数据库电子书

数据库安全审计手册

数据库系统功能强大而丰富，对于一个数据库环境而言，我们可以生成很多类型的审计记录。知道有哪些审计类型以及如何实施这些审计将有助于你满足合规需求。

- *审计数据库的登入和登出*
- *审计使用数据库的源头*
- *审计正常操作时间之外的数据库使用*
- *审计敏感数据的变更*

TechTarget 中国

数据库安全审计手册

前言：数据库系统功能强大而丰富，对于一个数据库环境而言，我们可以生成很多类型的审计记录。知道有哪些审计类型以及如何实施这些审计有助于你满足合规需求。要实施完善的数据库审计，必须理解的一个关键问题是需求，从而知道可以使用哪些审计类型来满足自己的需求。

审计数据库的登入和登出

在你要进入一家公司会晤某人的时候，一般需要在前台进行登记。这样做可确保公司完整的记录进入公司的任何人，在出现问题时，或在追踪和调查“这事儿是谁干的？”时，这种登录很有用处。这种日志通常会记录来客是谁、何时进入、何时离开。同样的过程也适用于数据库，多数环境中所需要的首要审计就是完整的记录哪些人曾登入过数据库。

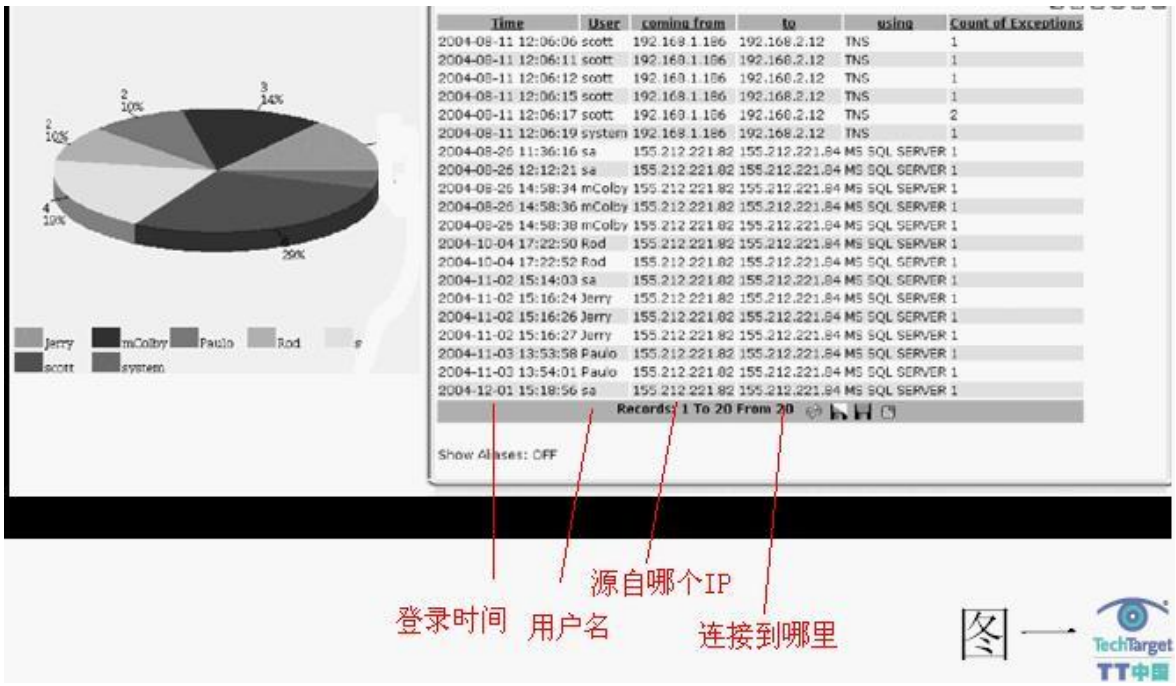
对于这类审计，你需要记录两类事件：登入事件和登出事件。对于每一个事件，你都需要保存登录名和时间，但你还应该考虑记录附加信息。这包括发起连接的客户端的 IP 地址，以及用于初始化连接的程序。例如，在 Oracle 环境中，你可能想知道该连接是否由 SQL Plus 等初始化。

除了这两类事件，你还应当记录所有失败的登录企图。事实上，从安全的观点来看，失败的登录事件甚至可能比成功的登录更重要。记录失败的登录企图不仅是为了审计和合规目的，而且可作为警告和停用某个账户的基础。

虽然你可将这三类事件放在同样的文件或表格中，但你可能会以不同的方式进行报告。成功的登录和登出报告不是多数人愿意关注的问题，除非要进行某种调查，因为这些日志反映了正常的操作。除此之外，还有一种例外情况，即比较不同时期的文件，看其是不是发生了改变。然而，过多的失败登录肯定是一个令

