北京金融科技产业联盟
BEIJING FINTECH INDUSTRY ALLIANCE

金融行业全密态数据库研究报告

北京金融科技产业联盟

2022 年 12 月

版权声明

编制委员会

主编

潘润红

编委会成员

郭志军　聂丽琴　林承军

编写组成员（按姓氏拼音排序）

曹　健　冯　程　郭　亮　江智睿　彭卫华　苏　强
汪　晟　王　栩　王　枫　杨　锐　赵　琼

主审

黄本涛

统稿

张　蕾

参编单位：

交通银行股份有限公司、中国工商银行股份有限公司、北京金融科技产业联盟、华为技术有限公司、阿里云计算有限公司、腾讯云计算（北京）有限责任公司、百度云计算技术（北京）有限公司、北京奥星贝斯科技有限公司

目 录

一、研究背景 1
二、主流安全技术 3
（一）访问控制 4
（二）全链路加密 7
（三）数据脱敏 16
（四）数据库审计 19
（五）数据库防火墙 22
（六）数据库漏洞扫描 25
三、全密态数据库技术 26
（一）硬件全密态技术 26
（二）软件全密态技术 26
（三）其他相关技术 26
四、产品现状与应用场景 30
（一）产品现状 26
（二）应用场景 26
五、应用挑战与应对 33
（一）保障密态计算机制的安全性 26
（二）实现业务的无缝迁移或轻量化迁移 26
（三）避免服务切换带来的性能损耗 26
六、研究情况总结 34

一、研究背景

数据库是数据存储的主要技术手段，数据库系统在整个IT架构中的重要地位不言而喻。攻击数据库，窃取信息是当今信息系统的一个主要安全问题。数据泄露或被篡改等安全事件对企业和组织带来的损失将是多方面的，既有可量化的经济损失，也有不可量化的品牌信誉和业务影响。因此，确保数据库安全越来越重要。

数据库安全，是指以保护数据库系统、数据库服务器和数据库中的数据、应用、存储，以及相关网络连接为目的，是防止数据库系统及其数据遭到泄露、篡改或破坏的安全技术。与传统的网络安全防护体系不同，数据库安全技术更加注重从客户内部的角度做安全，即信息安全。其内涵包括了保密性、完整性和可用性，即所谓的CIA(Confidentiality, Integrity, Availability)三个方面。

机密性专指受保护数据只可以被合法的（或预期的）用户可访问，其主要实现手段包括数据的访问控制、数据加密和密钥管理等手段;

完整性是保证只有合法的（或预期的）用户才能修改数据，主要通过访问控制来实现，同时在数据的传输和存储中可以通过校验算法来保证用户数据的完整性;

可用性主要体现在整体的安全能力、容灾能力、可靠度，以及各个相关系统（存储系统、网络通路、身份验证机制和权限校验机制等等）的正常工作保障。

事实上，数据库经过长期发展已经构建出体系化的安全

能力，同时许多专业化的评估测试机构也在帮助数据库厂商挖掘产品缺陷，加速完善数据库安全能力的构建，并出具专业化评估报告，作为第三方背书让用户“信得过”。这些成熟的安全技术手段，构建了数据库纵深防御的安全体系，保障数据库在应用中的安全。

由于云数据库服务的应用便捷性、高可靠、低成本等优势可降低企业运营成本，加速企业应用创新，云数据库已成为数据库业务未来重要的增长点。但无论是传统的线下数据库服务，还是日益增长的云数据库服务，数据库的核心任务都是帮助用户存储和管理数据，且在复杂多样的环境下保证数据不丢失、隐私不泄露、数据不被篡改以及服务不中断。这就要求面向开放市场的云数据库具备多层次的安全防御机制，以面对相较于传统数据库更加多样化、复杂化的风险。应用程序漏洞、系统配置错误还是恶意管理员都可能对数据安全与隐私保护造成巨大风险。云数据库的部署网络由“私有环境”向“开放环境”转变，系统运维管理角色被拆分为业务管理员和运维管理员。业务管理员拥有业务管理的权限，属于企业业务方，而运维管理员属于云服务提供商。数据库运维管理员虽然被定义成系统运维管理，其实际依旧享有对数据的完全使用权限，可通过运维管理权限或提权来访问数据甚至篡改数据。同时，由于开放式的环境和网络边界的模糊化，用户数据在整个业务流程中被更充分地暴露给攻击者，无论是传输、存储、运维还是运行态，都有可能遭受来自攻击者的攻击。

面对越来越复杂的云环境，我们需要一种能够彻底解决数据全生命周期隐私保护的系统性解决方案。事实上，近年来学术界以及工业界陆续提出了许多创新思路。其一是数据离开客户端时，在用户侧对数据进行加密，且不影响服务端的检索与计算，从而实现敏感数据保护。此时即便数据库管理员也无法接触到用户侧的密钥，进而无法获取明文数据。这一思路被称为密态数据库解决方案，或全加密数据库解决方案。

正如密态数据库定义所描述的那样，密态数据库的核心任务是保护数据全生命周期安全并实现基于密文数据的检索计算。在加密算法足够安全的情况下，外部攻击者及内部管理员均无法获取有效的数据信息。

二、主流安全技术介绍

目前，主流数据库安全技术包括权限与访问控制、全链路加密、数据脱敏、数据库审计、数据库漏洞扫描和数据库防火墙。目前国内分布式数据库产品基本具备这些安全特性。

权限与访问控制是利用授权和鉴权的方法来控制数据库用户对授权数据的访问。在权限分配的过程中，应遵循最小权限原则，实现细粒度的访问控制，从而提升数据资源的完整性和资源访问的安全性，实现访问控制过程的动态管理。

“全链路”指的是数据在传输、计算，存储的过程，而“全链路加密”指的是端到端的数据加密保护能力，即从应用系统到数据库的传输过程、到数据在应用运行时的计算过程（使用/交换），和到数据最终被持久化落盘的存储过程

中的加密能力。

数据脱敏是一种采用专门的脱敏算法对敏感数据进行变形、屏蔽、替换、随机化、加密，并将敏感数据转化为虚构数据的技术。按照作用位置、实现原理不同，数据脱敏可以划分为静态数据脱敏（Static Data Masking, SDM）和动态数据脱敏（Dynamic Data Masking, DDM）。

数据库审计能够实时记录数据库的活动，对数据库操作进行细粒度审计。除此之外，数据库审计还应能对数据库遭受到的风险行为进行告警，如：数据库漏洞攻击、SQL 注入攻击、高危风险操作等。数据库审计信息生成可以由数据库内核提供，也可通过旁路部署，通过镜像流量或探针的方式采集流量，并基于语法语义的解析技术提取出 SQL 中相关的要素（用户、SQL 操作、表、字段等）进而实时记录来自各个层面的所有数据库活动。

数据库漏洞扫描是专门对数据库系统进行自动化安全评估的专业技术，通过数据库漏洞扫描能够有效的评估数据库系统的安全漏洞和威胁并提供修复建议。

数据库防火墙系统是针对应用侧异常数据访问的数据库安全策略。一般采用主动防御机制，通过学习期行为建模，预定义风险策略；并结合数据库虚拟补丁、注入规则和应用关联防护机制，实现数据库的访问行为控制、高危风险阻断和可疑行为审计。

（一）访问控制

访问控制，就是通过某种途径显式地允许或者限制访问

能力及范围，以限制对关键资源的访问，防止非法用户的侵入或合法用户的不慎操作所造成的破坏。采用可靠的访问控制机制是数据库系统安全的必要保证。目前主流的访问控制技术包括自主访问控制，强制访问控制，和基于角色的访问控制。项目所选数据库很好的支持了自主访问控制、基于角色的访问控制，以及基于安全标记的细粒度访问控制能力，防止越权访问和信息泄漏。

