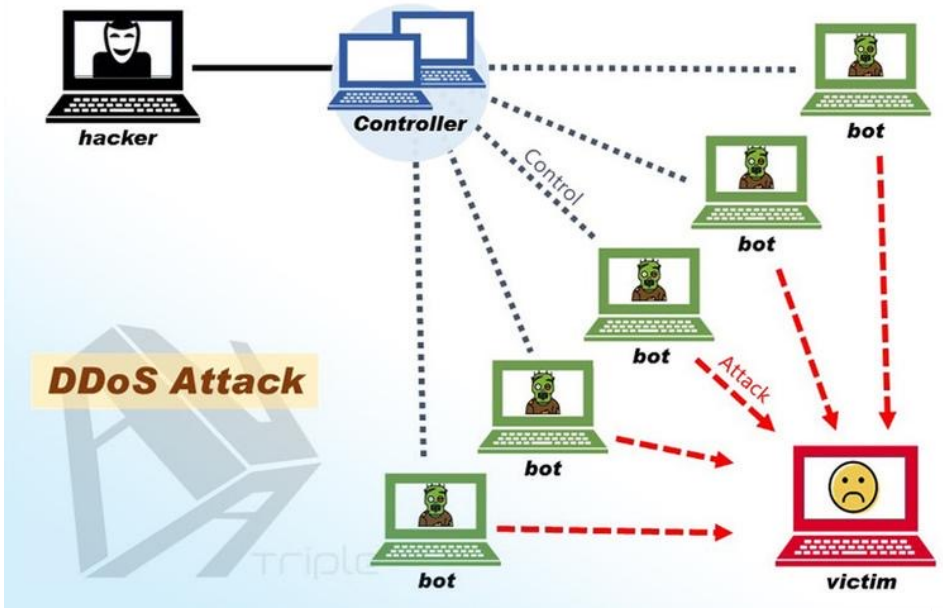
- **注入攻击**

*SELECT * FROM user WHERE name = '' AND pwd = ''*

SQL注入

*SELECT * FROM user WHERE name = '' AND pwd = '1' or 1 != ''*

- **缓冲区溢出攻击**
  - 将大量数据写入固定大小内存块，发生缓冲区溢出
  - 利用缓冲区溢出漏洞来破坏或控制数据库系统
- **拒绝服务攻击**
  - 用大量虚假请求压倒数据库服务器，导致数据库崩溃
- **恶意软件**
  - 为利用数据库系统漏洞或对损害数据库而编写的软件

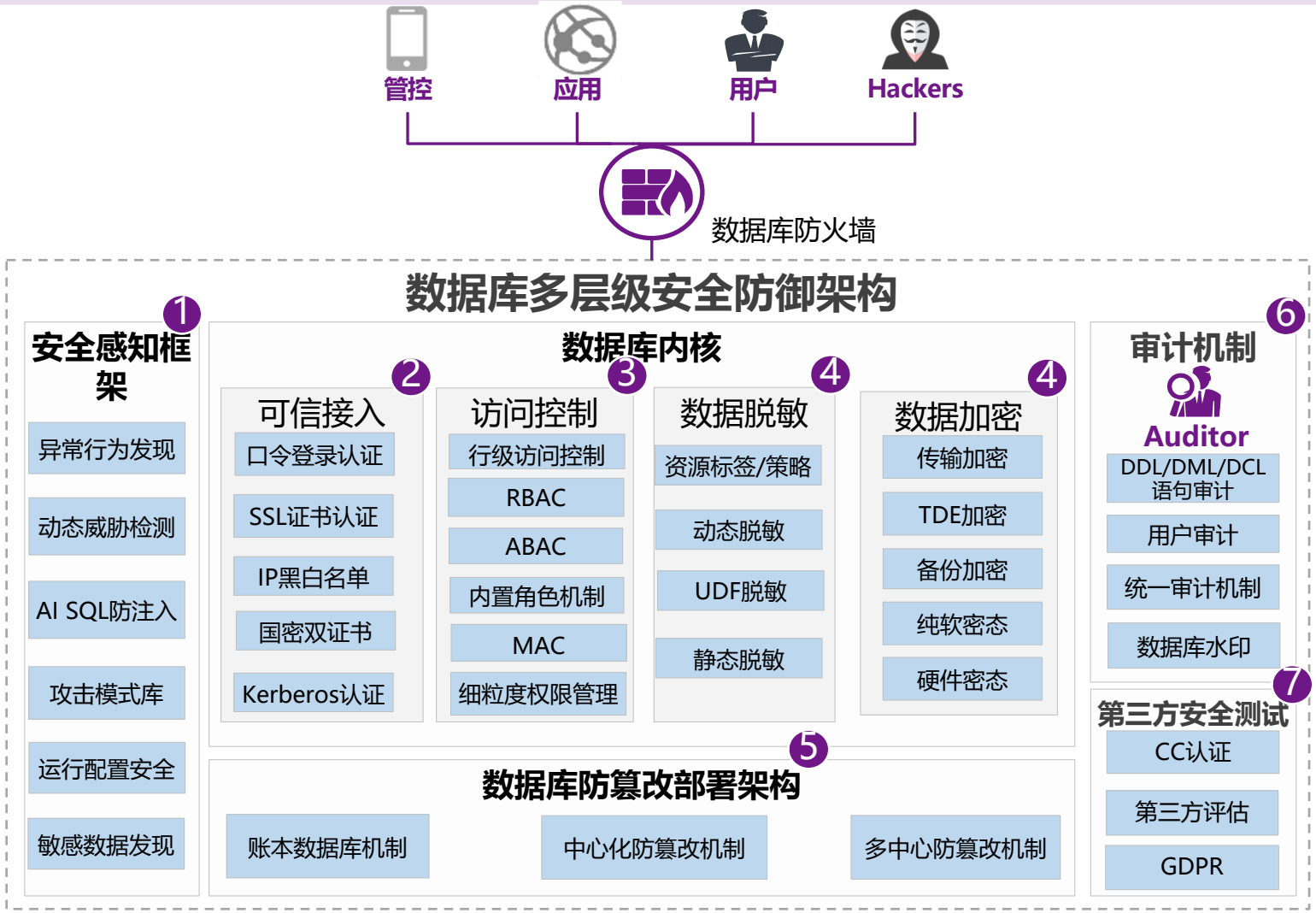


数据库安全需求

- **数据库系统安全**
  - 保障系统能够安全稳定运行，为用户正常提供服务
- **数据安全**
  - 即保护数据在数据库内能够安全存储、处理和传输

安全威胁	应对措施
使用不当	安全认证，数据库审计
权限管理不当	访问控制，安全配置策略，数据库审计
数据库软件漏洞	防火墙，入侵检测
社会工程学攻击	安全配置策略，数据加密，数据库审计
注入攻击	入侵检测，数据加密，隐私保护
缓冲区溢出攻击	访问控制，数据加密，隐私保护
拒绝服务攻击	防火墙，入侵检测，安全配置策略
恶意软件	防火墙，入侵检测，数据加密

数据库安全全貌



❶ 攻不破

❷ 进不来

❸ 拿不走

❹ 看不懂

❺ 改不了

❻ 赖不掉

❼ 信得过

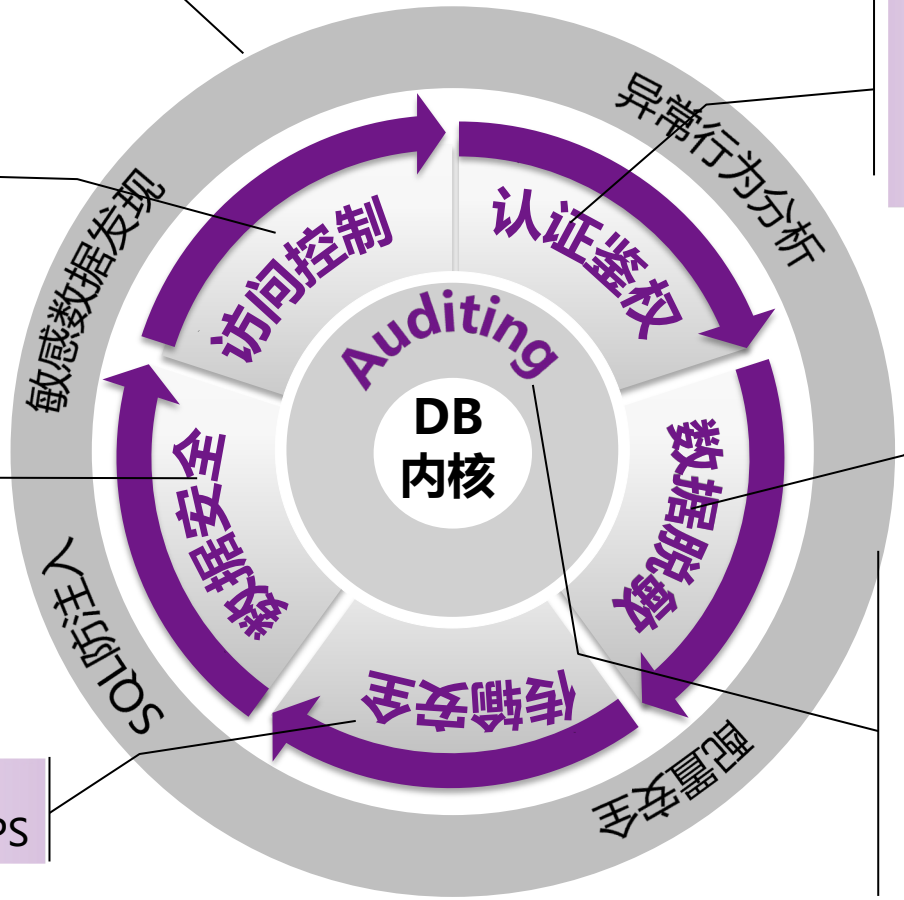
数据库安全层级

- 异常访问模式库/检测
- SQL注入检测
- 敏感数据发现

- 基于角色的权限管理模型
- 三权分立模型
- 角色访问控制RBAC
- 属性访问控制ABAC
- 细粒度对象权限管理
- 系统权限分离机制

- 密钥管理
- 透明加密(TDE)
- 全密态
- 防篡改
- 多方安全计算

- TLS
- HTTPS



- 用户与身份管理
- 口令登录认证机制
- 口令安全策略
- Kerberos认证

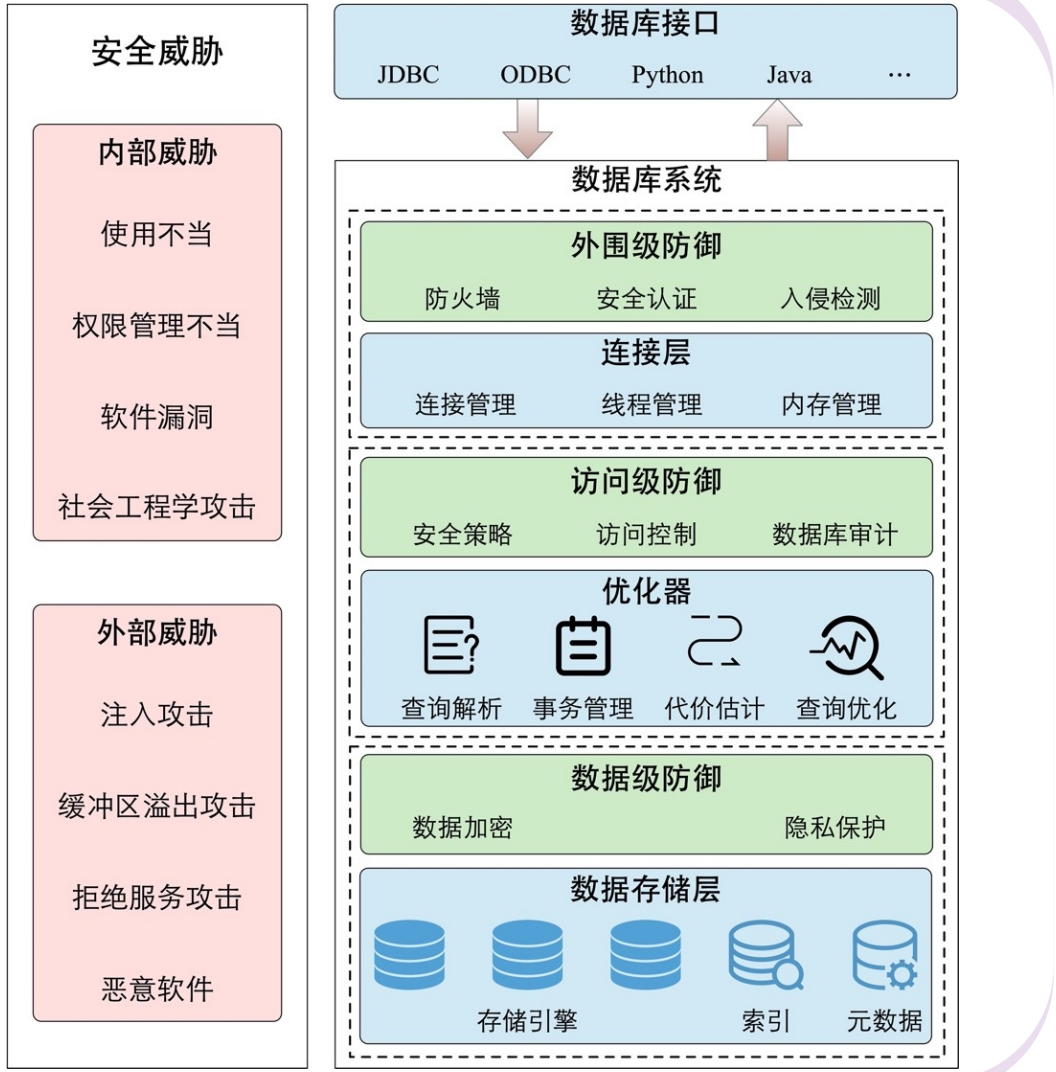
- 动态脱敏
- 静态脱敏
- 函数脱敏

- 传统审计机制，支持DDL&DML
- 统一审计
- 审计日志查询
- 审计日志管理
- 细粒度审计机制，自定义策略
- 账本防篡改

登录管理 → **访问控制** → **数据保护** → **审计追踪** → **数据展示**

数据库安全提纲

- **外围级防御**
  - 决定哪些用户可以访问数据库
  - 安全认证、防火墙、入侵检测
- **访问级防御**
  - 控制允许哪些用户有权访问某些数据或包含它的系统
  - 安全配置策略、访问控制、审计
- **数据级防御**
  - 预防和应对发生在数据库本身的安全隐患
  - 加密、隐私保护、脱密、防篡改、多方安全



数据库安全认证

- **安全认证**
  - 确认试图登录数据库的用户是否被授权访问数据库的过程
- **认证方式**
  - 数据库认证
  - 外部认证：
    - 由操作系统或网络服务执行身份验证
    - 通过安全套接字对用户进行身份验证
    - 使用企业目录验证和授权
    - 通过中间层服务器来验证用户的身份



数据库认证

数据库认证

- 使用存储在数据库中的信息对连接到数据库的用户进行验证
- 最常用的方法：密码认证
- 缺点：
  - 密码容易被盗、伪造和滥用
  - 难以应对复杂的网络攻击
  - 频繁的认证操作影响数据库性能

加强密码认证的安全措施

- 建立密码复杂性标准
- 密码不包含敏感词，如用户名等
- 设定密码时效性，定期修改密码

数据库安全认证方式

- 用户访问数据库时需要通过安全认证，数据库在认证时除了校验用户名和密码，还会校验客户端主机的IP地址，如果开启了SSL加密还会校验SSL证书信息。
- 用户在访问数据库前，需要在服务端的配置文件pg_hba.conf中配置客户端信息和认证方式。
- 配置文件pg_hba.conf的一行写一条信息表示一个认证规则，越靠前优先级越高，格式如下：