人感兴趣的安全报告，而且许多人会根据如下这几个方面定期地查看这些失败的登录企图：

- 用户名
- 连接失败的客户端 IP 地址
- 源程序
- 时间和日期



例如，图一展示了两个视图，这两个视图根据两个方面进行了分类总结，一是根据登录名（上图的左侧），二是根据显示失败登录的详细视图的报告（包括登录名、源 IP 地址、连接到哪个数据库服务器、通信类型）。

可使用数据库的特性或使用外部的数据库安全方案来审计登入和登出活动。

所有的数据库厂商都支持这种基本的审计功能，而且因为这些事件的数量是很小的（至少在与审计实际的 SQL 调用时所得到的事件数量相比时是这样的），所以数据库在执行这种审计时所遭受的性能损失很少。

正如在插入或更新时，Oracle 触发器会启动一样，系统级触发器会针对特定的系统事件（如登入、登出和 DDL 执行）而执行。下面看一下如何实施这种审计：

首先，创建一个你用来保存这种信息的表：

```
create table user_login_audit
(
   user_id           varchar2(30),
   session_id        number(8),
   host              varchar2(30),
   login_day         date,
   login_time        varchar2(10),
   logout_day        date,
   logout_time       varchar2(10)
);
```

下一步，创建一个在发生新登入时可执行的触发器：

```
create or replace trigger
user_login_audit_trigger
AFTER LOGON ON DATABASE
BEGIN
insert into user_login_audit values(
user,
sys_context('USERENV','SESSIONID'),
sys_context('USERENV','HOST'),
sysdate,
to_char(sysdate, 'hh24:mi:ss'),
null,
null
);
COMMIT;
END;
```

多数数据在用户登入时被记载，但在用户登出时，需要用触发器来记载其登出日期和时间，代码如下：

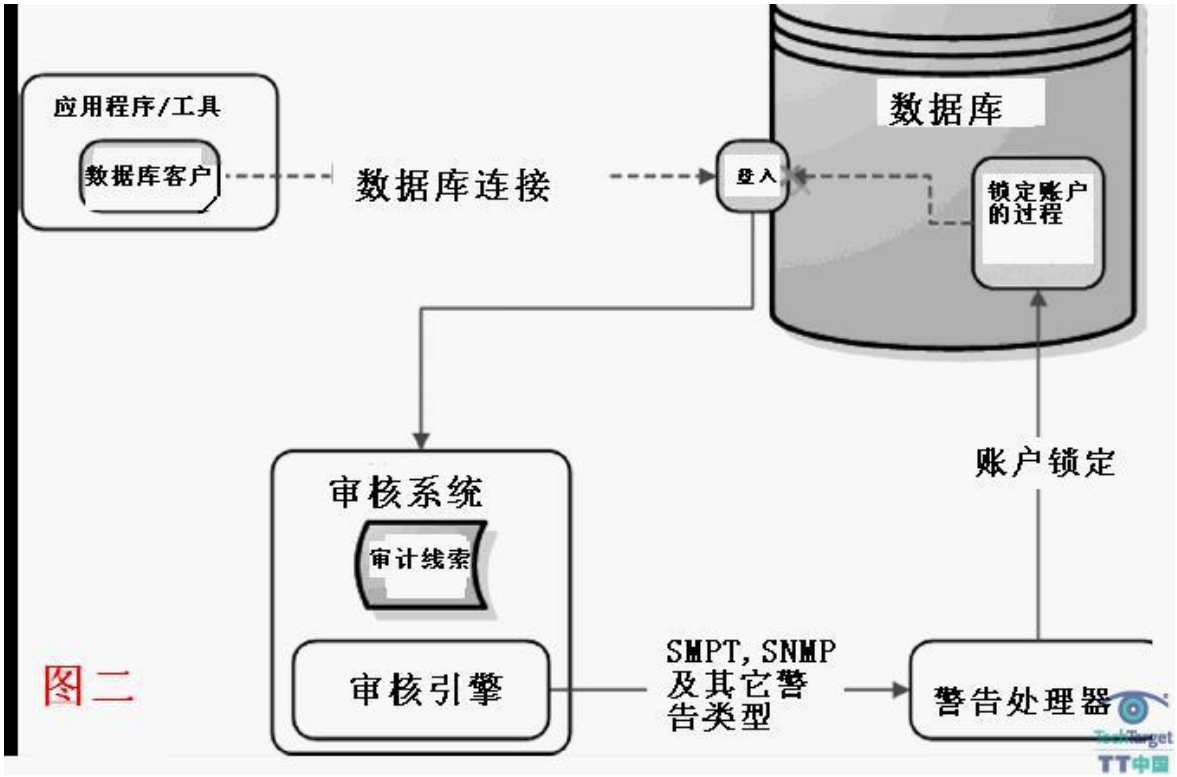
```
create or replace trigger
   user_logout_audit_trigger
BEFORE LOGOFF ON DATABASE
BEGIN
-- logout day
update
   user_login_audit
set
   logout_day = sysdate
where
   sys_context('USERENV','SESSIONID') = session_id;
-- logout time
update
   user_login_audit
set
   logout_time = to_char(sysdate, 'hh24:mi:ss')
where
   sys_context('USERENV','SESSIONID') = session_id;
COMMIT;
END;
```