1. 基于角色的访问控制

随着数据库结构的日益复杂，规模增大，用户增多，传统访问控制技术存在的问题越来越突出，权限的分配和管理很困难。如图 1 所示，基于角色的访问控制应运而生。其基本思想是授权和角色相联系，拥有某角色的用户可以获得该角色对应的权限。角色可以根据组合中不同的工作创建，再根据用户的职责和资格来分配。用户可以轻松的进行角色转换。

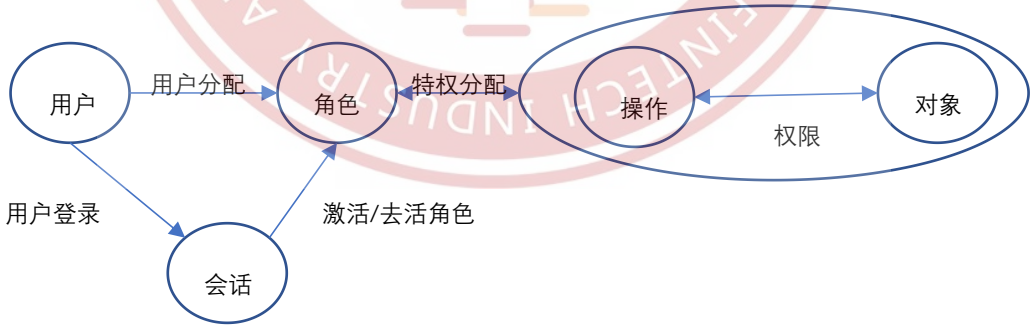

图 1 基于角色的访问控制

2. 基于权限的访问控制

包括五个基本的静态集合：用户集，角色集，对象集，操作集和权限集。如图一，用户集是可以执行操作的用户；对象集是被动实体，包括系统中需要保护的信息；操作集是

定义在对象上的一组操作；对象上的一组操作构成了一个权限集；角色集是基于权限的访问控制的核心，是角色成员授予的职责和责任，通过用户分配和权限分配将用户和特权关联起来。这样，用户的大幅增加不会导致权限管理强度的大幅提高，只需根据工作需要设立和调整角色及其拥有的权限，并将用户加入相应的角色即可。分布式数据库系统应独立支持基于角色的访问控制方式，简化权限控制的复杂度。

3. 安全标记及强制访问控制

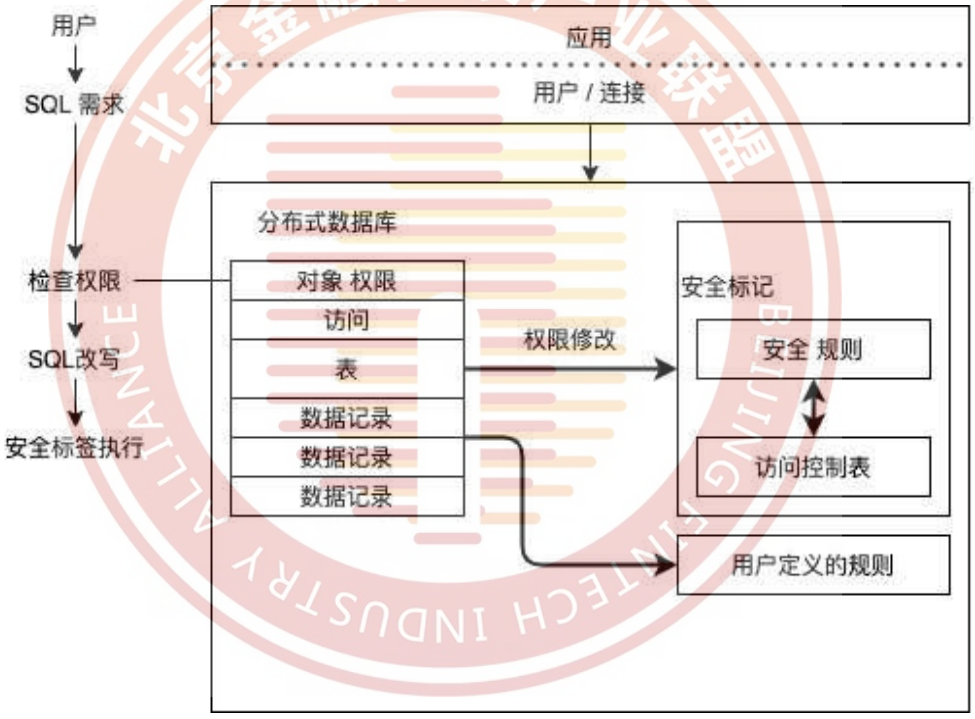

图 2 行安全标记的处理流程

不论是自主访问控制，还是基于角色的访问控制，都是通过授予主体对客体的访问权限，进行访问控制，通常客体是一个对象。但是，有时候这种访问控制无法满足更高层次的安全需求。例如，对一个对象的某一部分的访问控制。因此，开发实现基于行标签的安全标记方法是十分必要的，如

图 2 所示，利用安全标记控制某用户只有对某表内的某些行具有访问限，无法访问其他行数据，实现了更高级别的数据权限管理。

4. 权力分立

为了避免管理员权限的过于集中，数据库系统应能够更细致的对权限进行细分。参照行政、立法、司法三权分立的原则，可以设立数据库的三权分立如图 3 所示。区分管理员角色分别为系统管理员，安全管理员和审计管理员。分别完成数据库系统管理，安全规则/审计规则的创建和管理，以及数据库审计等功能。权限角色基于租户内置，不可删，不可改，避免系统管理员权限过于集中。

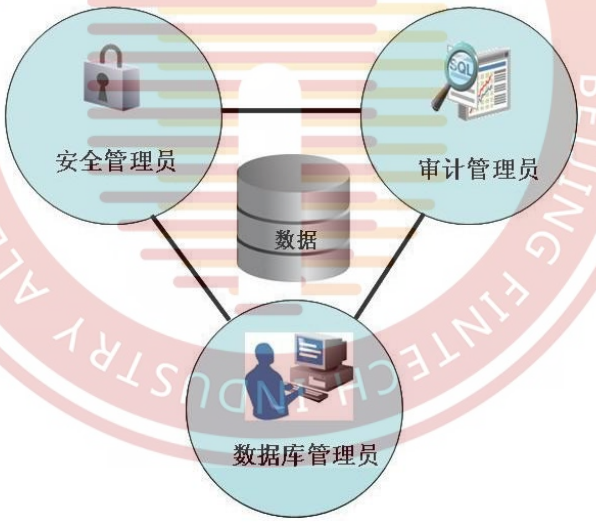

图 3 权力分立

另外，为了强化用户管理，符合信息系统等级保护的要求。数据库系统应具有对用户密码的复杂度控制、失效控制，以及对用户身份检查能力进行优化，防止恶意的密码攻击，进一步提升数据库的安全性。

（二）全链路加密

1.传输加密

典型的分布式关系型数据库，从产品架构层面可以分为以下三个基本部分，如图 4 所示:

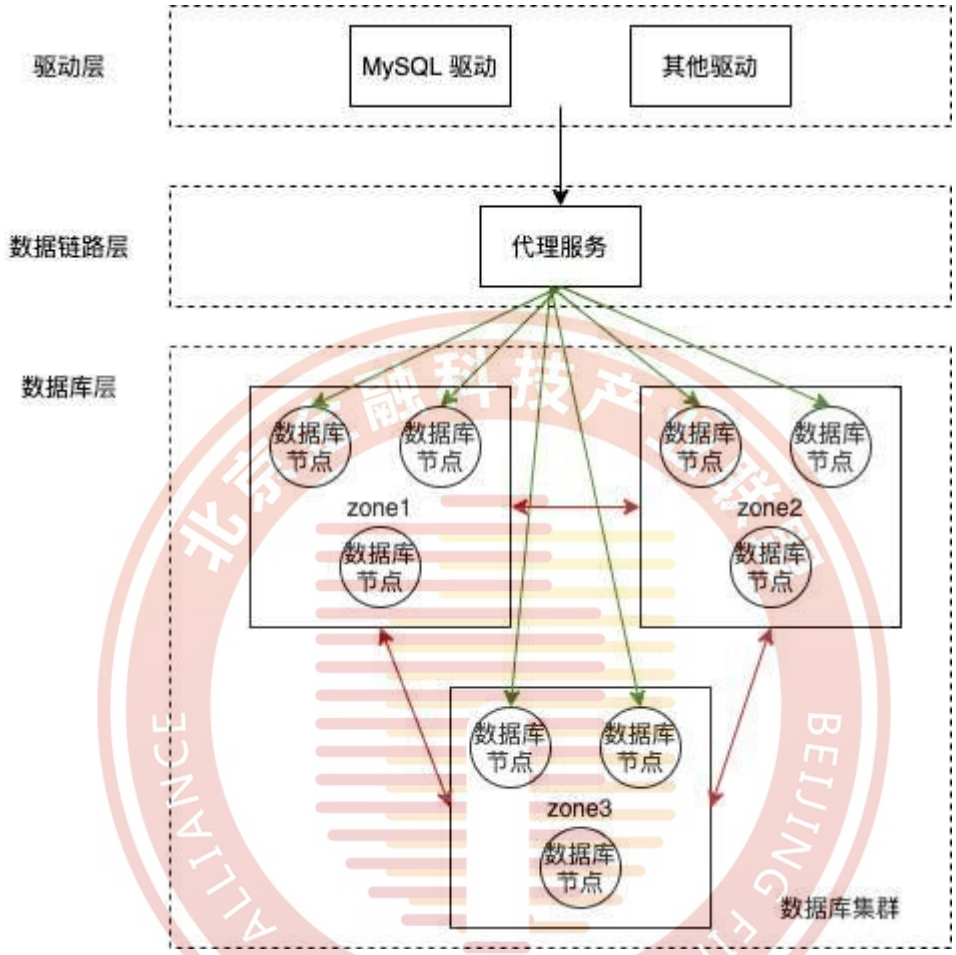

图 4 典型分布式数据库产品架构

数据库层：由若干个分布式数据库服务节点组成。包含 SQL 引擎、存储引擎和分布式事务引擎，共同配合提供一个高可用、可扩展的分布式关系型数据库系统。

数据链路层：提供从用户端到数据库端的最佳链路访问功能，屏蔽用户对分布式数据库的感知，保障分布式数据库的最高性能服务。

驱动层：直接面向用户和应用程序，提供访问分布式数据库的通信能力。主要组件应包括数据库客户端、jdbc 驱

动、odbc 驱动等。

加密方法。客户端和数据库端的通信协议一般采用 SSL/TLS 协议，该方面本身相对成熟。部分国内分布式数据库产品支持在不同租户里每个用户可采用不同的 SSL 认证机制，包括:SSL 单向认证，X509 双向认证，以及特殊的双向认证，如指定加密算法认证，指定发行方认证、指定 SSL 主题认证等。用户可以通过 MySQL 兼容的语法进行设置。

秘钥管理。数据库层的数据库服务节点、数据链路层和驱动层都需要使用 SSL 依赖的秘钥文件(私钥/公钥/证书)。数据库层和数据链路层的秘钥管理提供三种方式:

秘钥文件本地存放。密钥产生：数据库层面不负责 CA 颁发机构的建设，依赖外部认证的 CA 颁发中心。密钥存放:密钥被明文存储在数据库节点服务器端/代理端程序所在目录，服务器端/代理端启动时读取秘钥文件开启 SSL。密钥更新：人工将新的秘钥文件覆盖原文件进行更新。密钥生效:重启单个数据库节点服务器端/代理端进程后，即刻生效。整个分布式数据库集群生效需要全部数据库服务节点重启。风险：明文本地存储，容易泄露和篡改。

秘钥文件系统表存放。密钥产生：数据库层面不负责 CA 颁发机构的建设，依赖外部认证的 CA 颁发中心。密钥存放:密钥文件内容以字符串密文存储在数据库系统表中，使用对称加密方式，对称加密的算法和密码写在代码里不透明。系统表中的秘钥文件从数据库下载到本地配置文件中，数据库节点服务器端/代理端启动时读取配置文件中秘钥的密文，

内部解密后使用。密钥更新：使用系统租户更新秘钥文件对应的三个配置项为新的秘钥字符串。数据库系统表更新机制会保证秘钥更新同步到各个数据库节点中。密钥生效：重启单个数据库节点服务器端/代理端进程，单个数据库节点服务器端/代理端即刻生效。整个集群生效需要全部数据库节点服务器端重启。风险：如果分布式数据库端源代码泄露，可以破解出配置文件中加密的秘钥文件。系统租户泄露，会导致秘钥被触发更新为错误秘钥，数据库节点服务器重启SSL 异常会影响很多其他用户。

密钥文件统一管理服务。密钥产生：数据库层面不负责CA 颁发机构的建设，依赖密钥文件统一管理服务生成和颁发。密钥存放：秘钥文件存在统一管理的服务器中，数据库节点服务器端/代理端启动时向密钥服务器拉取三个文件并使用，用完就丢弃，不会保存在内存和系统表中。向密钥服务器索取所用的证书/私钥/私钥口令明文存在分布式数据库系统表中，并下载到本地配置文件中。密钥更新：使用系统租户手动执行 DDL 语句触发密钥服务器更新存储的秘钥文件，数据库节点服务器端/代理端再次启动时向密钥服务器拉取最新的秘钥文件并生效使用，运行时不会拉取秘钥文件。密钥生效：重启单个数据库节点服务器端/代理端进程，单个数据库节点服务器端/代理端即刻生效。整个集群生效需要全部数据库节点服务器重启。风险：向密钥服务器索取所用的证书/私钥/私钥口令明文存在分布式数据库系统表和本地配置文件中，存在泄露风险。进而当前使用的秘钥文件泄露，

即使加密存放，同样存在对称加密密码泄露的可能。如果数据库节点服务器启动时，密钥服务器无法访问，数据库节点将拉取不到最新的秘钥文件，此时数据库节点会重启失败。

2.存储加密

目前，不同场景下仍在使用的数据库加密技术主要有：应用系统加密、前置代理加密、后置代理加密、表空间加密、文件系统加密和磁盘加密。

应用系统加密（见图 5）。应用系统加密技术被认为是最早的数据库加密形式。但严格来讲，应用系统加密实际上是针对数据而非数据库进行的加密。在应用系统层的源代码中对敏感数据进行加密，加密后将密文存储到数据库中。可以直接在应用系统的源代码中以独立的函数或模块形式完成加密；也可以通过源代码的方式封装出应用系统相关业务专用的加密组件或定制的加密 API 来完成加密。通常情况下，当业务系统仅对有限的敏感数据存在加密需求时，可以考虑使用应用系统加密技术。这里的“有限”包含两方面含义：一方面，是需要加密处理的敏感数据对应的表或字段相对较少；另一方面，是需要加密处理的敏感数据在整个业务系统中的使用相对不多。比如，仅对业务系统中与员工薪资相关的敏感数据进行加密保护。在实际业务中，薪资信息作为员工信息的一个子部分，在数据库中通常以独立表的形式存在，与员工基础信息、教育信息、履历信息等共同构成员工信息子系统；同时，薪资信息通常只在员工薪酬模块或子系统中使用，与其它业务模块相关性不高，一般也不会在其它业务