格式：TYPE DATABASE USER ADDRESS METHOD

例如：*host database thu 1 92.168.0.1/32 sha256*

TYPE：连接数据库的类型，支持local、host、hostssl、hostnossl等类型。

DATABASE：声明记录所匹配的数据库名称，多个数据库可用逗号分隔，all表示匹配所有数据库。

USER：声明记录所匹配的数据库用户，多个用户名可用逗号分隔，all表示匹配所有用户。

ADDRESS：声明记录所匹配的IP地址，支持IPv4和IPv6。

METHOD：声明连接时候使用的认证方法，支持sha256(推荐)、cert(证书认证)、reject(黑名单)等。

数据库认证方式

- **检查客户端的IP地址是否合法**
  - 校验客户端IP是否在服务端配置文件中配置，并且认证方法为非reject。
- **检查被认证用户是否存在或被锁定**
  - 如果用户不存在或被锁定(涵盖自动被锁定或人工锁定)则认证失败。
- **校验口令是否正确**
  - 使用基于RFC5802协议的口令认证方案。简单来讲，客户端通过HMAC算法对口令加密后将密文发送到服务端，服务端将口令密文再经过HMAC计算后与系统表中存储的用户口令密文进行比较，如果相同则认证通过。

数据库认证方式

- **检查database的正确性**
  - 校验database是否在配置文件中体现，如果不存在，则登录认证失败。
- **检查用户有效期**
  - 通过CREATE ROLE或ALTER ROLE语法中的VALID BEGIN和VALID UNTIL参数可以设置用户生效和失效的时间，如果不在有效期内则认证失败。如果不设置则默认无有效期限制。
- **检查连接数是否已达到上限**
  - 最大连接数可以按照用户、数据库或者实例等不同级别进行设置，如果连接数达到上限后无法创建新连接。

数据库外部认证

- **外部认证**
  - 强身份认证：进行双因素(如密码+短信)或多因素身份验证
  - 代理认证：让中央设施对网络的所有成员进行身份验证
- **强身份认证优点**
  - 可根据需要选择不同的身份验证机制
  - 减少数据库管理开销，提高数据库性能
- **代理认证优点**
  - 有效解决网络上节点伪造身份
  - 减少每个数据库认证开销

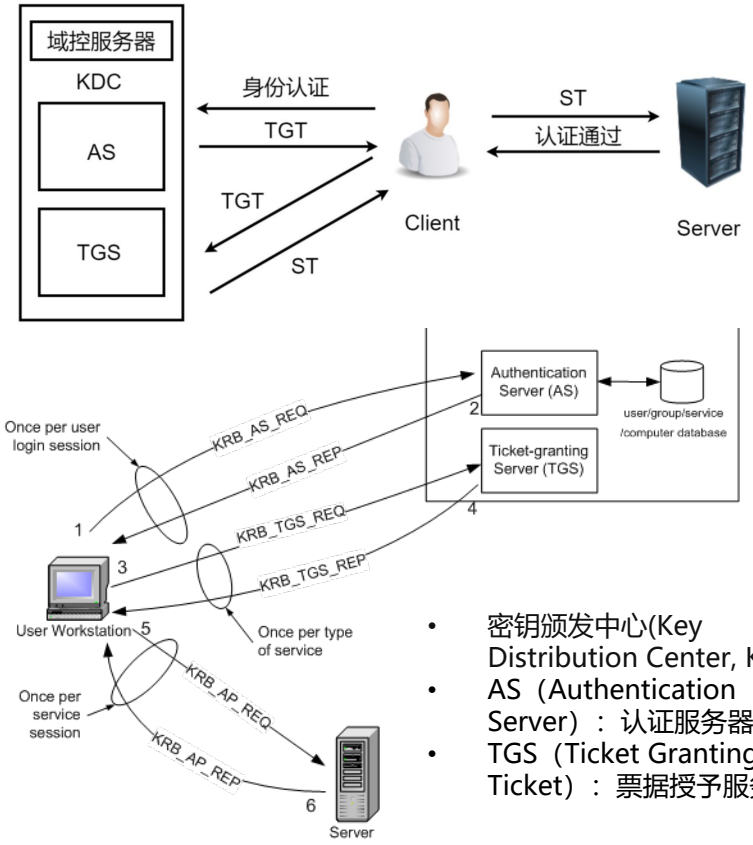


密码+动态口令

数据库强身份认证

- **Kerberos协议：网络身份验证协议**
  - 解决集群内节点或者进程之间的认证问题，恶意用户无法仿冒集群内节点或进程来登录数据库系统，只有内部组件才可以持有用于认证的凭证，消减了仿冒风险，提升了数据库系统的安全性
  - 使用密钥加密技术为客户端/服务器应用程序提供强身份验证
  - 集中进行密码存储，对数据库的连接进行身份验证
  - 防止窃听、防止重放攻击、保护数据完整性等场合
  - 控制网络访问时间，通过加密和完整性校验来防止未授权访问
- **令牌卡**
  - 动态显示与身份验证服务同步的一次性密码
  - 优点：安全性高，使用方便，提高了时候溯源能力
- **智能卡**
  - 内置内存和处理器，需要特定的读卡器读取
  - 优点：安全高效，可存储数字证书等机密信息





- 密钥颁发中心(Key Distribution Center, KDC)
- AS（Authentication Server）：认证服务器
- TGS（Ticket Granting Ticket）：票据授予服务器

数据库强身份认证

公钥基础设施

- 一组行业标准的程序和策略
- 证书颁发机构 （Certification Authority，CA）是受信任的第三方，证书的申请、颁发、作废、查询等，都是由证书机构完成的
- 使用数字证书为用户提供加密方法、密钥管理、身份验证服务
- 通过证书机构完成证书的申请、颁发、作废、查询等

生物识别

- 利用人体固有生理特性，如指纹或声音等物理特征
- 优点：安全性更高，更加方便，不易遗忘或丢失，不易被伪造



指纹



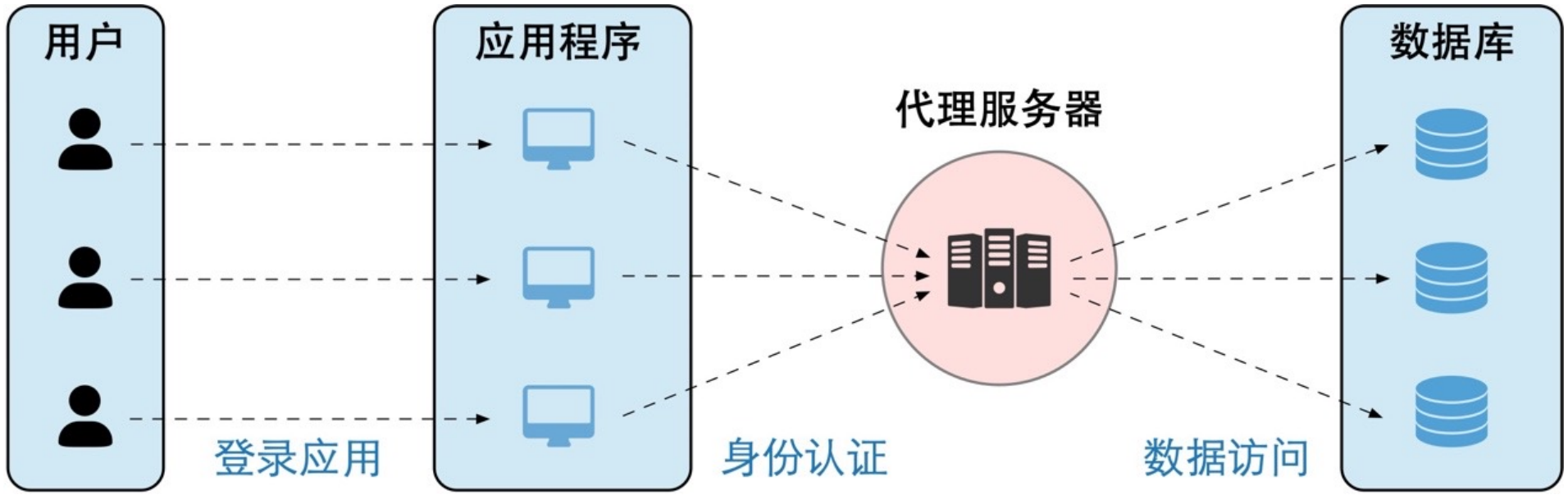
虹膜识别



声纹

数据库代理认证

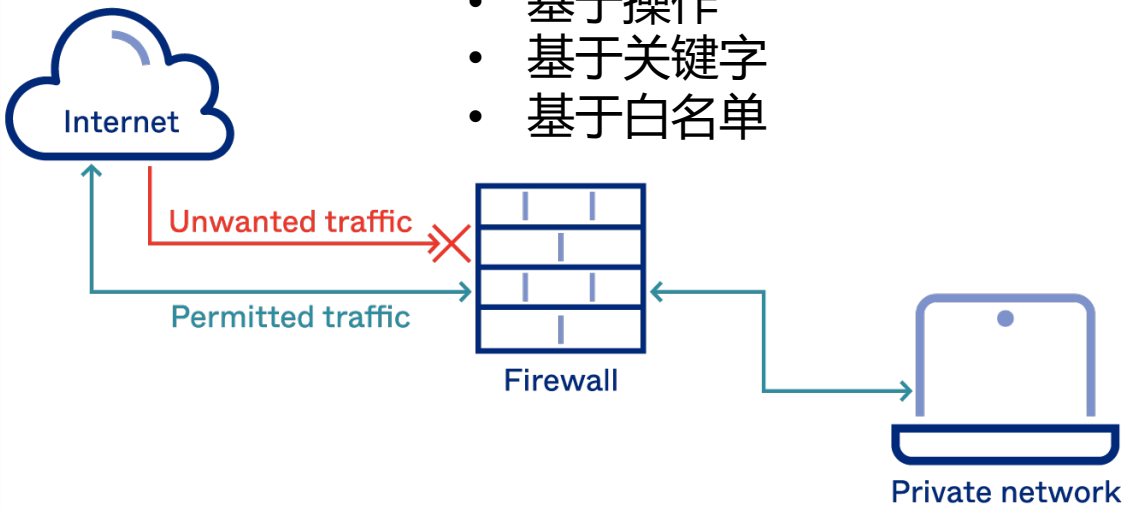
- **优势：**
  - 与数据库隔离开，数据库管理员可以动态调整用户和应用程序的访问权限
  - 集中管理和审计，降低数据库暴露风险；提供了更大的灵活性和扩展性，提高数据库溯源能力
- **原理**
  - 应用程序将用户凭据传递给数据库，安全代理负责安全认证



数据库防火墙

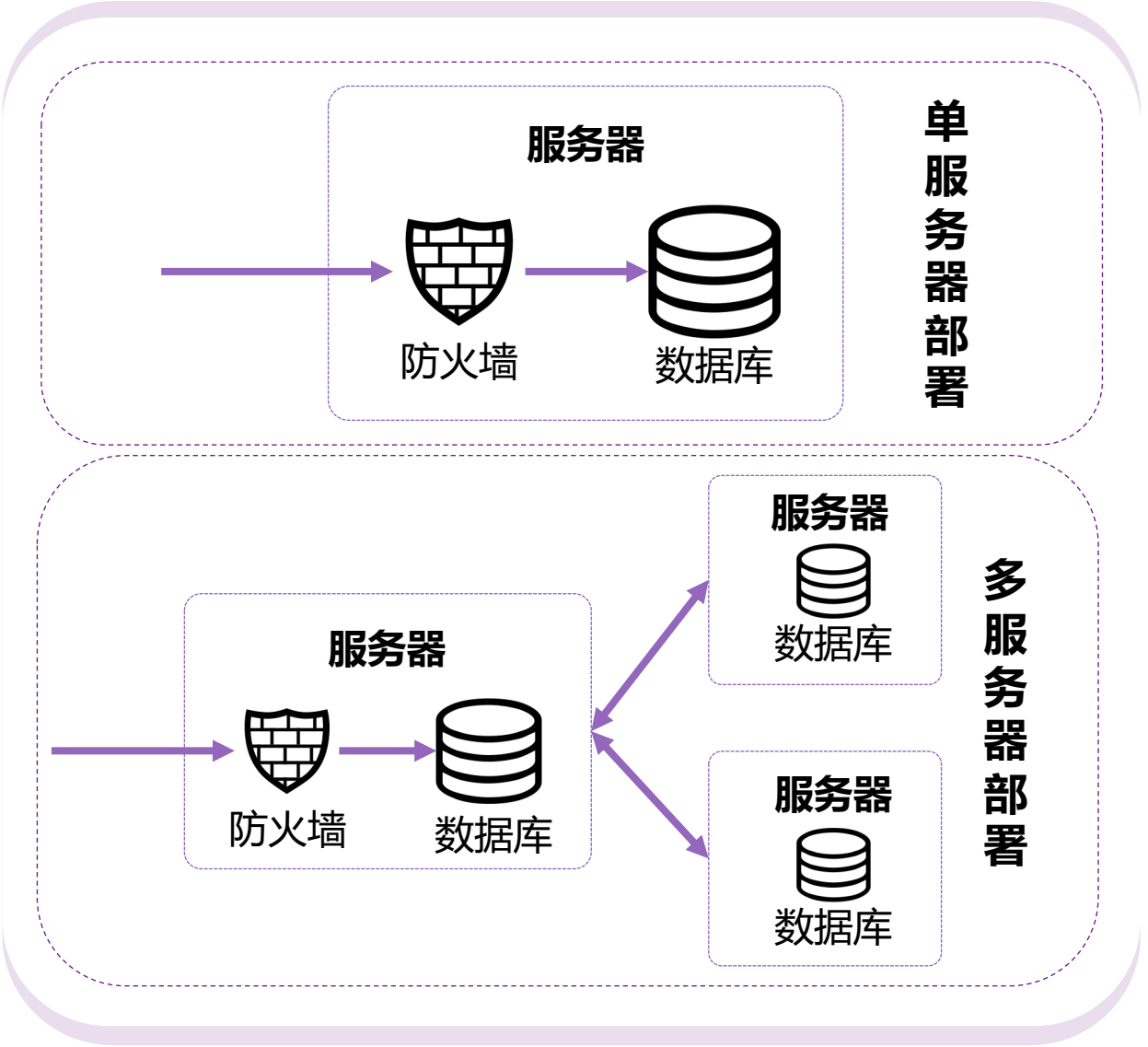
- **一组预定义的、可定制的安全审计策略，它们可以根据过去事件以及威胁命名空间识别数据库攻击**
- **通用防火墙的不足**
  - 缺少对数据库流量进行控制的规则和策略集
- **数据库防火墙的作用**
  - 识别和防止数据库特定攻击
  - 监视和审计对数据库的所有访问
  - 生成特定的合规性报告

- 基于IP
- 基于用户
- 基于操作
- 基于关键字
- 基于白名单



数据库防火墙

- **部署方式**
  - 单数据库服务器：部署在数据库服务器上
  - 多数据库服务器：部署在靠近网关的服务器上



数据库防火墙

- **基本原理**
  - “签名”：事件或威胁模式的识别特征
  - 预定义的、可定制的安全策略
  - 利用安全策略将SQL语句与签名比对，识别攻击行为
- **白名单与黑名单**
  - 白名单：SQL语句安全性已知，无威胁
  - 黑名单：SQL语句有特定威胁
  - 白名单的SQL语句会被执行，黑名单的SQL语句不会执行
- **应用**
  - 专注于数据库保护
  - 阻止注入攻击和缓冲区溢出攻击
  - 实时监控和报告

数据库入侵检测

- **对网络传输流量进行即时监视，发现威胁时采取反应措施**
- **通用入侵检测系统的不足**
  - 无法有效抵御针对数据库的特定攻击
  - 无法组织内部用户异常访问
- **数据库入侵检测组成部分**
  - 异常检测系统：发现异常和非法的数据库查询请求
  - 异常响应系统：对发现的异常查询请求进行处理