在 Oracle 环境中这样做就可以了。如果你运行的是 Sybase 环境，问题就更简单了，因为你可以用下面的命令审计对所有数据库的所有访问：

```
sp_configure “auditing”, 1
go
sp_audit “dbaccess”, “all”, “all”, “on”
go
```

根据失败的登录来实施警告或账户锁定这种功能，需要得到数据库厂商或数据库安全解决方案的支持。如果使用数据库来生成登录和登出的审计，并且你的数据库厂商可以实施账户锁定功能，那么，你可以在数据库环境中实现这一点。例如，你可能知道如何建立 Oracle 的口令策略。在另外一种环境中（如 SQL server 2000），你就无法使用本地数据库特性来完成此功能，而需要编写代码来检查 Windows 的事件日志，查找失败的登录，或者使用一个外部的安全系统来实现。

在使用外部安全系统时，你可以使用 SQL 防火墙来阻止经过一定数量的失败登入企图后，某个登录名的任何连接。在这种情形中，数据库并不会收到连接企图，因为这种连接企图会在防火墙水平上被拒绝。另外一个选择（该选择并不要求你将安全系统置于数据库之前）是使用数据库过程，如图二所示。在这种情形中，失败的登录次数超过某个上限后，审计系统就会生成一次审计。该审计被发送给一个负责连接到数据库的系统，并且调用一个过程来锁定账户。该系统一般还会通知数据库管理员，告知已经采取了该行动，以便于进行调查，并且在必要时可以释放被锁定的账户。



除了可用于创建审计线索，登录信息可用于创建一种可以帮助管理员确认异常的基准。用户登录活动的基准是“正常”登录行为的反映。这种基准是通过着眼于在某段时间（例如，一个月）的所有登录活动而构建的。通过将所有的可能性进行组合，对其分类而建立基准。例如，你可以根据网络位置、用户名、源程序、时间和日期来对登录进行分类，此时的基准可能会类似于下面这个样子：

```
user1   192.168.1.168    JDBC     24Hrs.
user2   192.168.X.X      Excel    Normal Business Hours (9-5)
user3   10.10.10.x       isql     Weekends
```

这个基准表明，根据相对记录期中所观察到的所有登录活动，user1 总是从 192.168.1.168（例如这是一台应用程序服务器），并且是 24 小时持续地连接。

User2 用于从 Excel 连接到数据库，它用于 192.168 子网中的多个网络节点中，不能用于正常上班时间之外。最后，在从 isql 发起访问时，就会用到 user3，它可以在周末从 10.10.10 子网的任何节点上工作。

有了这种基准，管理员就可以对背离正常操作的行为做出报告或发出警告。例如，如果你看到有人使用 user1 成功登录，但其 IP 地址却不同于 192.168.1.168，而且所使用的工具却是 SQL Navigator 之类的工具，可以断定，要么你的环境发生了变化，要么是某人成功地获得了应用程序服务器的用户名和口令，并用这些信息从数据库析取信息。再举一个例子，使用 user2 在午夜 2 点的一次登录就非常可疑。当然，这可能表明某个人工作得很晚，但根据具体环境、敏感性等，你可能需要深入地调查此事。

审计使用数据库的源头

与登录活动审计相关的是客户源信息的审计。这包括审计哪个网络节点连接到了数据库（例如，使用一个 IP 地址还是主机名），并且审计使用哪个应用程序访问了数据库。

虽然这种信息是我们在审计数据连接时通常都会捕获的值之一，但在 SQL 调用水平上捕获这种信息非常重要。除了知道一个用户是使用 Excel 而不是 SAP 系统连接之外，你还需要知道某次更新是由 Excel 电子数据表软件还是由 SAP 系统执行的。因此，你在每次查询和数据库操作中，应该将源程序收集在审计记录中，特别当 IP 地址能够唯一确认一个用户时。如果你的架构是基于客户/服务器的，那么源 IP 地址通常会确认一个唯一的用户。这种情况下，根据用户每次 SQL 调用的 IP 地址进行跟踪和报告，这同报告终端用户的数据库操作和数据查看同样有益。另一方面，如果你使用一种应用程序服务器架构，那么 IP 地址不会帮助你确认和报告终端用户，你需要借助于其它技术。