模块中被引用。

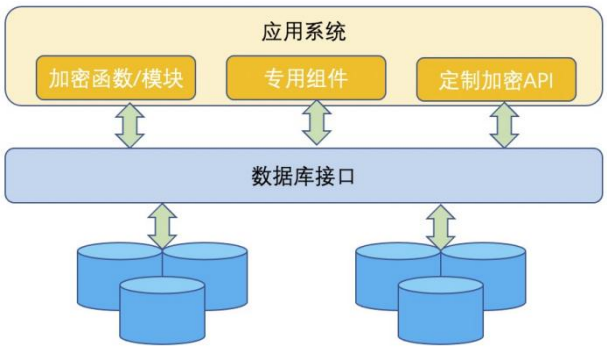

图 5 应用系统加密

前置代理加密（见图 6）。前置代理加密技术是在应用系统加密技术基础上发展起来的，其表现形式通常是由专业的数据安全厂商推出的数据库加密产品。类似于应用系统加密技术，前置代理加密技术也是在数据保存到数据库之前对敏感数据进行加密，并将密文存储到数据库中；而不同于前者的是，前置代理加密技术通常是以“前置代理加密网关”这种独立组件产品的形式实现的。通常情况下，当业务系统仅对有限的敏感数据存在加密需求，且用户自身无能力或不愿意进行加解密的相关研发工作时，可以考虑使用前置代理加密技术，即采用第三方厂商的前置代理加密网关系统对敏感数据进行加密保护。

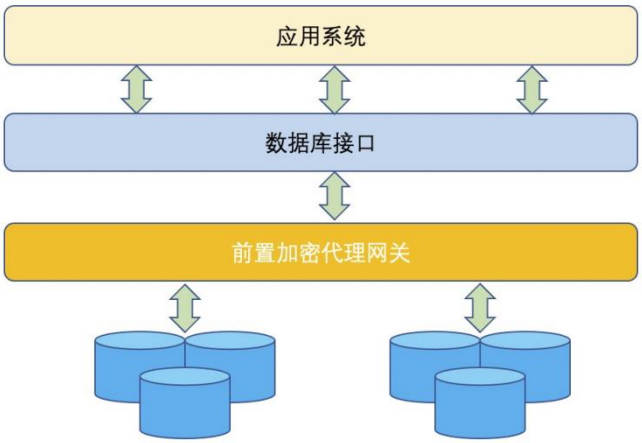

图 6 前置代理加密

后置代理加密（见图 7）。为了避免数据加密给数据访问和处理带来性能上的严重损失，部分数据库厂商在数据库引擎层提供了一些扩展接口和扩展机制。通过这些扩展的接口和机制，数据库系统用户可以通过外部接口调用的方式实现对数据的加解密处理，同时也能够在一定程度上降低对数据库系统性能的影响。后置代理加密技术是基于数据库自身能力的一种加密技术，可充分利用数据库自身提供的定制扩展能力实现数据的存储加密、加密后数据检索和应用透明等目标。Oracle 数据库可以通过“视图+触发器+扩展索引+外部方法调用”的方式实现数据加密，同时保证应用的完全透明。

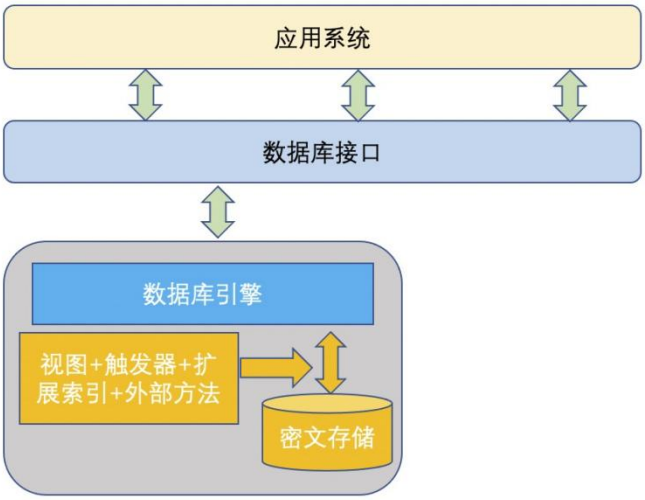

图 7 后置代理加密

透明数据加密（见图 8）。后置代理加密过于依赖数据库自身所具备的扩展机制，且数据在数据库共享内存中也是密文，导致在部分场景下的数据库性能表现不佳。因此，基于后置代理加密技术又发展出了透明数据加密技术，目的是在保持后置代理加密优势的同时，降低对数据库自身扩展机制的依赖性，从而让数据库系统性能保持在相对合理的水平之上。透明数据加密，全称为 Transparent Data Encryption（TDE），是一种对应用系统完全透明的数据库端存储加密技术，通常由数据库厂商在数据库引擎中实现——在数据库引擎的存储管理层增加一个数据处理过程，当数据由数据库共享内存写入到数据文件时对其进行加密；当数据由数据文件读取到数据库共享内存时对其进行解密。也就是说，数据在数据库共享内存中是以明文形态存在的，而在数据文件中则以密文形态存在。同时，由于该技术的透明性，任何合法且有权限的数据库用户都可以访问和处理加密表中的数据。透明数据加密技术由于其自身的优势特性，使其适用于几乎

全部有数据库加密需求的应用场景，尤其是在对数据加密透明化有要求，或需要对数据库超级用户进行数据访问权限控制，以及对数据加密后数据库性能有较高要求的场景中。

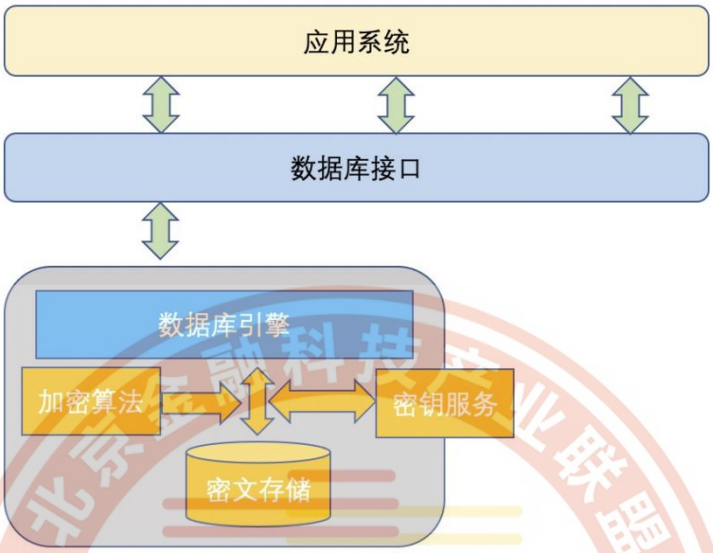

图 8 透明加密

文件系统加密。在数据库加密技术中，除了从前端应用及数据库自身角度实现数据库加密外，基于数据库底层依赖的文件系统或存储硬件，也可以实现数据库加密。文件系统加密技术是在操作系统的文件管理子系统层面上对文件进行加密，大多是通过对与文件管理子系统相关的操作系统内核驱动程序进行改造实现的。不同于文件加密只对单个文件设置访问口令，或对单个文件的内容进行加密转换，文件系统加密提供了一种加密文件系统格式（类似于 ext4、xfs 等文件系统格式），通过把磁盘存储卷或其上的目录设置为该文件加密系统格式，达到对存储于卷或卷上目录中文件进行加密的目的。文件系统加密技术本质上并不是数据库加密技术，但可以用于对数据库的数据文件进行存储层面的加密。

文件系统加密技术几乎可以适用于任何基于文件系统的数据库存储加密需求，尤其是原生不支持透明数据加密的数据库系统和大数据数据库系统。但是，由于文件系统加密技术无法提供针对数据库用户的增强权限控制，对于需要防范内部数据库超级用户的场景并不适用。