数据库入侵检测

异常检测

- 利用数据库日志文件，构建用户行为特征
- 将用户发出的SQL语句与用户行为特征对比
- 新的SQL与用户行为特征不一致，则判定为异常

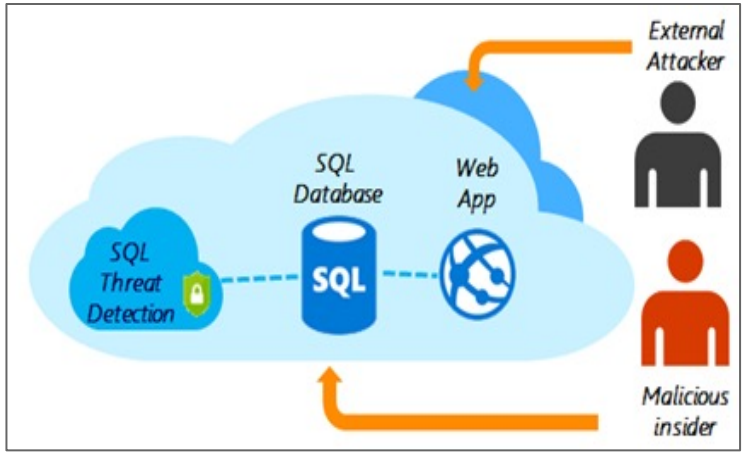
异常响应

- 粗粒度响应机制
  - 严格的响应机制：立即丢弃恶意请求
  - 宽松的响应机制：进发出警报，并允许请求通过
- 细粒度响应机制
  - 策略根据异常请求的详细信息指定适当的响应操作
  - 通过在访问控制系统中设置特权状态来进行更细粒度的响应动作

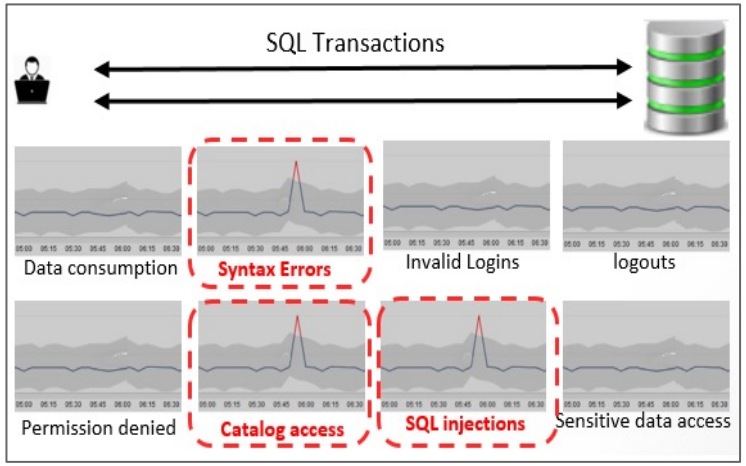
数据库异常检测

- SQL注入检测：检测模块内置于数据库优化器模块中，精准、高效识别优化过程中查询条件失效的时刻，拦截注入命令
- 异常行为检测：利用数据库日志数据和状态数据基于库内AI引擎不断学习，实时检测恶意用户的异常操作

SQL注入检测

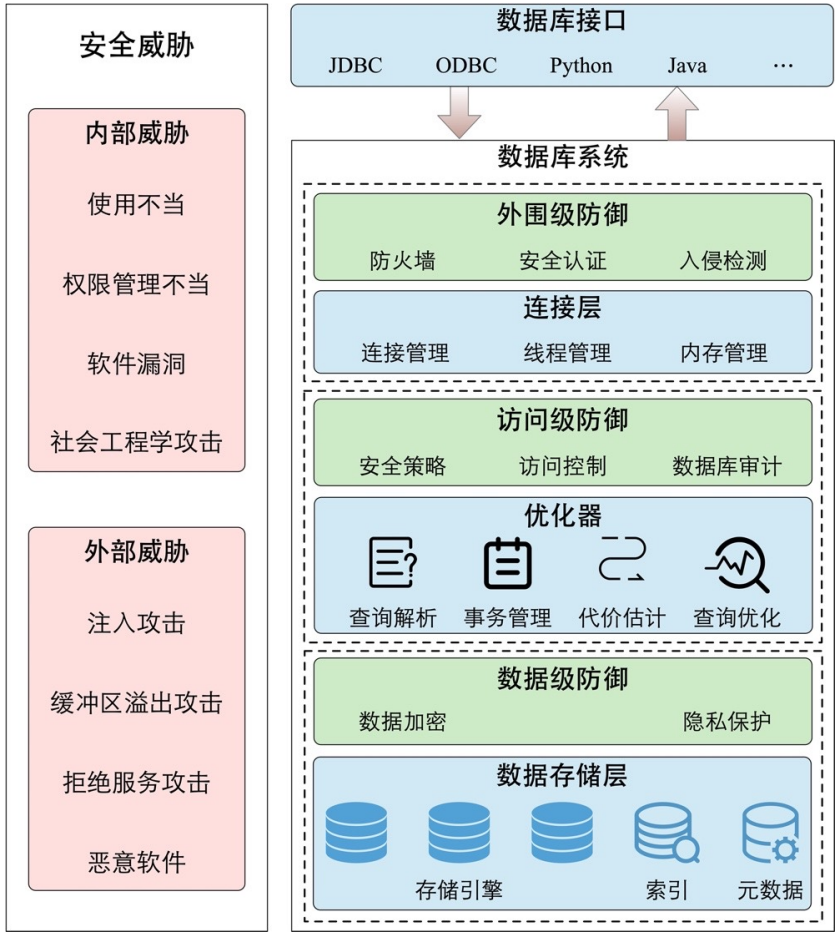


异常行为检测



数据库安全目录

- **外围级防御**
  - 决定哪些用户可以访问数据库
  - 安全认证、防火墙、入侵检测
- **访问级防御**
  - 控制允许哪些用户有权访问某些数据或包含它的系统
  - 安全配置策略、访问控制、审计
- **数据级防御**
  - 预防和应对发生在数据库本身的安全隐患
  - 加密、隐私保护、脱密、防篡改、多方安全



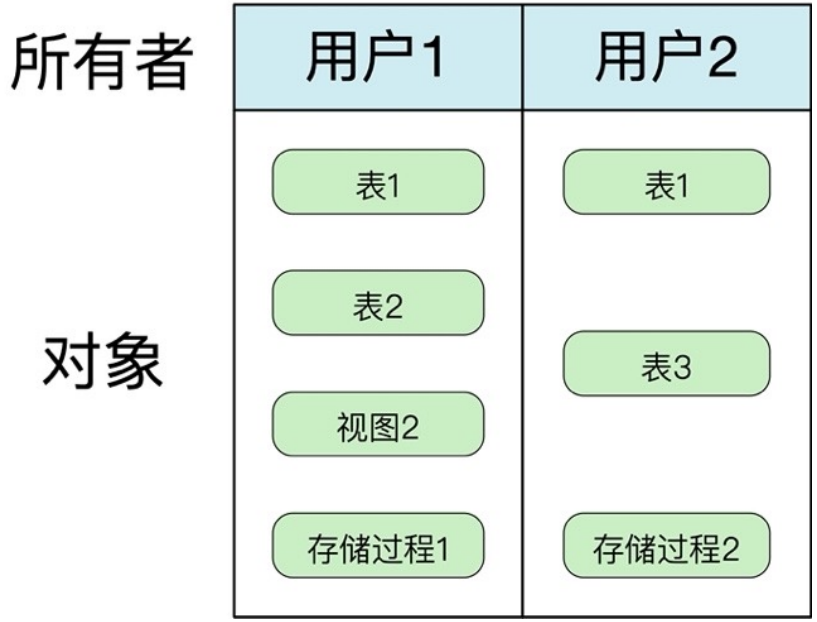
数据库访问控制

访问控制

- 按照用户的身份和权限，控制用户对数据库中数据访问
- 组成：安全认证，权限控制
- 安全认证不能避免数据未经授权的访问

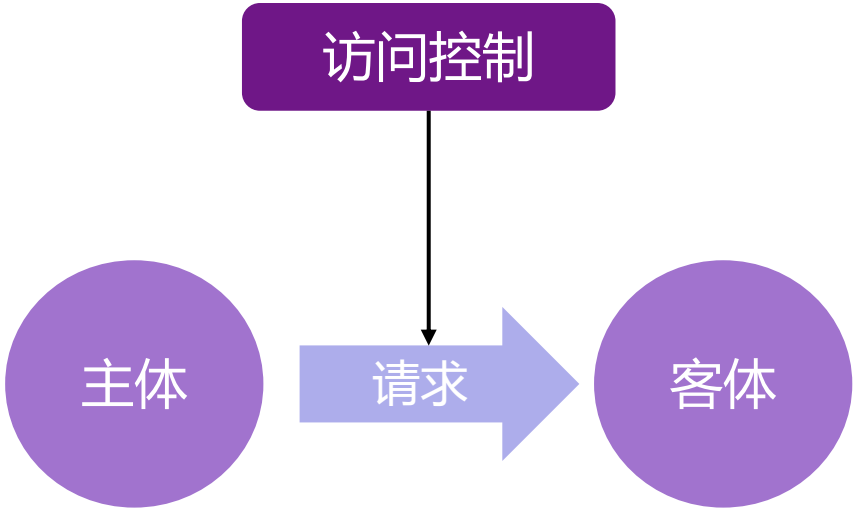
数据库命名空间

- 表示属于特定用户的对象组
- 对象包括表、视图、存储过程等
- 数据库可以有多个Schema
- Schema对象在逻辑上存储在表空间中
- Schema是数据库的概要，没有任何数据



访问控制的基本原理

- **权限**
  - 允许或拒绝数据库用户提出的数据访问请求
  - 主体：代表用户发出SQL命令的应用程序
  - 客体：数据库中的数据或资源
- **访问控制关注的问题**
  - 阻止访问：无权限时，主体不能访问客体
  - 确定访问权限：确定主体是否有权对客体进行访问
  - 授予访问权限：授予主体访问客体的权限
  - 撤销访问权限：删除主体对客体的访问权限
  - 审计访问：确定哪些主体可访问一个客体，或一个主体可访问哪些客体



自主访问控制DAC

- 自主访问控制（Discretionary Access Control，DAC）
  - 数据所有者决定哪些用户可以访问该数据
  - 主体可以决定谁可以访问其拥有的客体
  - 客体由称为参考监视器的保护装置保护
  - 参考监视器检查每个对数据访问的SQL命令
- 基本原理
  - 主体S，客体O，访问权限R，关系D
  - 核心：主体S是否对客体O拥有正确的访问权限R
  - 如果主体S对客体O有权限R，(S,O,R)在D中；否则，如果S没有对客体O的权限R， (S,O,R)不在D中

	表1	表2
小李	RW	RW
小王	-	R

- 主体小李可以对客体表1和客体表2进行“读（R）”和“写（W）”操作
- 主体小王只能对客体表2进行“读（R）”操作。

自主访问控制DAC

- **访问控制矩阵（数学关系D）**
  - 每一行对应一个主体
  - 每一列对应一个客体
  - 每个单元格都包含一组权限
- **访问控制列表：按列存储的实现方式**
  - 每个客体都对应一个条目列表$\{(S_1, R_{S_1}), \cdots, (S_n, R_{S_n})\}$
  - 每个条目主体S和主体S_i对客体的权限R_{S_i}
- **权限列表：按行存储的实现**
  - 每个主体都维护一个权限列表$\{(O_1, R_{O_1}), \cdots, (O_n, R_{O_n})\}$
  - 包含客体O_i和该客体的权限$R(O_i)$
- **例如windows权限控制**

强制访问控制MAC

- **强制访问控制（MAC，Mandatory Access Control）**
  - 根据中央机构的规定分配访问权限
  - 数据属于数据库，权限由数据库分配
- **标签：由一个敏感度级别和一组关键词集合**
  - 敏感度级别：公开，秘密，机密，绝密
  - 关键词：主体或客体所属的范畴
  - 每个主体和每个客体都有对应的标签
  - 例如：一张数据表Person用来记录人员信息
    - 表Person的标签设为（机密，{人事，财务}）
    - 人事部门的主管小王的标签设为（绝密，{人事，财务}）
- **例如共享文件**

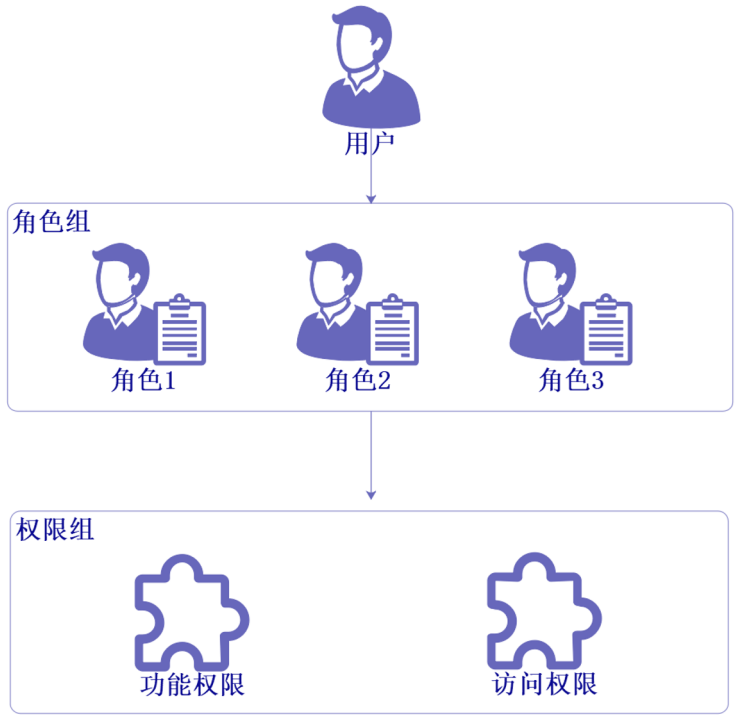
强制访问控制

- **BLP（Bell and LaPadula）策略**
  - 信息不能泄露给未获得信息许可的主体
  - 给定两个标签$L_1 = (S_1, C_1)$和$L_2 = (S_2, C_2)$
    - S_1和S_2是敏感度级别，C_1和C_2是关键词集合
  - 当$S_1 \leq S_2$且$C_1 \subseteq C_2$时，有$L_1 \leq L_2$，即L_2比L_1更具限制性
- **BLP安全条件**
  - 只有当$L(O) \leq L(S)$时，主体S才能读取客体O，即主体不允许“向上读” → 防止越权访问
  - 只有当$L(S) \leq L(O)$时，主体S才可以写客体O，即主体不允许“向下写” → 防止信息泄露

- 人事表Person（机密，{人事，财务}）。
- 人事部门主管小王（绝密，{人事，财务}）。
- 小王的标签（绝密，{人事，财务}）中敏感度级别比数据表Person的敏感度级别高（机密，{人事，财务}），那么小王就可以读数据表Person中的数据。
- 客体资源配置表
  - 资源: 人事表Person
    - 主体: 财务人员 ， 人事人员
    - 等级：机密级
    - 操作：查看主体配置表
- 主体: 小王 类别: 人事，财务 等级：绝密级