磁盘加密。磁盘加密技术通过对磁盘进行加密以保障其内部数据的安全性，从实现上有软硬两种方式：软件方式的磁盘加密技术，大多是通过专用的磁盘加密软件对磁盘内容进行加密，典型代表如 Windows 操作系统自带的 BitLocker，同类型的商业软件在国内也有很多，但这类软件由于加密原理和使用方式等因素，基本上无法满足数据库系统的数据加密需求；而硬件方式的磁盘加密技术，在实现上则有两个思路：一种是针对单块硬盘的磁盘加密，一种是针对磁盘阵列或 SAN 存储设备的磁盘加密。

（三）数据脱敏

1.数据脱敏介绍

数据脱敏也叫数据的去隐私化，在我们给定脱敏规则和策略的情况下，对敏感数据比如 手机号、银行卡号 等信息，进行转换或者修改的一种技术手段，防止敏感数据直接在不可靠的环境下使用。像政府、医疗行业、金融机构、移动运营商是比较早开始应用数据脱敏的，因为他们所掌握的都是用户最核心的私密数据，如果泄露后果是不可估量的。数据脱敏的应用在生活中是比较常见的，比如我们在淘宝买东西订单详情中，商家账户信息会被用 * 遮挡，保障了商户隐

私不泄露，这就是一种数据脱敏方式。

数据脱敏又分为静态数据脱敏（SDM）和动态数据脱敏（DDM）：

静态数据脱敏（SDM）：适用于将数据抽取出生产环境脱敏后分发至测试、开发、培训、数据分析等场景。

有时我们可能需要将生产环境的数据 copy 到测试、开发库中，以此来排查问题或进行数据分析，但出于安全考虑又不能将敏感数据存储于非生产环境，此时就要把敏感数据从生产环境脱敏完毕之后再在非生产环境使用。这样脱敏后的数据与生产环境隔离，满足业务需要的同时又保障了生产数据的安全。

如图 9 所示，将用户的真实 姓名、手机号、身份证、银行卡号 通过替换、无效化、乱序、对称加密等方案进行脱敏改造。

图 9 脱敏系统

动态数据脱敏（DDM）：一般用在生产环境，访问敏感数据时实时进行脱敏，因为有时在不同情况下对于同一敏感数据的读取，需要做不同级别的脱敏处理，例如：不同角色、不同权限所执行的脱敏方案会不同。

在抹去数据中的敏感内容同时，也需要保持原有的数据

特征、业务规则和数据关联性，保证我们在开发、测试以及数据分析类业务不会受到脱敏的影响，使脱敏前后的数据一致性和有效性。脱敏的原则，不影响数据的使用。

2.数据脱敏方案

数据脱敏系统的架构如图 10 所示。系统可以按照不同业务场景自行定义和编写脱敏规则，可以针对库表的某个敏感字段，进行数据的不落地脱敏。

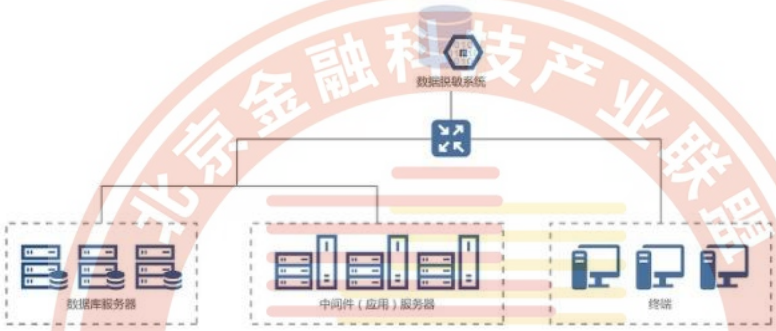

图 10 数据脱敏系统

数据脱敏的方式有很多种:

无效化。无效化方案在处理待脱敏的数据时，通过对字段数据值进行 截断、加密、隐藏 等方式让敏感数据脱敏，使其不再具有利用价值。一般采用特殊字符（*等）代替真值，这种隐藏敏感数据的方法简单，但缺点是用户无法得知原数据的格式，如果想要获取完整信息，要让用户授权查询。

随机值。随机值替换，字母变为随机字母，数字变为随机数字，文字随机替换文字的方式来改变敏感数据，这种方案的优点在于可以在一定程度上保留原有数据的格式，往往这种方法用户不易察觉的。

数据替换。数据替换与前边的无效化方式比较相似，不同的是这里不以特殊字符进行遮挡，而是用一个设定的虚拟

值替换真值。比如说我们将手机号统一设置成“13651300000”。

对称加密。对称加密是一种特殊的可逆脱敏方法，通过加密密钥和算法对敏感数据进行加密，密文格式与原始数据在逻辑规则上一致，通过密钥解密可以恢复原始数据，要注意的就是密钥的安全性。

平均值。平均值方案经常用在统计场景，针对数值型数据，我们先计算它们的均值，然后使脱敏后的值在均值附近随机分布，从而保持数据的总和不变。

偏移和取整。这种方式通过随机移位改变数字数据，偏移取整在保持了数据的安全性的同时保证了范围的大致真实性，比之前几种方案更接近真实数据，在大数据分析场景中意义比较大。

数据脱敏规则在实际应用中往往都是多种方案配合使用，以此来达到更高的安全级别。无论是静态脱敏还是动态脱敏，其最终都是为了防止组织内部对隐私数据的滥用，防止隐私数据在未经脱敏的情况下从组织流出。所以作为一个程序员不泄露数据是最起码的操守。

（四）数据库审计

审计是监视和记录用户对数据库进行的操作，以供审计员进行问题分析。利用审计功能，可以完成以下任务：第一，监视和收集特定数据库活动的数据。例如管理员能够审计哪些表被更新，在某个时间点上有多少个并行用户统计数据；第二，保证用户对自己的活动负责。这些活动包括在特定模

式、特定表、特定行等对象上进行的操作；第三，审计数据库中的可疑活动。如一个未经授权的用户正从表中删除数据，那么数据库管理员必须审计所有数据库连接，以及在数据库中所有成功和失败的删除操作。

1.审计功能

如图 11 所示，根据审计类型不同，审计记录中的信息也有所不同。通常，一条审计记录中包含用户名、会话标识、终端标识、所操作的模式对象名称、执行的操作、执行的完整语句代码、日期和时间戳、所使用的系统权限。

数据库支持的审计类型一般包括:

语句审计（Statement Auditing），对特定的 SQL 语句进行审计，不指定具体对象;

对象审计（Object Auditing），对特定的模式对象上执行的特定语句进行审计;

权限审计（Privilege Auditing），对特定的系统权限使用情况进行审计;

网络审计（Network Auditing），对网络协议错误与网络层内部错误进行审计。

审计还包括两个额外属性，这两个属性相互正交组合:

审计时机。根据用户是否成功执行，可以分为，“对执行成功的语句进行审计”、“对不成功的语句进行审计”，以及“无论成功与否都进行审计(默认)”。

审计频率。根据对同一个语句审计次数不同，可以分为“session 级别“,指对某个用户或所有用户的同一非 DDL 语句只审计一次，形成一条审计记录。“access 级别”，指对某个用户或所有用户的同一语句每执行一次审计一次，形成多条审计记录。