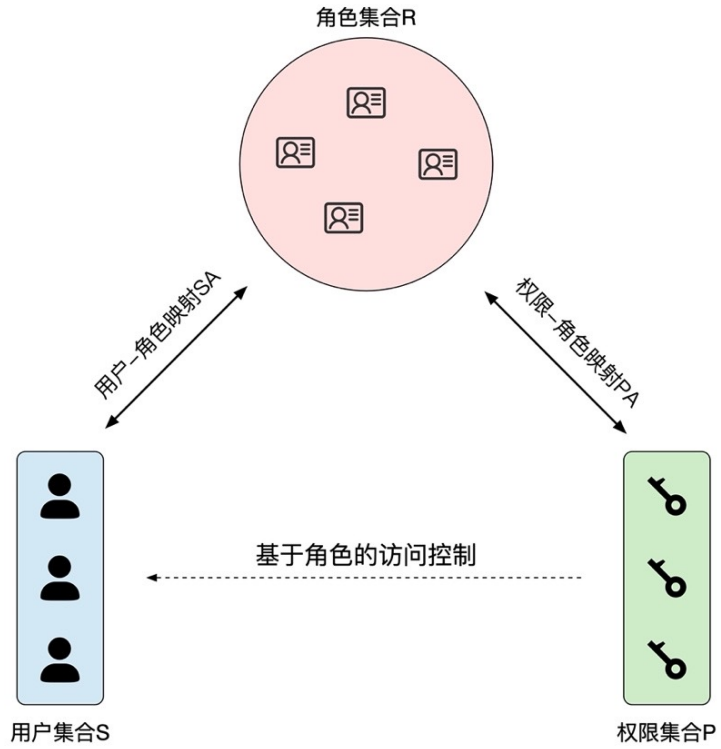
基于角色的访问控制

- **基于角色的访问控制（Role Based Access Control，RBAC）**
  - 根据用户的角色授予访问权限
  - “最小权限”和“权限分离”等关键安全原则
  - 更加高效和安全
- **角色**
  - 一个授予用户或其他角色相关权限的命名组
  - 每个角色名称必须是唯一的
  - 将一组相关的权限授予一个角色，然后将该角色授予用户就可以实现权限分配
  - 简化了用户系统和对象权限的管理
- **例如企业网盘**



基于角色的访问控制

- **组成部分**
  - 用户集合S→用户表
  - 权限集合P→权限表
  - 角色集合R→角色表
  - 用户-角色映射SA→用户-角色映射表： 多对多
  - 权限-角色映射PA→权限-角色映射表： 多对多
- **根据用户的访问需求，将用户分成具有共同访问需求的角色**
  - 角色尽可能简单，具有不同的优先级
  - 简化了基础操作，通过属性自动为新用户分配访问权限（可以用来快速入职）
  - 简化管理，快速重新分配权限，降低高级访问控制的成本



基于角色的访问控制

- 用户（USER）主要用来连接数据库，创建、访问数据库对象和执行SQL语句。
- 角色（ROLE）通常用来组织和管理权限。
- 通过将多种权限打包成角色授予用户，使得用户获得该角色的所有权限；若改变角色的权限，则继承该角色的用户权限会被自动修改。一个用户也可以继承多个不同角色的权限。
- 使用角色控制哪些用户拥有哪些对象的哪些权限，可以大大简化权限的管理。

- 权限表
  - 名称：创建客户 资源: 客户信息 操作：创建
  - 名称：删除客户 资源: 客户信息 操作：删除
  - 名称：查看客户 资源: 客户信息 操作：查看
  - 名称：修改客户 资源: 客户信息 操作：修改
- 角色表
  - 名称：销售角色
    - 权限: 创建客户、删除客户、查看客户、修改客户
- 用户表
  - 主体：小王 角色: 销售角色

角色管理

- **创建用户/角色：**
  CREATE USER/ROLE role_name [[WITH] option [...]]
  [ENCRYPTED | UNENCRYPTED] { PASSWORD | IDENTIFIED BY } { 'password' [EXPIRED] | DISABLE };
  注：UNENCRYPTED已禁止使用
- **修改用户/角色：**
  ALTER USER/ROLE role_name [[WITH] option [...]];
- **删除用户/角色：**
  DROP USER/ROLE [IF EXISTS] role_name [, ...];
- **授予/回收用户/角色：**
  GRANT role_name TO role_name [WITH ADMIN OPTION];
  REVOKE [ADMIN OPTION FOR] role_name FROM role_name [CASCADE | RESTRICT];

子句option语法为：

{SYSADMIN | NOSYSADMIN}
| {MONADMIN | NOMONADMIN}
| {OPRADMIN | NOOPRADMIN}
| {POLADMIN | NOPOLADMIN}
| {AUDITADMIN | NOAUDITADMIN}
| {CREATEDB | NOCREATEDB}
| {USEFT | NOUSEFT}
| {CREATEROLE | NOCREATEROLE}
| {INHERIT | NOINHERIT}
| {LOGIN | NOLOGIN}
| {REPLICATION | NOREPLICATION}
| {INDEPENDENT | NOINDEPENDENT}
| {VCADMIN | NOVCADMIN}
| {PERSISTENCE | NOPERSISTENCE}

角色管理

例1：创建角色role1和role2，密码均为****。**

```
CREATE ROLE role1 WITH password ******';
CREATE ROLE
CREATE ROLE role2 WITH password ******';
CREATE ROLE
```

例2：修改角色role1具有LOGIN属性。

```
ALTER ROLE role1 WITH LOGIN;
ALTER ROLE
```

例3：查看系统表authid或系统视图roles获取角色的相关信息。

```
SELECT oid,rolname,rolcanlogin,rolpassword FROM
pg_roles WHERE rolname='role1' or rolname='role2';

oid    | rolname  | rolcanlogin | rolpassword
---------+------------+----------------+-----------------
 16442 |  role2       | f                | ********
 16439 |  role1       | t                | ********
```

例4：将角色role1授予role2，则角色role2属于组role1，继承role1的相应权限，查看系统表auth_members获取角色成员关系。

```
GRANT role1 TO role2;
GRANT ROLE
SELECT * FROM pg_auth_members;
  roleid  | member | grantor  | admin_option
----------+------------+-----------+--------
-----------
  16439  |  16442    |        10   | f
```

例5：删除角色role1和role2。

```
DROP ROLE role1,role2;
DROP ROLE
```

系统权限

- 系统权限：又称用户属性，指系统规定用户使用数据库的权限，比如连接数据库（LOGIN），创建数据库（CREATEDB），创建用户（CREATEROLE）等。
- 通过CREATE USER/ROLE和ALTER USER/ROLE来指定系统权限；系统权限不能通过角色被继承。

例1：创建角色role1，同时授予role1创建用户的权限和创建数据库的权限。

```
CREATE ROLE role1 WITH CREATEROLE CREATEDB password  "*******";
```

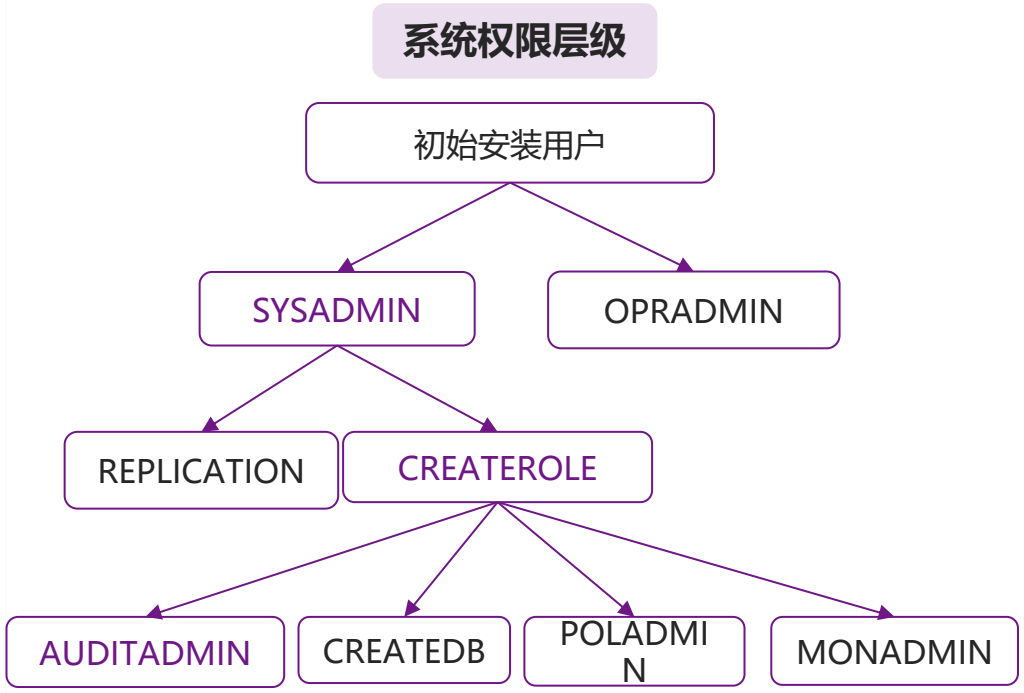
例2：授予用户role1监控管理员的权限，同时取消创建数据库的权限。

```
ALTER ROLE role1 WITH MONADMIN NOCREATEDB;
```

例3：查看系统表authid或系统视图roles获取角色的相关信息。

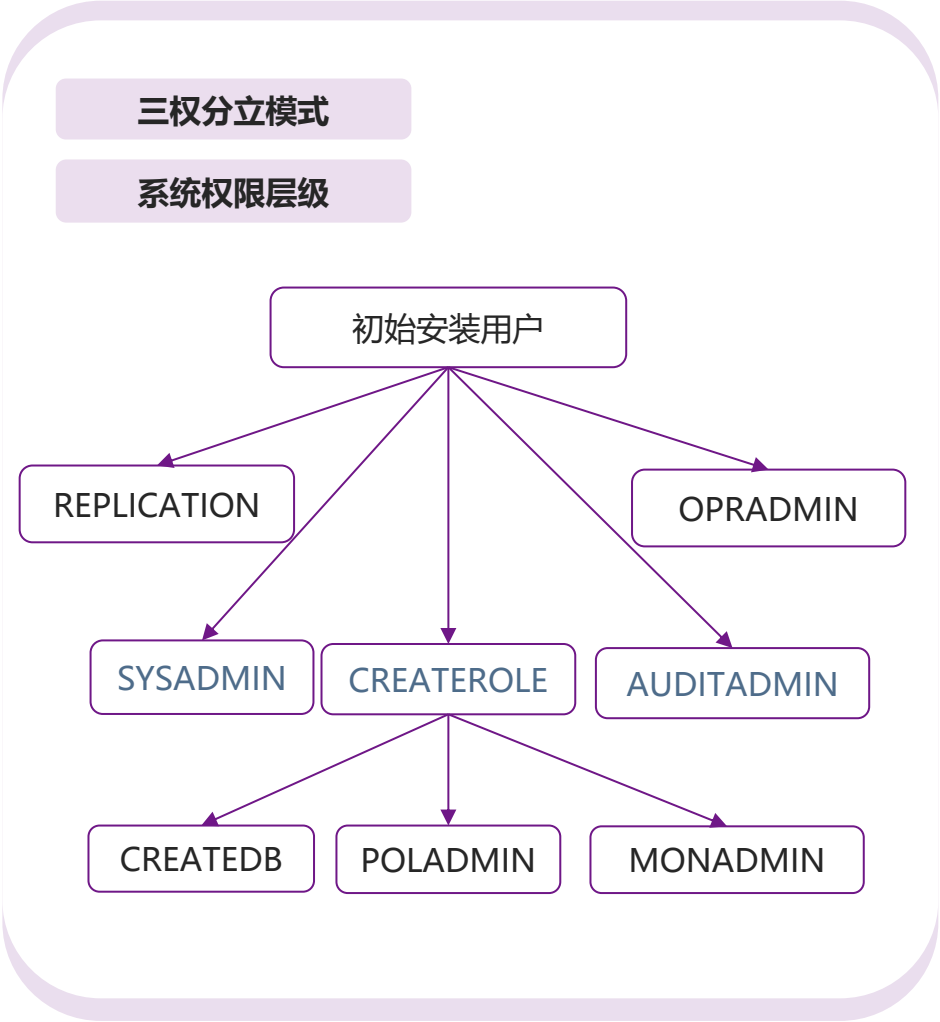
```
SELECT rolname,rolcreaterole,rolcreatedb,rolmonitoradmin FROM pg_roles WHERE rolname=  'role1';
 rolname | rolcreaterole | rolcreatedb | rolmonitoradmin
-----------+-----------------+---------------+----------------------
 user1      | t                | f              | t
```

权限管理



- **初始安装用户**：集群安装过程中自动生成的帐户，拥有系统的最高权限，能够执行所有的操作。
- **SYSADMIN**：系统管理员权限，权限仅次于初始安装用户，默认具有与对象所有者相同的权限，但不包括监控管理员权限和运维管理员权限。
- **MONADMIN**：监控管理员权限，具有监控模式dbe_perf及视图和函数的访问权限和授予权限。
- **OPRADMIN**：运维管理员权限，具有使用Roach工具执行备份恢复的权限。
- **CREATEROLE**：安全管理员权限，具有创建、修改、删除用户/角色的权限。
- **AUDITADMIN**：审计管理员权限，具有查看和维护数据库审计日志的权限。
- **CREATEDB**：具有创建数据库的权限。
- **POLADMIN**：安全策略管理员权限，具有创建资源标签，数据动态脱敏策略和统一审计策略的权限。

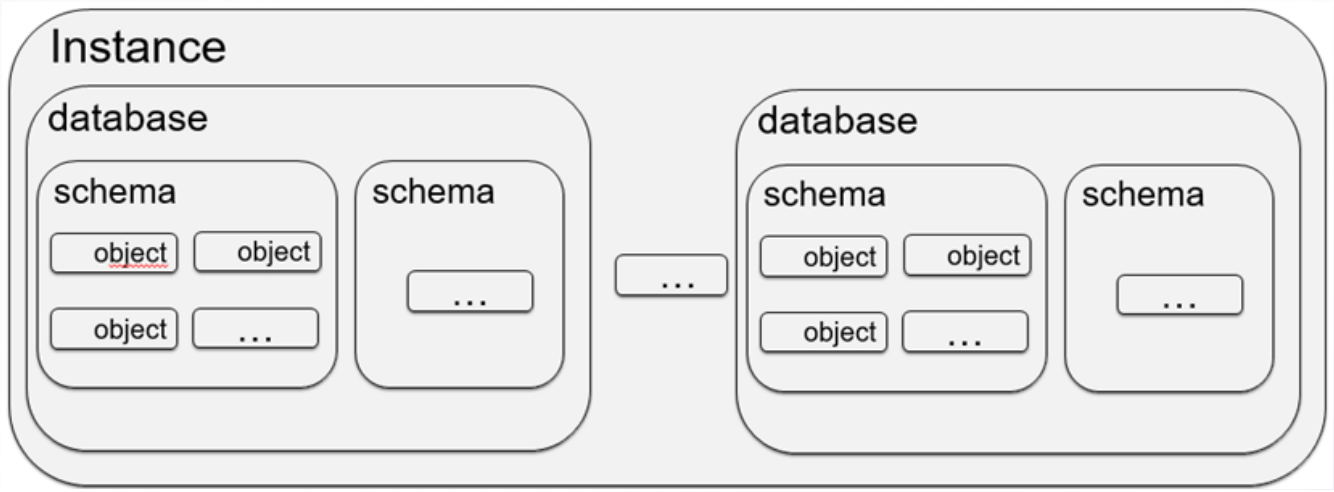
系统权限-三权分立



- **SYSADMIN**：系统管理员权限，不再具有创建、修改、删除用户/角色的权限，也不再具有查看和维护数据库审计日志的权限。
- **CREATEROLE**：安全管理员权限，具有创建、修改、删除用户/角色的权限。
- **AUDITADMIN**：审计管理员权限，具有查看和维护数据库审计日志的权限。
- **一个用户/角色只能具有SYSADMIN、CREATEROLE， AUDITADMIN中的一项系统权限。**

对象权限

- 对象权限：指在数据库、Schema、表等数据库对象上执行特定动作的权限，比如：SELECT、INSERT、UPDATE、DELETE、CONNECT等。
- 针对不同的数据库对象有不同的对象权限，相应地可以被授予用户/角色。
- 通过GRANT/REVOKE来传递对象权限，对象权限可以通过角色被继承。



对象权限

例1：将对表tbl进行select的权限以及将select再赋权的权限授予用户user1

```
GRANT select ON TABLE tbl TO user1 WITH GRANT OPTION;
```

赋权后用户user1可以对tbl进行查询（SELECT）操作且user1有权限将select权限再赋予其他用户

例2：将对表tbl进行alter和drop的权限赋给用户user1

```
GRANT alter, drop ON TABLE tbl TO user1;
```

赋权后用户user1可以对tbl进行修改（ALTER）和删除（DROP）操作。

例3：撤销用户user1对表tbl进行select的权限

```
REVOKE select ON tbl FROM user1;
```

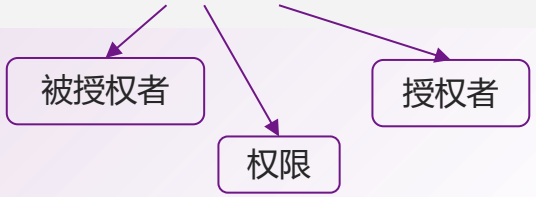
撤销后用户user1对tbl进行select操作会报错：ERROR: permission denied for relation tbl.

对象权限-授权与回收

例：查看系统表class的relacl字段获取表相关的授权信息。

```
select relacl from class where relname='tbl';

------------------------------------------------------------------------
{omm=arwdDxt/omm,  user1=r*/omm,  user2=r/user1}
```



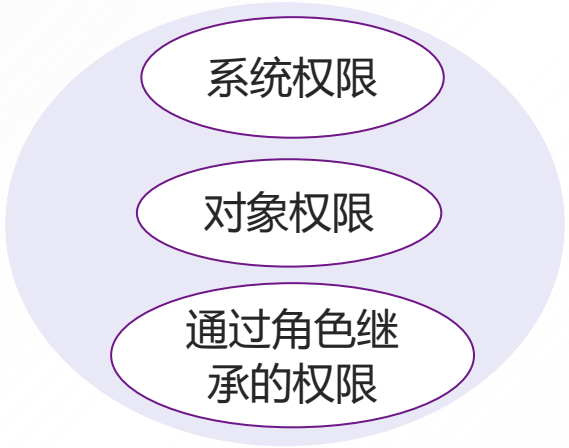
- 表tbl为用户omm所拥有的表，默认具有表tbl的所有权限。
- 用户user1具有表tbl的select的权限以及将select再赋权的权限。
- 用户user2具有表tbl的select权限。

参数	参数说明	参数	参数说明
r	SELECT	C	CREATE
w	UPDATE	c	CONNECT
a	INSERT	T	TEMPORARY
d	DELETE	A	ALTER
D	TRUNCATE	P	DROP
x	REFERENCES	m	COMMENT
t	TRIGGER	i	INDEX
X	EXECUTE	v	VACUUM
U	USAGE	*	再授权选项

第31位	第30位	第29位	第28位	第27位	第26位	第25位	第24位	第23位	第22位	第21位	第02位	第19位	第18位	第17位	第16位	第15位	第14位	第13位	第12位	第11位	第10位	第9位	第8位	第7位	第6位	第5位	第4位	第3位	第2位	第1位	第0位
Grant Option																0	W	R	p	c	T	C	U	X	t	x	D	d	w	r	a

权限管理

- **一个用户/角色拥有的权限集合：**



- **权限最小化原则**：只授予用户/角色完成任务所需要的最小权限，不要随便授予ALL权限。
- **管理好权限周期**：及时使用REVOKE回收不再需要的权限。

基于属性的访问控制

基于属性的访问控制（attribute-based access control, ABAC)

- 基于用户、资源和环境的属性进行访问控制决策。属性可以包括用户的角色、组织信息、时间、位置、设备等。管理员可以定义丰富的策略，根据这些属性来决定用户是否有权访问特定资源。
- 属性：任何可以被定义并且可以为其分配值的特性
- 对主体属性、客体属性和环境条件进行评估，按照属性和环境条件之间的规则进行访问控制

例子

- 不允许员工在非工作时间访问公司机密数据

优点

- 适合于需要更细粒度、动态和灵活的访问控制的场景
- 使最大范围的主体能够访问最大范围的客体
- 无需指定每个主体和每个客体之间的单独关系

- 用户属性(比如用户年龄,地址,角色)
- 环境属性(比如时间,地点)
- 操作属性(增、删、改、查)
- 对象资源属性(比如一篇文章)

数据库安全配置策略

安全配置策略

- 合理的安全配置策略可以使数据库免受潜在的安全威胁
- 安全配置策略随着技术、漏洞和安全要求的变化而不断更新

组成部分

- 系统安全配置策略
- 数据安全配置策略
- 用户安全配置策略
- 密码管理策略
- 审计配置策略

用户类型	用户需求	安全配置策略
终端用户	访问权限批量管理	利用角色进行分组管理
	个性化访问需求	权限单独授予
管理员	访问数据库	定期修改密码，保护对数据库的访问
	管理员权限分离	划分多个管理角色
应用程序开发人员	创建数据库对象	授予特定系统权限
	不对生产数据库产生影响	仅限于测试数据库进行开发
应用程序管理员	创建并管理应用程序角色	授予特定系统权限
	创建和管理应用程序的对象	
	维护和更新应用程序代码	

系统安全配置策略

- **包含数据库用户管理策略和操作系统安全配置策略**
- **数据库用户管理策略**
  - 系统管理员的权限可以进行分解，三权分立
    - 系统管理员负责对数据库中的表、数据等对象进行管理
    - 安全管理负责权限的管理、授权和撤销
    - 审计管理员负责管理数据库审计，维护审计日志
- **操作系统安全配置策略**
  - 管理员必须具有操作系统权限才能创建和删除文件
  - 普通数据库用户不应该具有操作系统权限
  - 操作系统标识了用户的数据库角色，安全管理员必须具有操作系统权限才能修改操作系统帐户的安全域

数据安全配置策略

- 确定用户是否有权访问数据或资源
- 确定允许用户对数据执行何种操作类型
- 取决于数据库中数据的安全级别
  - 最低安全性：允许任何用户创建任何对象，控制访问权限
  - 最高安全性：管理员是唯一有权创建对象并将权限授予用户
- 实现方法
  - 权限管理：细粒度地定制数据安全配置策略
  - 视图：通过视图授权用户，防止包含敏感数据的列被无权限的用户访问

用户安全配置策略

- **通用用户安全配置策略**
  - 密码安全
  - 权限管理
- **终端用户安全配置策略**
  - 用户分组管理
  - 权限合理分配
- **管理员安全配置策略**
  - 管理员权限分割，如“三权分立”
- **应用程序开发人员安全配置策略**
  - 授予特定权限，如创建表
  - 只可访问测试数据库
- **应用程序管理员安全配置策略**
  - 为应用程序创建并管理角色
  - 创建和管理数据库的对象
  - 维护和更新应用程序代码

密码管理策略

口 基本的密码安全规则

- 密码不可包含用户的用户名
- 密码至少由8位字符组成
- 密码至少要包含大写字母、小写字母、数字和特殊符号
- 密码应该定期更换，例如每两个月更新一次密码

口 按需指定细粒度密码管理策略

数据库审计

- 为在特定命名空间、表或行中采取的操作或影响特定内容的操作进行记录，以便在事后进行追踪和问责
- 调查可疑活动，审计与数据库的所有未被授权的连接，以及在数据库执行的所有未被授权用户执行的非法操作
- 监视和收集有关特定数据库活动的数据
- 审计还可以根据需要进行定制，可以对指定用户的数据库操作进行监视和记录。数据库审计记录可以仅包含必要的信息， 例如运行的SQL类型；也可以生成包含名称、应用程序、时间等因素的详细记录。当访问或更改数据库中的指定对象（包括内容）时，会触发审计机制，记录用户的相关操作。



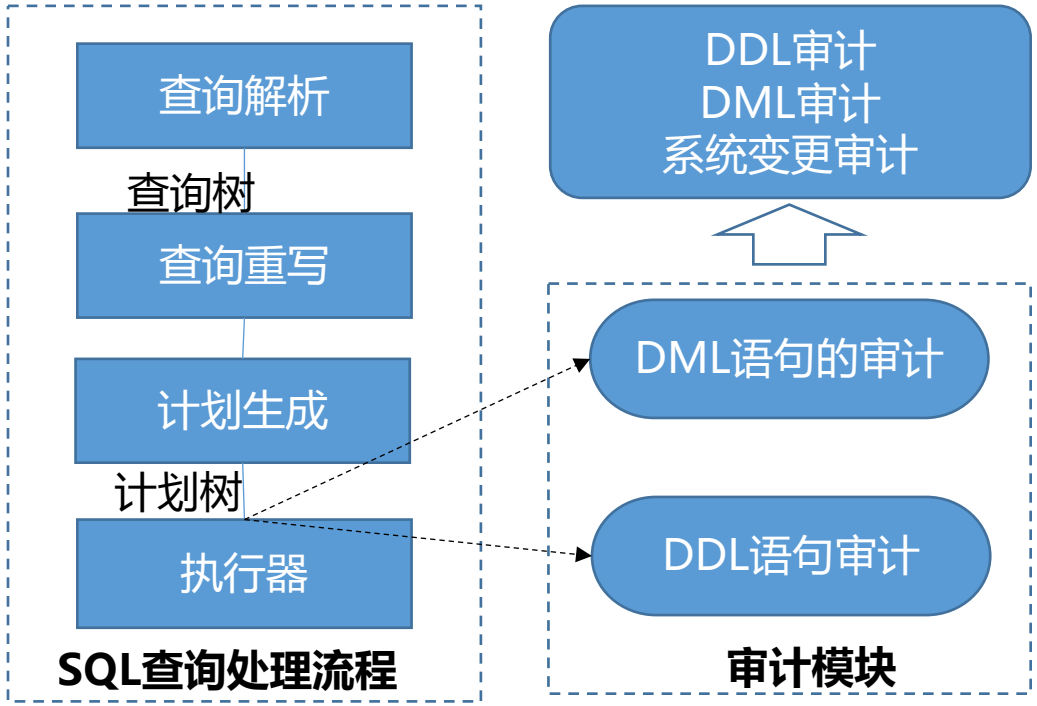
数据库审计

数据库审计

- 跟踪数据库记录和权限的使用
- 监视所有用户对数据的操作并记录到审计日志中
- 例如：数据被执行了何种操作、执行操作的帐户和时间

应用场景

- 为操作进行记录，以便在事后进行追踪和问责
- 调查可疑活动
- 监视和收集有关特定数据库活动的数据
- 对选定用户的数据库操作进行监视和记录



数据库审计记录

口 审计记录

- 包括被审计的操作、执行操作的用户以及操作的日期和时间等

口 审计类型

- 语句审计：按照SQL语句的类型进行审计
- 权限审计：针对一些系统权限进行审计
- 对象审计：在选定数据库对象上，对特定语句进行审计
- 细粒度审计：根据内容审核数据访问和操作

口 不同类型的审计对应不同类型的审计记录

审计记录的产生

- **语句审计**
  - 审计所有数据库用户或仅审计选定用户的活动
  - 主要是针对DDL语句和DML语句
- **权限审计**
  - 只有当权限不足时才会触发权限审计
- **命名空间对象审计**
  - 审计命名空间对象权限允许的语句，以及权限授予和撤销的语句
  - 针对所有数据库用户的，不能指定用户
- **细粒度审计**
  - 查询返回的任何行与审计条件匹配即触发审计
  - 未指定相关列，则会应用于所有列

审计追踪

口 数据库审计追踪：使用存储在数据字典中的记录进行审计

- 数据字典中一张特殊的表用来保存审计记录
- 审计追踪包含不同类型的信息，取决于审计的事件和审计选项
- 例如：用户名、实例编号、访问对象的名称等

口 操作系统审计追踪：使用存储在操作系统文件中的记录进行审计

- 审计记录将写入数据库外部的文件中，而非存储在数据字典中
- 无论是否启用数据库审计，一些与数据库相关的操作总是会被审计并记录到操作系统
  - 例如：数据库启动时的操作系统用户、用户的终端标识符、日期和时间戳以及数据库审计功能是否启用等

审计配置管理

- 提供审计总开关audit_enabled来实现系统级审计控制，支持动态加载。
- 各个审计配置项通过对应开关控制，累计提供11个审计配置项，支持动态加载。

配置项	审计开关	描述	默认值
用户登录、注销审计	audit_login_logout	默认值为7，表示开启用户登录、退出的审计功能。设置为0 表示关闭用户登录、退出的审计功能。不推荐设置除0和7之 外的值	7
数据库启动、停止、恢复和切换审计	audit_database_process	默认值为1，表示开启数据库启动、停止、恢复和切换的审计 功能	1
用户锁定和解锁审计	audit_user_locked	默认值为1，表示开启审计用户锁定和解锁功能	1
用户访问越权审计	audit_user_violation	默认值为0，表示关闭用户越权操作审计功能	0
授权和回收权限审计	audit_grant_revoke	默认值为1，表示开启审计用户权限授予和回收功能	1
数据库对象的CREATE，ALTER， DROP操作审计	audit_system_object	默认值为12295，表示只对DATABASE、SCHEMA、USER、 DATA SOURCE、NODE GROUP这五类数据库对象的 CREATE、ALTER、DROP操作进行审计	12295
具体表的INSERT、 UPDATE和DELETE操作审计	audit_dml_state	默认值为0，表示关闭具体表的DML操作（SELECT除外）审计功能	0
SELECT操作审计	audit_dml_state_select	默认值为0，表示关闭SELECT操作审计功能	0
COPY审计	audit_copy_exec	默认值为1，表示打开copy操作审计功能	1
存储过程和自定义 函数的执行审计	audit_function_exec	默认值为0，表示不记录存储过程和自定义函数的执行审计日志	0
SET审计	audit_set_parameter	默认值为1，表示记录set操作审计日	1

审计配置管理

- 对于audit_system_object，需要清晰了解数据库内部系统对象。
- 通过审计配置项和审计日志配置，用户可以监控整个系统的操作行为，并控制审计日志整体对资源的消耗。
- 在默认配置场景下，审计日志对性能影响较小。
- 在执行器进行DML、DDL审计

配置项	含义	默认值
audit_directory	审计文件的存储目录。	$audit
audit_resource_policy	审计日志的保存策略。	on（表示使用空间配置策略）
audit_space_limit	审计文件占用的磁盘空间总量。	1GB
audit_file_remain_time	审计日志文件的最小保存时间。	90
audit_file_remain_threshold	审计目录下审计文件的最大数量。	1048576

查看删除审计

- 查看、删除审计日志需要具有auditadmin属性(审计管理员)的用户才可以查看。
- 传统审计日志定义了审计的行为，并没有关联具体的用户、具体的访问源，部分压力场景下日志刷新较快。

通过query_audit(‘start-time’, ‘end_time’) ; 查看审计日志。

```
select * from pg_query_audit('2021-05-15 08:00:00', '2021-05-15 16:00:00') limit 1;
        time         |    type     | result | userid | username | database | client_conninfo | object_name |        detail_info        |  node_name   |
thread_id          | local_port | remote_port
-----------------------+------------+--------+--------+----------+----------+-----------------+-------------+---------------------------+--------------+--------------------------------+----
--------+-------------
 2021-05-15 08:00:00+08 | user_logout | ok     | 10     | zhujw    | postgres | gs_clean@::1    | postgres    | logout db(db) success | coordinator1 |
139867939927808@674352000063410 | 8601       | 58426
```

通过delete_audit (‘start-time’, ‘end_time’) ; 删除审计日志。

```
select pg_delete_audit('2016-06-23 12:00:00', '2016-06-23 16:20:00');
 delete_audit
----------------
```

数据库统一审计

口 传统审计：

- 需要定义每个审计对象
- 每个单独的组件都有单独的审计跟踪
- 不同层次审计策略配置方式不统一
- 存储格式及位置不统一
- 查看方式不统一

口 统一审计：

- 统一的审计策略
- 以统一格式存放在相同位置，统一通过UNIFIED_AUDIT_TRAIL视图查看
- 提升了审计记录的管理和安全性
- 整体审计性能大大提高
- 简单的审计策略管理方式

数据库统一审计

- 传统审计机制会在并发场景(如高并发)下，导致日志产生较多，更新较快，不利于审计管理员管理。
- 传统审计机制无法区分访问源、访问对象等信息，审计日志无法支撑定向管理。
- **核心思想**：定向定义期望审计的行为，实现对审计行为对象的有效过滤，保障只审计期望的行为。

统一审计语法格式：CREATE AUDIT POLICY

```
CREATE AUDIT POLICY [ IF NOT EXISTS ] policy_name { { privilege_audit_clause | access_audit_clause }
[ filter_group_clause ] [ ENABLE | DISABLE ] };
```

- privilege_audit_clause:
  PRIVILEGES { DDL | ALL } [ON LABEL (resource_label_name [, ...])]
- access__audit_clause:
  ACCESS { DML | ALL } [ON LABEL (resource_label_name [, ...])]
- Filter_group_clause:
  FILTER ON { (FILTER_TPYE (filter_value [, ...])) [, ...]