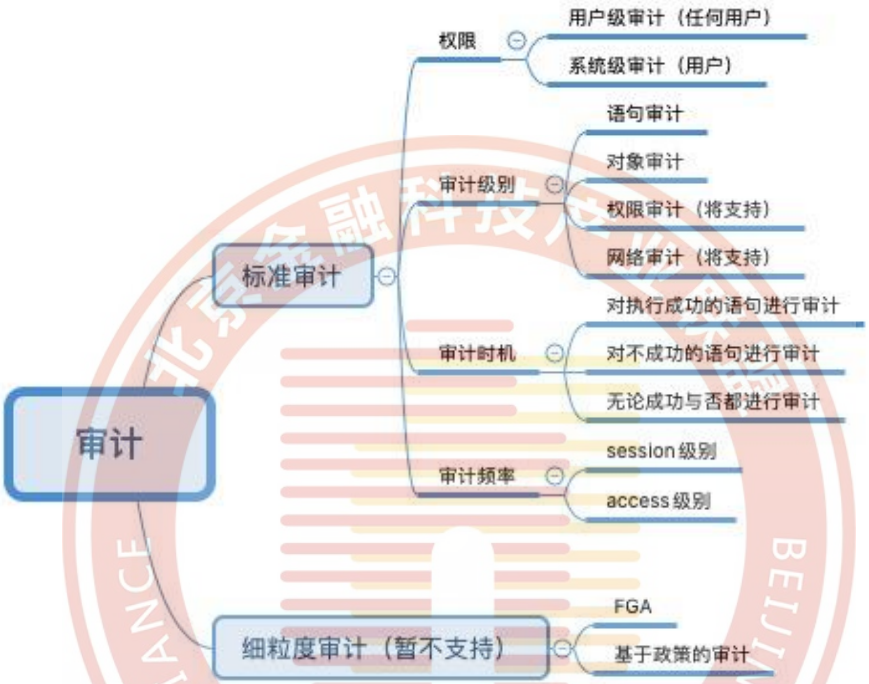

图 11 审计功能定义

2.审计的非功能需求

（1）性能

审计开启后，会在 client 请求执行完毕，准备回包给 client 前，进行审计检查，判断是否需要审计。如果需要审计，则生成审计记录写入文件或者内部表。这个审计过程需要检查是否审计，并可能生成保存审计记录，会对原 query 有一定的性能影响。

如果每次审计都发现不需要审计，则影响比较小，都只是些规则查表状态检查。如果需要审计写文件，审计写文

件时受磁盘 IO 影响。如果需要审计写系统表，影响会稍大。因为审计记录表的 leader server 可能不是本机，需要产生远程执行。执行结束后才能继续用户请求。

以上影响在业界都存在，因为审计不可避免的要进行规则检查，会记录保存，记录保存成功才能将请求结果返回给用户，但性能影响和业界能力可比。

（2）可用性

审计记录如果写文件，受磁盘空间影响，如果磁盘空间不足，需要审计记录写入会失败阻塞，用户请求需要阻塞。审计记录如果写系统表，受租户资源影响，因为审计记录表是租户系统表的一部分，如果租户资源不足，则审计记录表插入动作会失败，用户的当前请求也需要阻塞。

（3）安全性

审计规则只有审计员可以创建。审计记录应只有特定权限的用户才可以查看。租户管理员也无法删除审计记录，只有系统管理员有权限删除。

（4）兼容性

分布式数据库的审计语法与应与主流数据库兼容，并提供兼容的完整视图，尽可能小的对上下游产品产生影响。

（五）数据库防火墙

数据库防火墙是一款基于数据库协议解析与控制技术的数据库安全防护系统，实现了对数据库访问行为的控制，高危操作的拦截，可疑行为的监控，风险威胁的拦截，提供可靠的数据库安全保护服务。数据库防火墙可以提供协议分

析、SQL 语句解析、参数化匹配、长语句解析、多语句解析、应用关联等核心技术，并提供丰富的数据库安全策略，如阻断、拦截等控制类策略，记录、告警等审计类操作。数据库防火墙主动地、实时地、全方面地保障数据库安全，使其免受数据库漏洞、高危、恶意操作以及敏感数据泄露的威胁。

1.数据库防火墙应具备的功能

（1）学习期行为建模

数据库防火墙可基于学习期完成语句、会话的建模分析，构建数据库安全防护模型。并且具备语法分析能力，可以对 SQL 语句进行抽象描述，将海量的 SQL 语句归类成 SQL 模板。基于 SQL 模板关联会话信息，可预定义数据库黑白名单和风险规则。

（2） 数据库入侵行为防护

外部系统利用数据库漏洞进行数据库攻击时，数据库防火墙可以实时捕获到对应的 SQL 语句及相关会话信息，及时阻断风险会话并发送告警，帮助用户实时防护数据库漏洞攻击并有效追溯风险来源。针对 SQL 注入和 XSS 攻击行为数据库防火墙基于精准的 SQL 语法分析，可以准确定位 SQL 语句中的操作谓词及常量表达式，保障注入、攻击行为防护的准确性。

（3）数据库违规行为防护

数据库防火墙提供丰富的规则类型，可以针对不同的数据库访问来源，提供对敏感表的访问权限、操作权限和影响行数的实时有效管控，并结合对 NO WHERE 语句风险的判断，

避免大规模数据泄露和篡改。

（4）数据库异常行为监控

数据库防火墙可对数据库通讯协议进行完全解析，将SQL 语句归类成模板，并结合会话信息、应用关联信息，实现学习期行为建模，并以学习期建立的模型为“蓝本”，对数据库访问行为进行周期性对比，以此绘制行为趋势图，快速定位“波动点”识别可能存在的异常行为并预定义风险规则，帮助用户准确定位异常行为。

（5）阻断与拦截功能

数据库防火墙支持会话阻断，可准确定位风险来源并阻断会话请求。在会话阻断的基础上，提供“语句拦截”的处理机制，仅针对会话里产生风险的 SQL 语句进行拦截，保持会话内其他合规语句的正常操作。

（6）行为审计功能

数据库防火墙从风险、语句和会话的角度提供风险行为的审计功能，用户可以在此基础上进行关联查询，深入挖掘风险来源和风险行为模型，实现数据库风险行为分析和问题追溯。

（7）提供多样化的风险分析报表

数据库防火墙可实现将报表划分为“风险综合分析报告”、“风险防护报告”、“数据库风险分析”、“客户端风险分析”等，帮助安全管理人员更加便捷、深入的剖析数据库运行风险。

2.数据库防火墙技术

（1）SQL 特征抽象

SQL 语句解析是识别 SQL 语句攻击行为的关键。SQL 语法特征技术是在不改变原 SQL 语句的语义的情况下，重写 SQL 语句，有效地捕获 SQL 语句的特征，快速对 SQL 语句进行策略判定，以实现高效处理。

（2）应用关联防护

数据库防火墙采用应用端插件部署方式，基于应用会话捕获“应用账户”及“应用登录 IP”等关联审计信息，并添加到风险规则进行风险行为管控。针对应用端与业务服务器分离的“四层关联系统”，可以有效追溯到应用系统的原始访问者和请求信息，实现精准的业务匹配。

（3）虚拟漏洞补丁

虚拟补丁是数据库防火墙的核心技术之一，通过该技术用户可以针对数据库漏洞攻击行为进行实时的管控，从而保证数据库安全与应用稳定的平衡。虚拟补丁技术是系统通过协议解析技术，捕捉外部系统利用数据库漏洞进行的网络攻击行为并对其进行管控，从而防止通过已知漏洞对数据库的攻击。本系统通过网络解析技术捕获以下访问特征：用户名、对象、操作、应用模块、语句内容，通过内部的漏洞库进行访问行为匹配。