资源标签（RESOURCE LABEL）：基于数据库对象定义具有某种相同属性的资源集合（例如数据库、表、列、视图、触发器），通过标签进行审计，降低了策略配置的复杂性和冗余程度

```
CREATE RESOURCE LABEL [IF NOT EXISTS] label_name ADD label_item_list[, ...]*;

示例: CREATE RESOURCE LABEL IF NOT EXISTS table_label add TABLE(public.tb_for_label);
      CREATE RESOURCE LABEL IF NOT EXISTS column_label add COLUMN(public.tb_for_label.col1);
```

数据库统一审计

- 只有poladmin，sysadmin或初始用户用户才能进行此操作；
- 需要打开enable_security_policy开关统一审计策略才可以生效；

```
CREATE TABLE tb_for_audit(col1 text, col2 text, col3 text);
CREATE RESOURCE LABEL adt_lb0 add TABLE(tb_for_audit);

--对数据库执行create操作创建审计策略
CREATE AUDIT POLICY adt1 PRIVILEGES CREATE;

--对数据库执行select操作创建审计策略
CREATE AUDIT POLICY adt2 ACCESS SELECT;

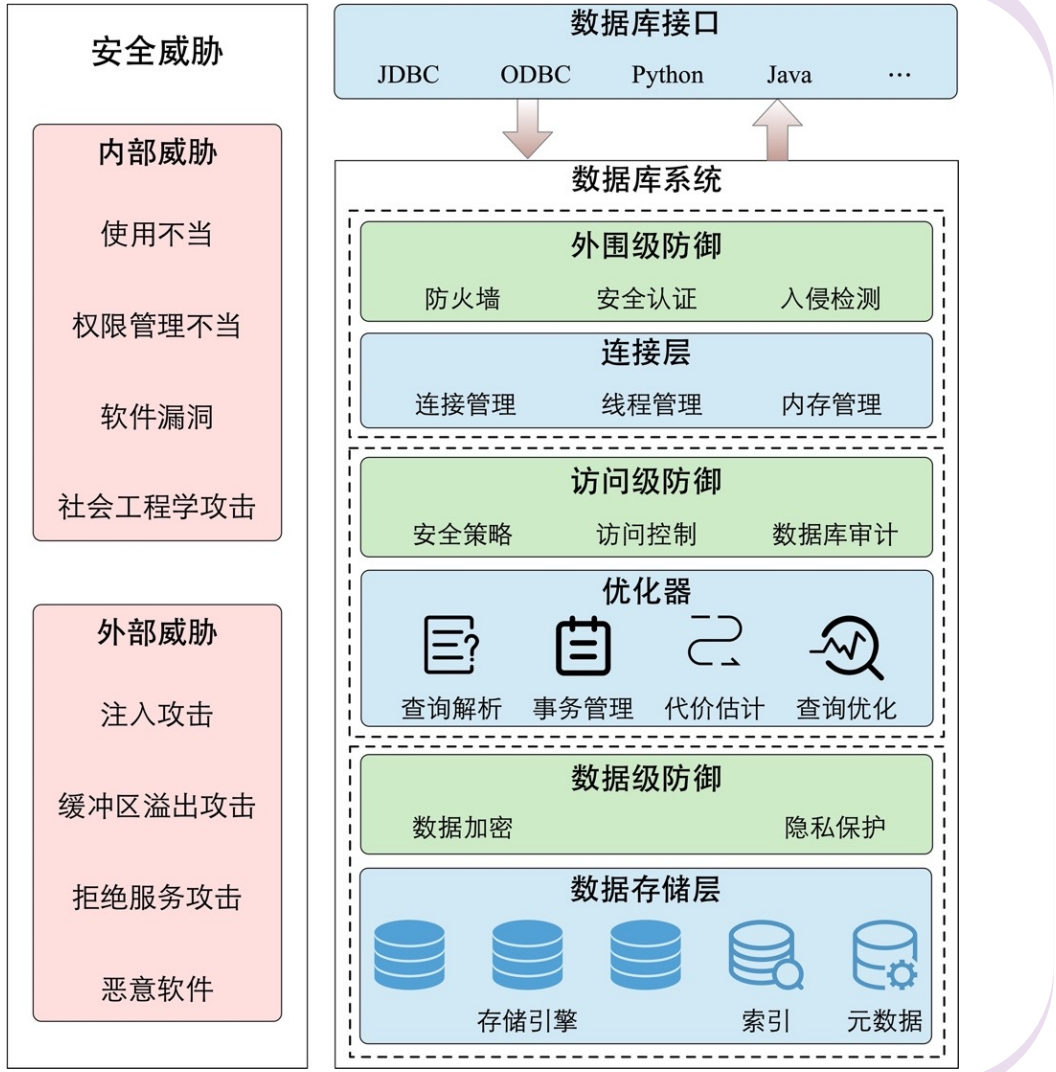
--仅审计记录用户dev_audit和bob_audit在执行针对adt_lb0资源进行的create操作数据库创建审计策略
CREATE AUDIT POLICY adt3 PRIVILEGES CREATE ON LABEL(adt_lb0) FILTER ON ROLES(dev_audit, bob_audit);

--仅审计记录用户dev_audit和bob_audit，客户端工具为psql和gsql，IP地址为'10.20.30.40', '127.0.0.0/24'，在执行针对adt_lb0资源进行的select、insert、delete操作数据库创建审计策略
CREATE AUDIT POLICY adt4 ACCESS SELECT ON LABEL(adt_lb0), INSERT ON LABEL(adt_lb0), DELETE FILTER ON ROLES(dev_audit, bob_audit), APP(psql, gsql), IP('10.20.30.40', '127.0.0.0/24');

--修改统一审计信息，增加adt2审计范围
ALTER AUDIT POLICY adt2 PRIVILEGES ANALYZE;
--删除统一审计策略
DROP AUDIT POLICY adt3;
```

数据库安全提纲

- **外围级防御**
  - 决定哪些用户可以访问数据库
  - 安全认证、防火墙、入侵检测
- **访问级防御**
  - 控制允许哪些用户有权访问某些数据或包含它的系统
  - 安全配置策略、访问控制、审计
- **数据级防御**
  - 预防和应对发生在数据库本身的安全隐患
  - 加密、隐私保护、脱密、防篡改、多方安全



数据加密

- **将数据转换为另一种形式，使其中信息被编码并且只能由拥有正确密钥的用户访问或解密**
  - 未加密的数据称为明文，加密数据称为密文
  - 对于未经许可访问的个人，密文看起来是混乱的或不可读的
  - 数据库加密算法的安全取决于密钥长度
  - 密钥太长会降低数据库吞吐量，占用更多存储空间
- **加密是使用的最广泛和最有效的数据安全方法之一**
  - 如果数据库遭到破坏，数据仍然只有拥有正确加密密钥的用户才能读取

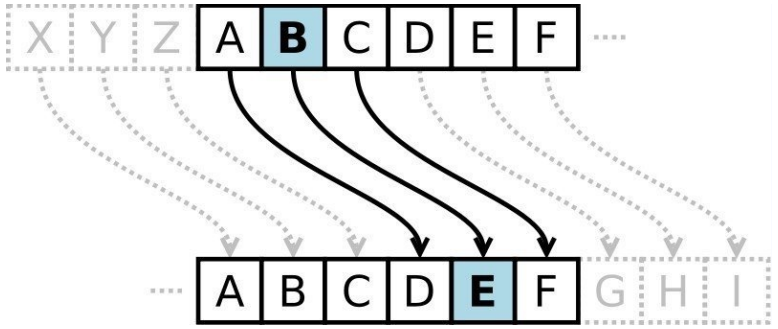
数据加密

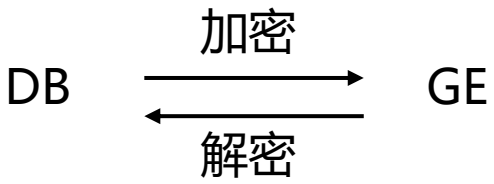
数据加密带来的挑战

- 数据库性能严重下降
- 数据库架构的设计复杂性大大增加

数据加密与数据库

- 数据库系统架构正在为适应加密数据进行改进或重构
- 出现了各种不同的用于数据库加密的算法
- 在数据库运行效率和数据安全性之间权衡





数据库加密维度

存储加密

- 对存储介质上的数据进行加密
- 即使存储介质被盗，也不泄露敏感信息
- 挑战
  - 密文数据膨胀
  - 影响数据的检索和计算效率

计算加密

- 不能影响数据库原有的功能
- 同时对用户和应用程序应该透明
- 密态计算的性能
  - 范围查找
  - 聚集求和

传输加密

- 在发送者和接收者之间提供安全通信信道

数据库加密粒度

- **数据项加密**
  - 每个单独的数据项都有自己唯一的密钥
- **列级加密**
  - 同一列中的数据使用相同的密钥进行加密
  - 使用最广泛的加密粒度
- **表空间级加密**
  - 每个数据表都有自己的密钥
  - 对数据库性能影响相对较小
- **文件级加密**
  - 加密整个数据库文件

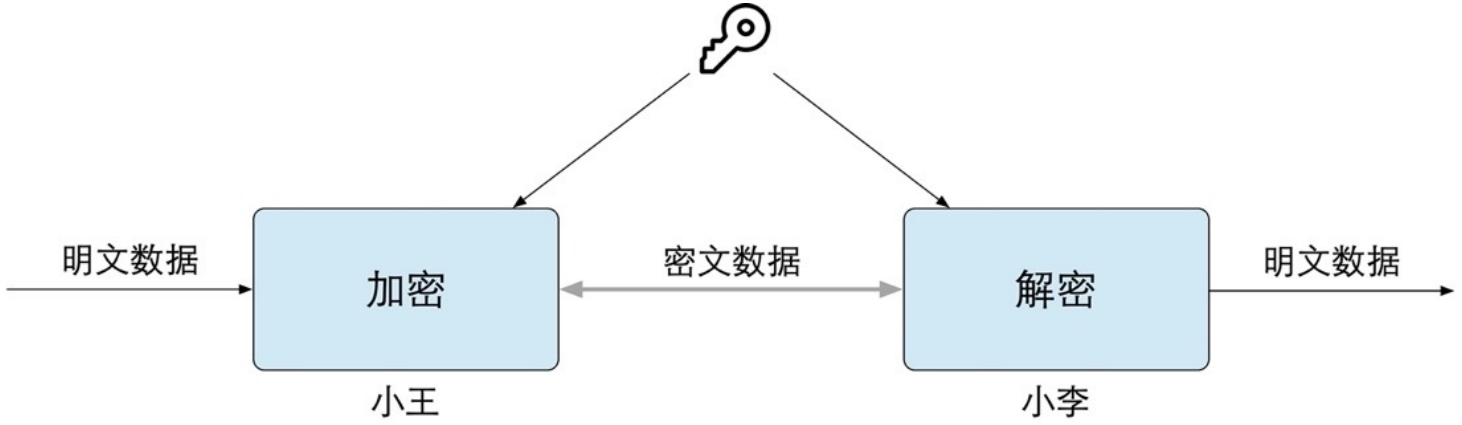


加密粒度越小
安全性越高
性能影响越大

加密算法

对称加密

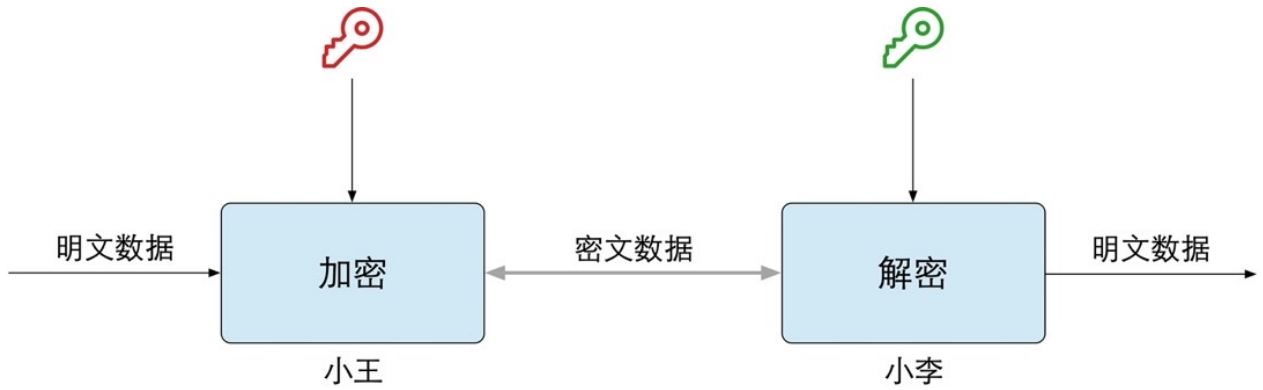
- 发送方和接收方使用相同的密钥
- 缺点：私钥可能被不当共享，导致数据泄露
- 典型算法
  - DES（Data Encryption Standard，数据加密标准）
  - AES（Advanced Encryption Standard，高级加密标准）
  - RC4（Rivest Cipher 4）



加密算法

非对称加密

- 加解密过程中使用两个密钥，一个公钥和一个私钥
- 一个密钥进行加密，另一个用于解密
- 不需要共享私钥，更为安全
- 典型算法：RSA算法



数据库加密实现方式

- **API方法：加密和解密任务由应用程序中的加密代理来处理**
  - 适用于不同数据库产品的应用程序级加密
  - 如果包含加密列，需要在应用程序中进行手动修改
- **插件方法：独立于应用程序和数据库的插件包**
  - 在数据库管理系统上安装加密模块或插件包
  - 独立于应用程序工作
- **透明数据加密（TDE）方法：数据库处理加解密**
  - 在数据库引擎本身内执行加密和解密
  - 对整个数据库（包括备份的数据）进行加密
  - 对使用数据的用户和应用程序是不可见的