（六）数据库漏洞扫描

数据库漏洞扫描系统，是对数据库系统进行自动化安全评估的数据库安全产品，能够充分扫描出数据库系统的安全漏洞和威胁并提供智能的修复建议，对数据库进行全自动化

的扫描，从而帮助用户保持数据库的安全健康状态，实现“防患于未然”。

三、全密态数据库技术

相较于传统数据库针对数据所处阶段来制定安全保护措施的，如在数据传输阶段使用安全传输协议 SSL/TLS，在数据持久化存储阶段使用透明存储加密，在返回结果阶段使用 RLS(Row Level Security)或者数据脱敏策略。全密态数据库技术的目标是使得用户（在客户端）自行加密后的敏感数据在任何时刻都不以明文形式暴露给服务器，但数据库服务仍然可以在密文数据上支持所有的计算查询、事务等操作，做到数据库内数据的“可用不可见”。通过该技术，我们可以完全杜绝数据库服务及数据拥有者以外的任何角色接触到用户的明文数据，避免数据泄漏在数据库服务器端发生。全密态技术关注数据全生命周期隐私保护的系统性解决，但缺少标准化功能命名，且与较受关注的隐私计算技术范畴存在交集。就调研情况来看，国内数据库厂商认定的全密态数据库技术范畴主要包括基于硬件的可信执行环境（Trusted Execution Environment, TEE），基于密码学的密态计算等技术手段。国内数据库厂商有阿里云的 RDS PostgreSQL 和 PolarDB、华为 openGauss 提供类似功能。

（一）硬件全密态技术

全密态数据库中的软件方案和硬件方案目前均已取得了很多进展，特别的，工业界已开始在逐步采用硬件方案。硬件方案主要依赖一类可信执行环境技术（Trusted

Execution Environments，TEE）。该类技术通过在服务器上构建一个隔离且安全的容器环境 Enclave，保证 Enclave 内计算和数据的机密性，从而可以安全地对密文进行解密后直接在明文上进行计算。常见的 TEE 环境包括 Intel SGX，ARM TrustZone，RISC-V Keystone 等，其中以 Intel SGX 产品成熟度和普及度最高。这类技术的最大优势是可以支持任意计算，对数据库应用的兼容性高，且计算性能相对较好。但硬件方案目前存在两个较大的缺陷。首先由于数据在 TEE 内部均为明文存在，因此数据的安全性完全依赖于硬件本身的安全性。目前针对硬件的攻击方式如侧信道攻击等越来越多，但是一般硬件设备更新迭代周期较长，一旦出现漏洞无法及时更新修补，将直接导致用户数据长时间暴露在风险之下。其次用户在使用该特性时，密钥需要离开客户端环境发送给 TEE 使用，而该传输过程的安全直接依赖于硬件设备厂商的证书签名。恶意的硬件设备厂商人员完全有能力攻击并窃取用户的数据及密钥，因此硬件方案也需要用户在使用过程中持续信任硬件设备厂商。

（二）软件全密态技术

全密态数据库的软件方案目前在学术界发展较快，即通过一系列数学算法在密文空间直接对密文进行查询运算，保障数据隐私不泄露。软件方案可以不依赖于硬件能力，也不需要在服务侧获取密钥对数据进行解密，但同时也存在着巨大挑战。软件方案主要依赖一类可直接查询和操作密文的密码学算法。这类密码学算法常见的包括：同态加密

（Homomorphic Encryption），可以支持密文上的加减乘除等数值及逻辑运算；保属性加密（Property Preserving Encryption），支持加密后密文与明文具备相同的数据属性；保序加密（Order Preserving Encryption），可以支持密文上的数值比较操作；可检索加密（Searchable Encryption），可以支持密文数据集上的关键字查询操作。从上述算法中我们可以看出，不同的操作需要使用对应的加密算法，当密文数据存于数据库中并需要同时参与不同类型的计算时，无法满足查询需求。性能方面，这些加密算法的计算量一般较大，相比明文上的计算会带来几个量级的性能下降。安全性方面，不同加密算法基于的攻击模型和保护面也各不相同，除同态加密以外的多数加密算法仍会在使用数据过程中泄露部分数据信息。

（三）其他相关技术

1.软硬结合自适应技术

软硬结合自适应技术可以结合软件模式与硬件模式各自的优缺点，推出融合策略，实现硬件模式和软件模式的自由切换，能够支持全场景应用，适应公有云、混合云以及终端智慧业务等不同的业务场景。

2.用户操作无感知技术

由于全加密数据库要求客户端的明文数据在出域前已经被加密，这与传统的明文数据库的使用行为是完全不同的。如果由用户对数据在语句中进行显示加密，会增加用户后续的开发、迁移和升级成本，另一方面也存在因失误而造成敏

感数据未加密风险。

用户操作无感知技术，通过将这一系列的复杂加密操作，全部封装在客户端 SQL 驱动内部，实现完全自动化的敏感信息加密，同时在数据库中存储了所有加密相关的元信息，数据库可以很好的识别和处理对应的加密数据。

3.硬件解耦技术

由于基于 TEE 环境的机密计算,依赖于具体的硬件能力，而各硬件厂商的硬件能力不同，会导致数据库的安全表现不一致。通过硬件解耦技术，全密态数据库能够自适应地支持底层不同的 TEE 技术，并且屏蔽不同 TEE 技术在安全性、使用方式、性能上的差异点，同时尽可能发挥特定 TEE 技术的优势，进一步提升系统性能。

4.防侧信道攻击技术

端到端全程密态处理并不代表绝对的数据安全，机密环境 TEE 在访问数据库数据密文的过程中，会存在元数据信息泄露，访问模式泄露，返回结果大小泄露等问题。通过不经意随机访问、数据容量保护等技术，可以消减这类风险，进一步强化数据库系统的安全性，但会带来一定的性能和存储开销。

5. 可信代码拆分技术

在硬件方案中，其数据的安全性依赖于整个可信基（Trusted Computing Base，TCB），包括 TEE 硬件以及其中加载的代码。也就是说，在 TEE 中加载的代码逻辑是需要被无条件信任的，一旦可信代码中出现漏洞，将会对整个系

统的安全性造成严重的破坏。可信代码拆分技术对数据库系统代码进行合理拆分，最小化需要被加载进 TCB 中的可信代码规模，从而降低可信代码中出现漏洞的可能性；同时，需要避免可信环境和非可信环境的频繁切换回带来的严重性能下降。

四、产品现状及应用场景

（一）产品现状

目前业界有三家云数据库产品推出类似全密态数据库功能。

1. RDS PostgreSQL 和 PolarDB

阿里云的 RDS PostgreSQL 和 PolarDB 提供全密态数据库功能，使得数据在用户侧加密后传入数据库，在数据库上数据全程以密文形式存在，避免平台软件及管理人员接触到明文数据，同时仍然支持所有的数据库事务、查询、分析等操作，做到了数据库内数据的可用不可见。该功能具备软硬件一体化能力：1）使用 Intel SGX 及阿里自研密态计算卡（神盾卡）作为其可选可信执行环境，提供数值计算、字符串操作、范围查询、多表连接等通用密态计算功能；2）使用纯密码算法在无 TEE 环境下支持等值查询、多表连接等少数密态计算功能。阿里云的全密态数据库支持透明无感知加解密 EncJDBC 驱动，提供公共云、混合云以及 DBStack 等产品输出形态。

2. Azure SQL

微软 Azure SQL 提供始终加密功能（always encrypted），

旨在保护存储在 Azure SQL 数据库或 SQL Server 数据库中的敏感数据，例如信用卡号或国家标识号（例如，美国社会保险号）。始终加密允许客户端对客户端应用程序内的敏感数据进行加密，而从不向数据库引擎（SQL 数据库或 SQL Server）公开加密密钥。因此，“始终加密”将拥有数据并可以查看数据的用户与管理数据但不应具有访问权限的用户区分开。通过确保本地数据库管理员，云数据库操作员或其他高特权未授权用户无法访问加密数据，始终加密功能可以使客户可以放心地存储敏感数据，而不受其直接控制。

3.GaussDB

华为的 GaussDB 提供全密态数据库功能，结合软件模式与硬件模式各自的优缺点，推出融合策略，实现硬件模式和软件模式的自由切换，该方案支持全场景应用，包括公有云、混合云以及终端智慧业务，实现用户透明无感知。支持多硬件平台能力，如 Intel CPU 的 SGX，以及华为自主研发鲲鹏 ARM TrustZone 能力。并支持软件模式的密态查询能力，通过对多种密码学算法的深度性能优化，构建出不同的密态查询引擎，以完成不同的检索和计算功能，实现数据等值查询、范围查询、保序查询等特性。

（二）应用场景

全密态数据库技术在金融业务场景中的应用尚处于探索阶段。目前能带来显著安全水位提高的典型应用场景包括:

1.敏感数据数据库安全

针对数据库系统中业务敏感数据的高安全性数据管理。

业务敏感数据是指被泄露后将对国家、行业、企业、个人等造成严重损害的数据，比如企业金融相关数据、个人财务相关数据等。由于这些敏感数据也需要与其他数据一样，在应用系统中被使用，其安全性严重依赖于承载这些数据的数据库和服务器，以及管理开发这些应用的技术人员。使用全密态数据库后，可以消除系统研发人员、数据库运维人员等非业务直接相关人员在管理数据库时可能发生的越权访问、拖库等行为，大幅降低业务敏感数据的被攻击面。

2.敏感数据流转安全

解决敏感数据在不同数据库之间流转带来的安全水位不一致的问题。由于实际业务的复杂性，数据需要被不同的应用单元生产、使用及消费。通常情况下这些应用单元会由不同的业务团队负责，底下接入了独立的数据库实例。当前应用单元的数据库实例之间以明文形式交换数据，因此其数据的安全性由各自的数据库系统环境分别保障。最低安全水位的系统决定了整个应用体系中数据安全强度。使用全密态数据库后，可以使不同数据库系统间以密文形式交换数据，密钥由原始数据持有方独立持有，其安全性不受经流转的数据库系统影响。

3.数据联合使用安全

避免数据在跨组织联合使用过程中的泄露风险，实现高价值数据的安全有效利用。在联合风控、多头借贷风险分析等场景下，通常需要借助来自不同数据源的数据进行更有效的判断和决策。在此过程中，数据提供方担心其高价值数据

会被数查询方窃取，而数据查询方也同样担心自己的查询行为会向数据提供方泄露敏感信息。使用全密态数据库后，数据查询方可以独立在密文库中查询获得有效信息，且无法获得被查询信息以外的任何额外信息，保障双方数据与查询的安全。

除上述场景外，随着全密态数据库在金融行业应用的不断尝试探索，一定会有更多高价值、高创新的新型应用场景和业务模式的产生。

五、应用挑战与应对措施

从技术角度来看，用户从已有数据库服务切换成全密态数据库或者直接将应用部署于全密态数据库，也需解决多个主要技术问题：

（一）保障密态计算机制的安全性

全密态数据库从原理上可以有效保障数据安全，但这要求密文数据处理算法、可信执行环境等在机理和工程上要达到该安全水位要求。特别是，在可信执行环境可能存在漏洞和侧信道攻击的情况下，如何保证数据的安全。可行的应对方式是对全密态数据库的系统架构进行合理的设计与模块化，避免其与具体的密码学算法、可信执行环境强耦合，可以根据应用场景的安全要求自由切换满足条件的算法与环境。

（二）实现业务的无缝迁移或轻量化迁移

一方面，全密态数据库最显著的特征是数据存储信息的变更，与加密数据相关的各类参数都要同步进行变更，否则

会因为计算数据形态的不对等导致查询紊乱。另一方面，目前市面上已有的全密态数据库产品有限，其在数据库协议、查询操作支持上存在一定的功能限制，如何使复杂的存量应用在迁移过程中免受功能限制的影响是一大挑战。可行的应对方式是结合业务场景探索开发有效的数据迁移、应用灰度、无感知加解密等一系列面向应用开发者的生态工具。

（三）避免服务切换所带来的性能损耗

由于密码学和可信硬件等技术都会带来计算、传输、存储等方面的额外性能损耗，本质上需要将算法实现和工程实现所产生的性能回退控制在一个合理的范围内，避免因为不合理的数据加解密和数据存储膨胀带来性能急速下降，影响到在线及离线应用系统的实时性和投入成本。可行的应对方式包括从全密态数据库侧不断提升算法和工程的成熟度，持续降低性能回退程度；从应用系统侧评估安全性和性能的合理平衡，选择小规模但高敏感的数据子集进行全密态保护提高安全性，同时将大规模但低敏感的数据继续以明文形式处理，最大程度上保留系统性能。

六、研究情况总结

从主流数据库安全技术的调研情况来看，国内分布式数据库产品大多数已具备权限与访问控制、全链路加密、数据脱敏、数据库审计、数据库漏洞扫描和数据库防火墙等安全能力。而目前主要数据库风险，是由于访问控制机制绕过，sql 注入，软件漏洞，安全配置错误等原因，导致了数据泄露。追其根源，是由于数据在未加密或需要解密处理导致，

业界开始逐步探索全密态数据库方向，试图从根本上解决数据库内部导致的敏感数据泄露问题。

从全加密数据库技术的调研情况来看，该技术仍处于初期阶段，在功能完整性、性能、安全性上都亟待提高。

从功能的完整性上看，业界较为成熟的是密态等值查询，语法自动解析，数据隐式转换，硬件全密态查询等能力。语法自动解析通过在客户端增加了一个轻量级解析器，用户输入语法后，客户端解析器进行词法和语法解析，获取到明文值及其位置，而加解密驱动则自动将明文替换为加密后的密文，然后将查询语句发送到服务端。在服务端将执行密文结果返回给客户端后，客户端加解密驱动自动将返回的密文数据进行解密并返回给用户。可以让全密态数据库实现 SQL 统一标准，操作语法与通用数据库保持一致。而数据隐式转换技术，可以屏蔽底层具体的加密方式，保持数据存储的原始类型，让应用程序的业务无需考虑底层数据加密的细节，聚焦在业务逻辑上。等值查询技术则允许业务在数据加密情况，实现等值类的条件操作。硬件全密态查询能力，则通过将数据在可信环境中解密之后，进行一系列明文处理，而在不可信任的环境中处于密文状态。未来从全密态的发展趋势来看，功能上会逐步将范围查询、模糊查询、数学计算等密态功能更加成熟、高效。通过硬件全密态查询，以及软硬结合的查询能力，也需要在一致性、分布式、多模等能力上有更多能力的提升。

从性能上来看，软件全密态方式根据场景的不同，性能

差异会比较大，如同态加密、密态模糊查询等技术，都存在性能或存储的严重劣化，有待一定程度的提升。而硬件及软硬结合全密态，也由于存在数据解密处理，内存数据交换等过程，导致性能会有普遍的劣化，会导致使用者成本有较大提升，需要应用厂商在安全与成本之间权衡。预计未来会以单次普通加密的明文查询性能为目标，越来越逼近。

从安全性上来看，软件全密态也存在各算法上的安全差异性，在频率攻击，背景知识攻击等方面有较大限制。而硬件及软硬结合全密态，由于存在解密过程，对于侧信道攻击，密钥攻击等方面，需要有更强的保护措施。

从未来发展来看，通过数据库产业侧的努力希望能够达到：所有常见的数据库系统均能支持全密态功能、所有明文的数据库操作都能提供对应的全密态操作、应用开发成本与明文数据库无异、性能回退无限趋近于普通数据加密等，使得在任何环境内（如公共云、行业云、本地部署）的数据库均能具备一致的数据机密性保护水位，免除数据拥有者对数据库环境不可信的担忧，促进数据库向中心化、规模化、低成本化方向集中。同时进一步与防篡改、隐私保护、多方安全计算等技术相互结合，保障数据在业务使用中的全方位安全，促进数据的自由流通与有效利用。