全密态：全程加密

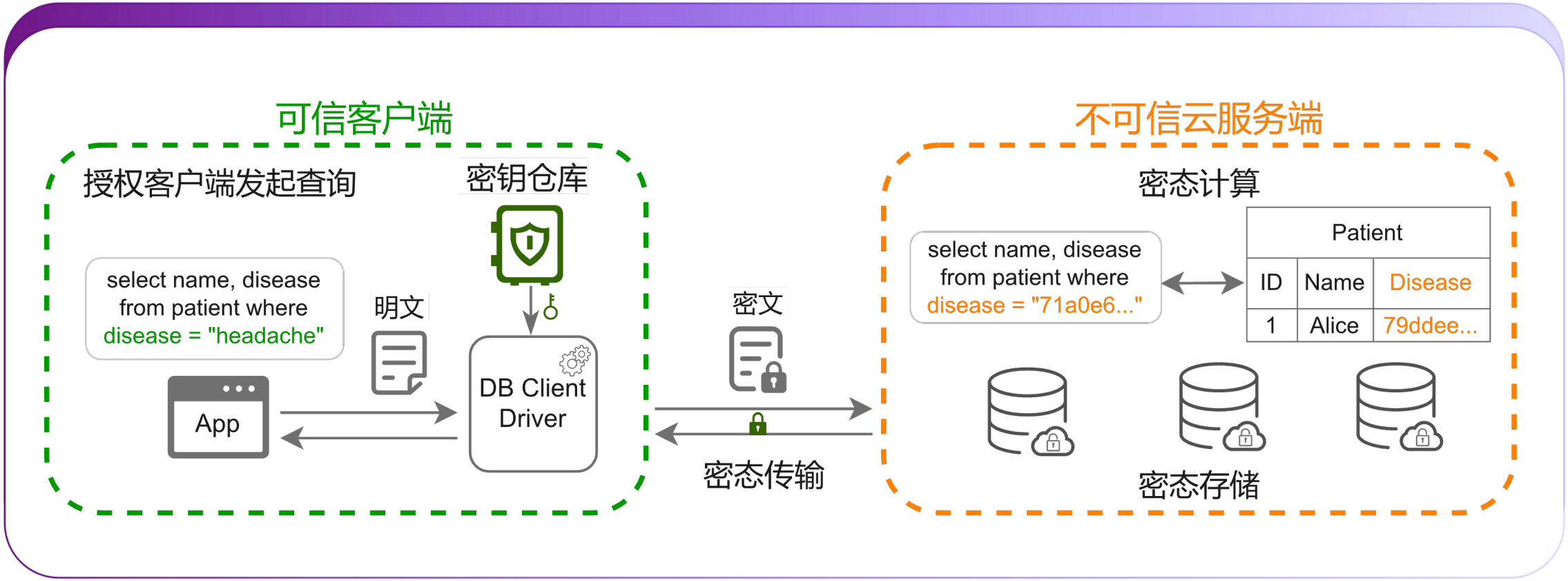
全程加密

- 允许客户端加密客户端应用程序内的敏感数据
- 永远不会向数据库引擎泄露加密密钥
- 管理者不能访问数据

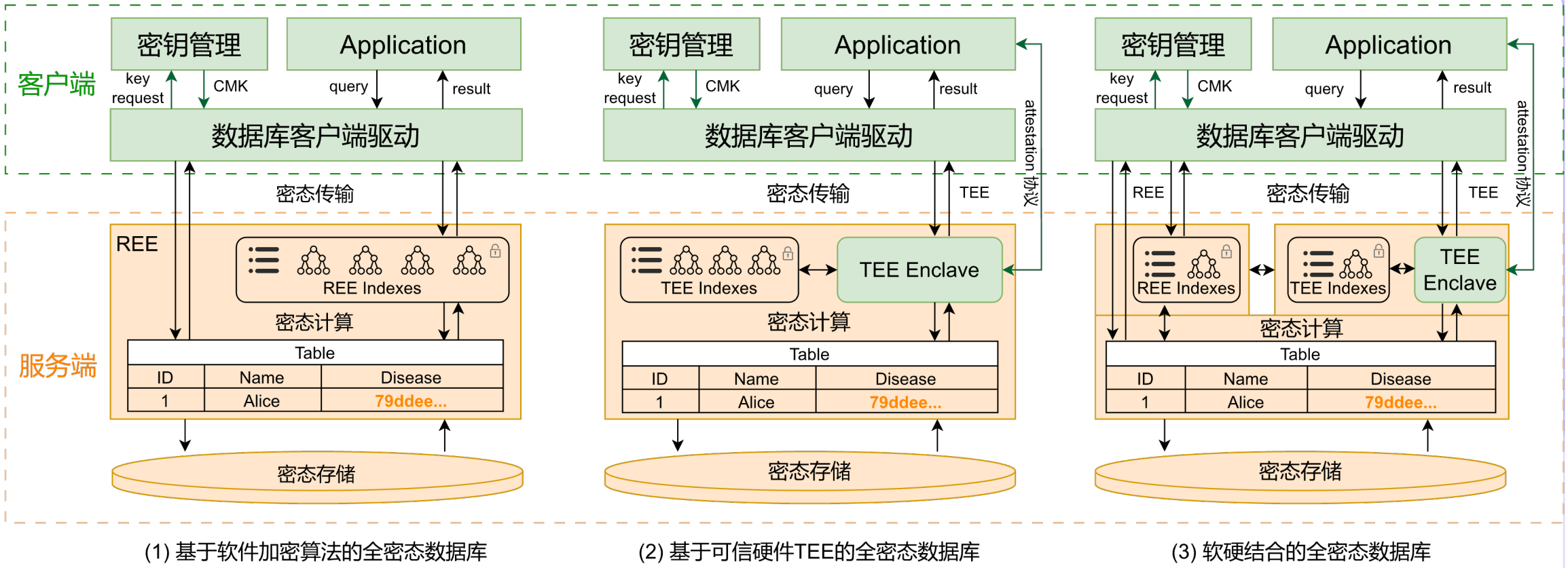
挑战

- 加密方式：如何兼顾安全性和效率
- 密钥管理：如何保护数据库密钥的安全
- 索引：如何利用索引加速检索和计算，特别是范围和LIKE

全密态架构


可信客户端
授权客户端发起查询
密钥仓库
select name, disease
from patient where
disease = "headache"
明文
App
DB Client
Driver
密文
密态传输
不可信云服务端
密态计算
select name, disease
from patient where
disease = "71a0e6..."
Patient
ID
Name
Disease
1
Alice
79ddee...
密态存储

全密态实现方法



(1) 基于软件加密算法的全密态数据库　(2) 基于可信硬件TEE的全密态数据库　(3) 软硬结合的全密态数据库

全密态：秘钥管理

□密钥管理：三层密钥管理系统

- 列加密密钥：对列属性的不同数据进行加密
  - 泄露了一个列加密密钥，只有该密钥加密的属性会受到影响
- 客户端主密钥：用户使用自己客户端主密钥来加密列加密密钥
  - 客户端主密钥永远不会离开用户的可信环境
- 设备密钥：保护不同的客户端主密钥

全密态：纯软方法

支持不同查询算子的多种加密方案

- 等值查询：确定性加密
- 范围查询：OPE
- 求和：同态加密
- ……

洋葱模型

- 同一列上不同算子对应不同加密方案
- 不同加密方案的加密结果都存储在数据库中

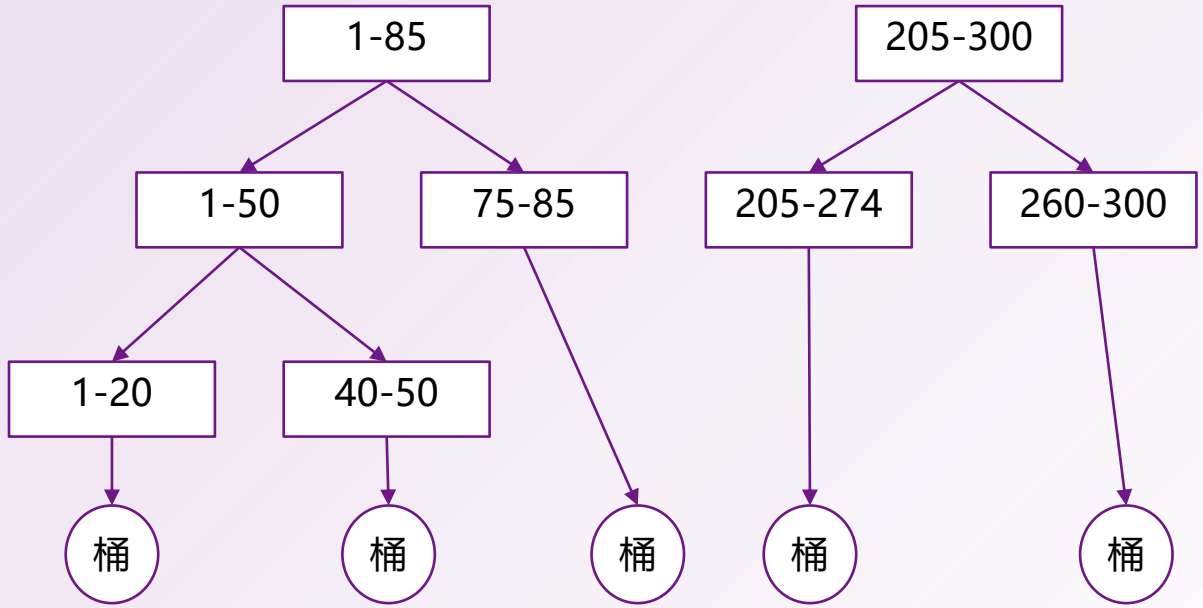
NAME	AGE	GPA
Tom	13	85
Alice	12	89
Jack	14	87

NAME-Eq	AGE-Ord	GPA-Ord	GPA-Add
N1-eq	A1-ord	G1-ord	G1-add
N2-eq	A2-ord	G2-ord	G2-add
N3-eq	A3-ord	G3-ord	G3-add
支持NAME列等值查询	支持AGE列范围查询	支持GPA列范围查询	支持GPA列加法计算

全密态：密态索引

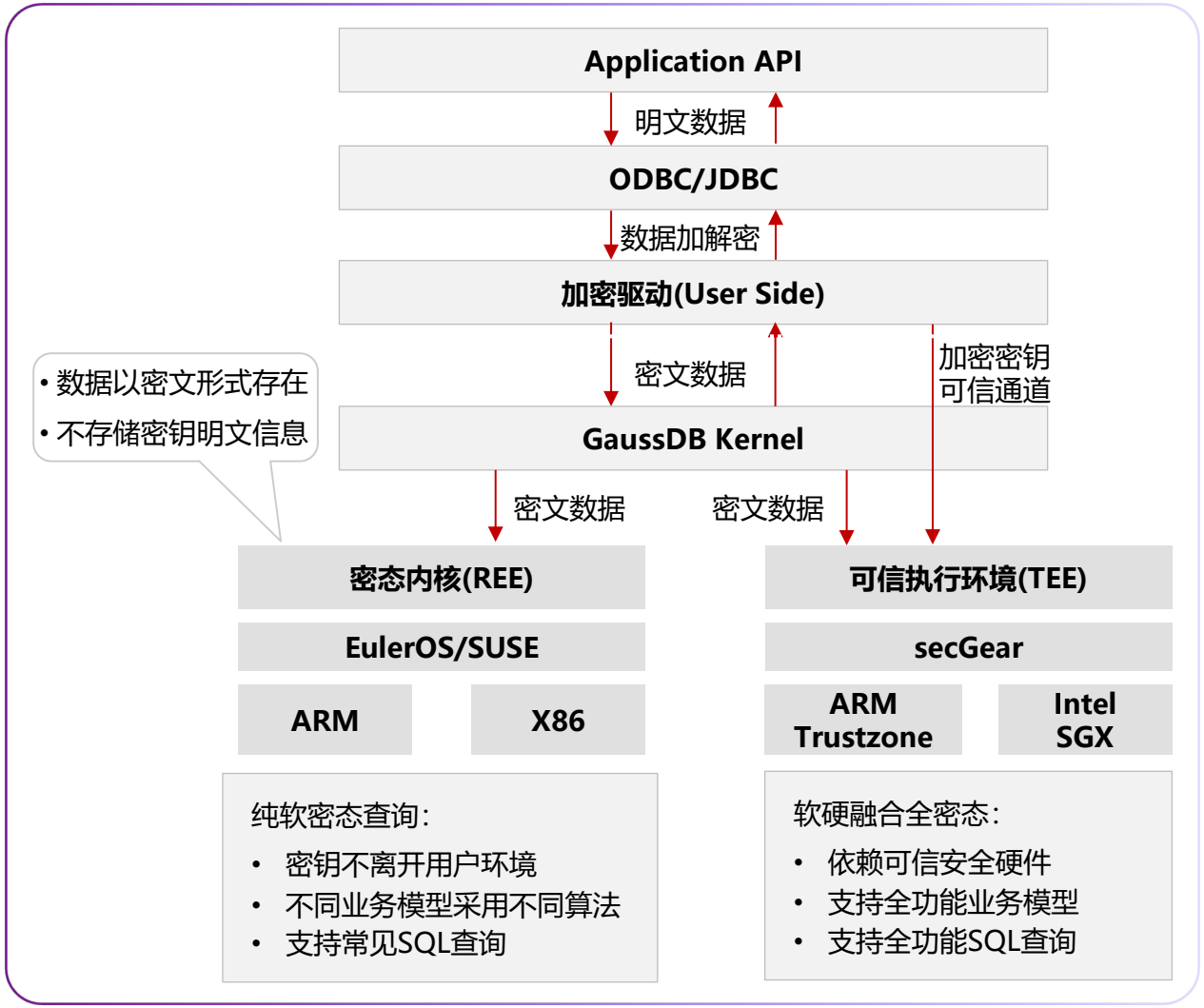
口 密态索引

- 等值索引
  - 确定性加密：构造标准的B+树索引
- 顺序索引
  - 先将数据分成不同“桶”，桶都是有序的



全密态：软硬结合方法

- 借助物理隔离或逻辑隔离明文计算空间，通过**访问控制和代价优化**，实现计算及数据对外“不可见”，防止数据泄露，并保证查询效率
- 减少TEE和REE的数据交换
- 密态索引+TEE执行



隐私保护

数据库隐私

- 对包含在数据库中的数据和数据库本身的保护
- 保护这些信息不被未经许可的第三方访问

数据库隐私主的问题

- 数据库本身的实际安全性
- 存储在数据库中的内容的法律和道德影响
- 数据库安全专业人员承担的内在道德责任

实现方法

- 数据脱敏：创建虚假数据版本的方法，保护敏感数据，如K匿名
- 差分隐私：在数据库存储的数据中加入噪声

隐私保护：数据脱敏

- **目标：创建一个无法破译或逆向工程的版本**
- **优点**
  - 解决了数据丢失、数据泄露、帐户泄露等威胁
  - 降低云数据库上相关的数据风险
  - 保持数据固有的功能特性，但使数据对攻击者无用
  - 允许数据共享，而不会暴露生产数据
- **数据脱敏类型**
  - 静态脱敏：更改所有敏感数据，直到数据可以安全地被共享
  - 确定性脱敏：映射具有相同数据类型的两组数据（不够安全）
  - 动态脱敏：从生产系统传输到测试或开发系统进行脱敏

隐私保护：数据脱敏分类

- **数据加密**
  - 最安全的脱敏形式
  - 实施较为复杂
- **数据加扰**
  - 将数据的内容以随机顺序重新组织，替换原来的内容
  - 安全性差
  - 例如：将值为76498的ID全部替换为84967
- **数据剔除**
  - 当未经授权的用户访问相应数据时，数据显示为缺失或“空”
  - 数据可用性差

隐私保护：数据脱敏技术

- **数据值差异**
  - 使用函数将原始数据替换，隐藏真实的数据
  - 例如，如果客户可以使用购买的商品中最高价格和最低价格之间的值来替换商品的真实价格
- **数据替换**
  - 将数据的值替换为虚假但真实的替代值
  - 例如：真实客户姓名被电话簿中随机选择的姓名替换
- **数据混洗**
  - 使用随机序列在每列中重新排列数据
- **数据匿名化**
  - 删除直接标识符，并且要避免标识符组合在被用来识别个人

隐私保护：数据脱敏实现

- **核心思想：**对生产库返回的数据进行实时脱敏处理，做到即查即脱敏，返回的数据真实有效、安全不泄露隐私、满足最小权限原则。
- **应用场景：**一般情况下系统访问权限无法解决对隐私数据的访问控制，需要为系统提供数据权限控制的能力，限制特定用户对敏感数据的访问。
- 数据动态脱敏特性是保障隐私数据“看不了”，该特性能够标记敏感数据属性列并进行数据脱敏，解决非法用户访问敏感数据窃取敏感信息的问题。
- 传统方法使用列级访问控制限制用户对敏感列的访问，但该方法用户对敏感列不可见，返回数据无法体现全部信息；而使用动态数据脱敏将敏感数据脱敏后呈现，返回信息完整且隐私安全。

CREATE MASKING POLICY policy_name masking_clause[, ...]* policy_filter [ENABLE | DISABLE];

- masking_clause：masking_function ON LABEL(label_name[, ...]*)
- masking_function：maskall | randommasking | creditcardmasking | basicemailmasking | fullemailmasking | shufflemasking | alldigitsmasking
- policy_filter:FILTER ON FILTER_TYPE(filter_value [,...]*)[,...]*

隐私保护：数据脱敏实现

导入原始数据：

```
select * from user_info_table;
 uid |     name     |     tel     |         email          |     creditcard
-----+--------------+-------------+------------------------+--------------------
   1 | Alexander    | 15112455532 | Alex15@huawei.com      | 4511-8454-2178-6551
   2 | Peter Magram | 13745621125 | Peter123Peter@huawei.com | 4211-2264-0000-1545
   3 | Gil Cizer    | 13323546225 | Gil15SR@huawei.com     | 6216-6304-0000-0241
   4 | Yakov        | 15922155312 | Yak0@huawei.com        | 5562-1004-2154-6999
```

定义资源标签：动态数据脱敏机制所需的资源标签仅能是列

```
create policy_label resource label_tel add column(user_info_table.tel);
create policy_label resource label_email add column(user_info_table.email);
create policy_label resource label_creditcard add column(user_info_table.creditcard);
```

定义动态数据脱敏策略：根据不同的数据类型使用不同的脱敏函数

```
create masking_policy mask_telephone alldigitsmasking on label(label_tel);
create masking_policy mask_email basicemailmasking on label(label_email);
create masking_policy mask_creditcard creditcardmasking on label(label_creditcard);
```

查询脱敏后的数据：

```
select * from user_info_table;
 uid |     name     |     tel     |         email          |     creditcard
-----+--------------+-------------+------------------------+--------------------
   1 | Alexander    | 00000000000 | xxxxxx@huawei.com      | xxxx-xxxx-xxxx-6551
   2 | Peter Magram | 00000000000 | xxxxxxxxxxxxx@huawei.com | xxxx-xxxx-xxxx-1545
   3 | Gil Cizer    | 00000000000 | xxxxxxx@huawei.com     | xxxx-xxxx-xxxx-0241
   4 | Yakov        | 00000000000 | xxxx@huawei.com        | xxxx-xxxx-xxxx-6999
```

隐私保护：差分隐私

差分隐私简介

- 让用户在使用数据库时，不泄露数据库中个人的隐私数据
- 在数据中加入适量的随机噪声
- 不仅能更好地保护隐私，也将对数据库的影响降到最低

应用

- 推荐系统、社交网络、地图定位服务

优点

- 当个人数据发生变化时，整个数据集上的统计信息几乎无变化
- 适用于任何个人和任何数据集

隐私保护：差分隐私

口特点

- 可组合：使用多个差分隐私算法的结果仍然是差分隐私的
- 多用户隐私：对不同群体（例如家庭）的隐私信息进行保护
- 对后续过程的稳健性：差分隐私结果上的计算结果仍是差分隐私的

口ε-差分隐私机制

- ε是隐私预算，决定了隐私保护的强度和要引入的噪声量
- ε的值越小，隐私保护强度越高

口实现机制

- 随机化回答

隐私保护：随机化回答

场景：统计学生及格率

- 有n个同学，每个同学i及格与否视为属性$X_i \in \{0,1\}$
- 每同学向班长发送一条消息Y_i
- 及格率$p = \frac{1}{n}\sum_{i=1}^{n} X_i$，及格率的估计$\tilde{p} = \frac{1}{n}\sum_{i=1}^{n} Y_i$

如实回答

- 每个同学如实地发送给班长
- 此时，$Y_i = X_i$，$\tilde{p} = p$
- 同学们的隐私没有得到任何保护

隐私保护：随机化回答

完全随机回答

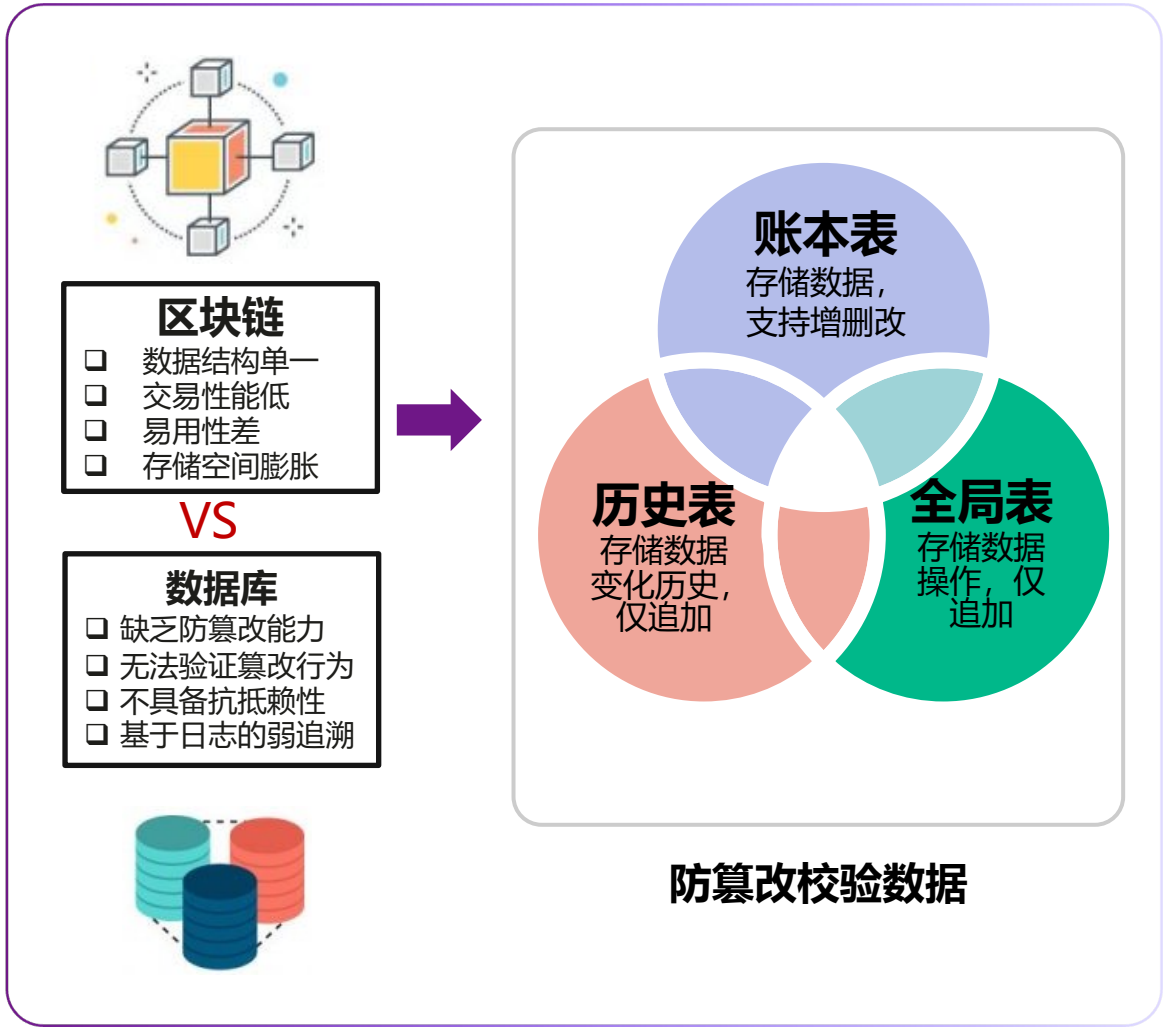
- 每个同学随机地在0和1中选择
- 若选中0，则把X_i发送给班长，否则，把$1 - X_i$发送给班长
- 此时Y_i完全是保密的，无法使用 $\tilde{p}$估计p

随机化回答

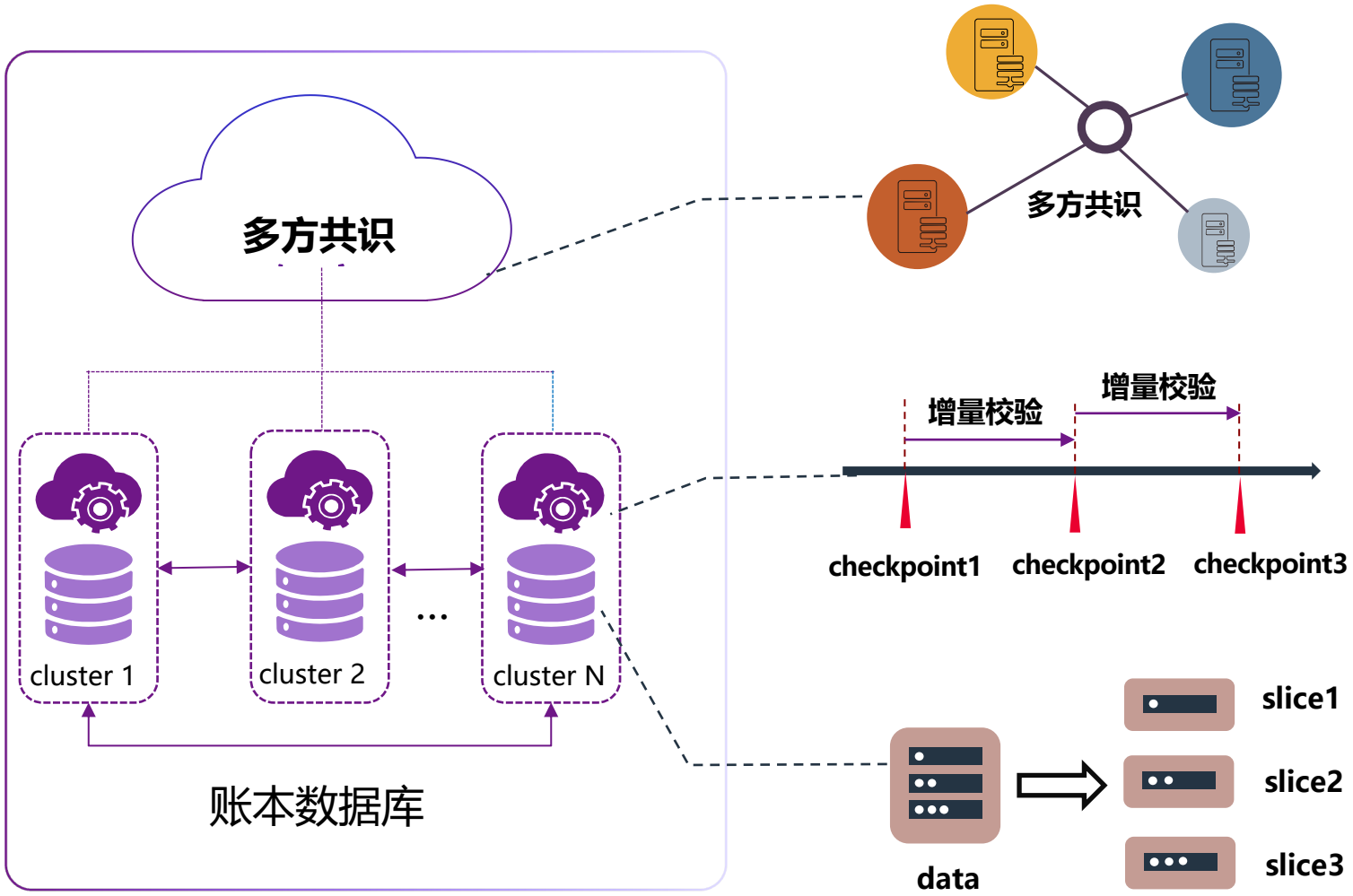
- 随机数$\gamma \in [0, \frac{1}{2}]$
- 以$\frac{1}{2} + \gamma$的概率将X_i发送给班长，以$\frac{1}{2} - \gamma$的概率将$1 - X_i$发送给班长
- 一定程度上能够反映同学们的及格率，同时保护同学们的隐私

数据防篡改

- 防篡改账本数据库：使用Multi-Set方式生成数据及SQL操作的校验数据，提供高性能、强追溯的防篡改
- 对数据增删改操作，系统会记录行级数据变化追溯信息及操作的追溯信息，并通过算法**高效生成校验信息**
- 通过**对校验信息的保护和校验**，起到对重要数据的完整性保护，识别、阻止未授权更改



数据防篡改



多方共识防篡改：

数据的修改需要取得参与节点的一致。

高性能共识协议

高效篡改校验算法：

保证用户在不接触敏感信息的条件下校验数据库中数据的一致性和完整性。

高性能、高并发的篡改校验算法

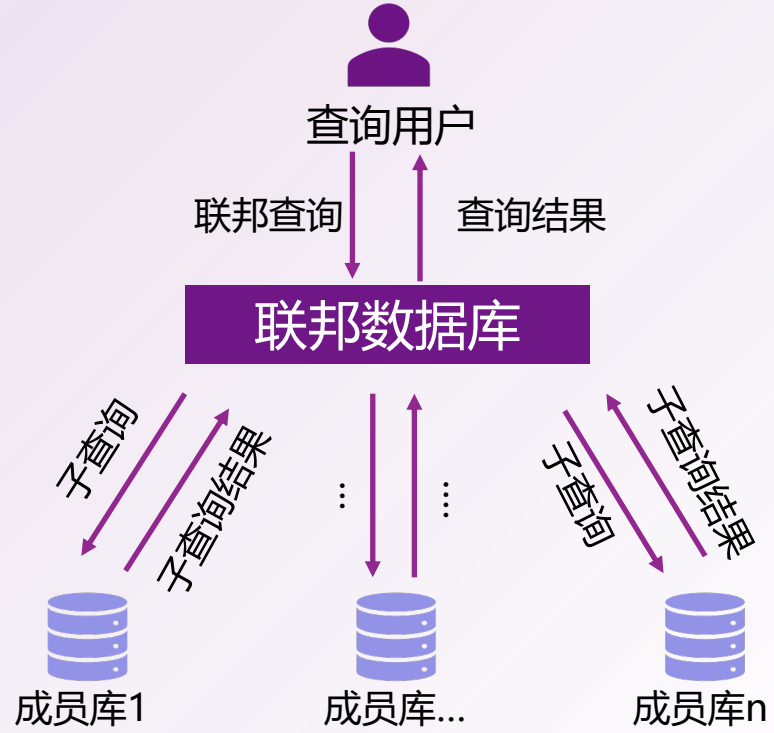
账本数据存储：

忠实记录用户数据操作的完整信息，保证操作链可追溯。

保证追溯信息充分情况下降低存储开销

多方安全计算：联邦查询

- **联邦查询：提供聚合查询服务，如COUNT, AVG**
  - **生成查询**：联邦生成各成员库的查询计划
  - **执行查询**：各成员库内部执行查询获得结果
  - **聚合结果**：联邦负责聚合各成员库返回结果



本章小结

1. **数据库面临的安全威胁**
   - 内部威胁：使用不当，权限管理不当，软件漏洞，社会工程学攻击
   - 外部威胁：注入攻击，缓冲区溢出攻击，拒绝服务攻击，恶意软件
2. **数据库安全需求**
   - 保护数据库中的数据
   - 保护数据库系统
3. **数据库三层防御架构**
   - 外围级安全措施：将安全风险控制在数据库外
   - 访问级安全措施：避免来自内部的威胁造成严重的安全问题
   - 数据级安全措施：对数据提供全生命周期的安全保障