在审计和提供审计信息时，你还得做另一个决定，这与你是提供“原始数据”还是更易于使用的数据有关。例如，上图三中的左侧显示了哪些源程序被用于访问运行在 155.212.221.84 上的 SQL 服务器上。这种信息对于了解环境的人员来说非常有用。而上图右侧所提供的信息对一般人来说更有意义，这些人并不关

心 IP 地址是什么，却知道 HR（人力资源）数据库是怎么回事。如果相关人员理解与开发者工具登录进入人力资源（HR）数据库有关的风险，这种信息显然更有意义。

数据提取问题不仅仅与审计数据库使用的客户源有关。这个问题基本与本文所讨论的所有审计有关。对于源的确认来说，如下图四所示，这尤其重要，其中的 IP 地址可能没有深刻的意义，但属于节点的主机名甚至标签都可提供很多信息。

审计正常操作时间之外的数据库使用

与审计数据库登录有关的另外一个问题是，审计在正常操作时间之外的活动。这是一个非常直接的需求，并且是企业和合规所要求的一种审计。

审计正常操作时间之外的数据库使用非常必要，这是因为在下班期间所进行的活动通常都是可疑的，它有可能是未授权用户试图访问或篡改数据的一个结果和证明。当然，黑客在伪装过程中通常也会试图破坏数据库。

审计下班后的活动时，仅仅跟踪上班时间之外的登入和登出是不够的。一般来说，你还需要捕获执行了哪些活动，这通常在 SQL 水平上完成。如果这种登入可疑，捕获这种登入在使用了什么工具进行了操作是很重要的。全面审计任何用户在正常操作时间之外的所有活动线索是一种很不错的审计，这有助于满足许多规章性的和内部的合规要求。

虽然从直观上看，对下班之后的审计更有意义，但在技术层面上，你必须对其定义保持清晰的头脑，因为多数数据库环境是全天工作的，而且你也不希望生成大量的虚假的警告，尤其是当 ETL 脚本在正常操作时间之外执行海量的数据上传时更是这样。因而，要很好地实施这种审计，关键在于不能把一些一直在下班后运行的任务作为审计线索的一部分。

要排除那些发生在上班时间之外的正常活动，还可以使用另一种方法，即使

用一种基准。如果你为你的数据库访问制定了基准，就会看到下面的活动：

```
user1  192.168.1.168   SQLLoader  2am-4am
user2  192.168.1.168   ETL        12am-6am
```

如果你发现每天晚上这类活动都会发生，那么你对下班后的审计就应当排除这些应用程序（SQLLOADER、ETL）所执行的、使用这些登录名的并且是来自这些 IP 地址的任何活动。仅审计那些偏离基准的对象有助于减少审计内容。

审计 DDL（数据描述语言）活动

设计的变更审计，或者更具体地讲，DDL 的活动审计一直很重要，并且是最常实施的审计线索之一。这是因为设计的变更审计在安全和合规方面，以及从配置管理和过程方面都很重要。从安全的观点来看，DDL 命令都是潜在的最具有破坏力的命令，易被攻击者利用从而破坏系统。窃取信息也会经常涉及到 DDL 命令（例如，创建另外一个表，在析取数据之前将数据复制到其中）。从合规的观点看，许多规范都要求安全人员(管理员)审计对数据表和视图等数据结构的任何修改。

对设计变更进行审计的需求并非总是为了安全原因。有时，这是为了避免错误以及为了快速发现问题。因而，对设计变更进行的审计合规需求，通常与配置管理和 IP 监控任务的部分需求类似。安全人员需要审计数据库的设计变更，并进行保存，作为将来的参考，或作为一种确认和快速解决错误（这种错误可能会破坏数据的便携性或导致数据受到损害）的方法。另外，DDL 活动的审计是为了清除开发者和数据库管理员所引起的错误，以及一些可能会引起灾难影响的错误。笔者的一位客户就由于开发人员的一次变更（开发者认为是自己是在开发服务器上完成的，但实际上却错误地在生产服务器上完成了。）而“宕机”了几乎两天时间。对配置管理过程的紧密控制是很重要的，它是 DDL 审计的首要动因之一。

有三种主要的方法可审计数据库的设计变更：

- 使用数据库的审计特性
- 使用外部的审计系统
- 比较设计快照

多数数据库环境都允许你使用审计机制、事件监视器等来审计 DDL 活动。例如，Oracle 允许管理员使用基于 DDL 的系统触发器：

```
create table ddl_audit_trail
(
   user_id           varchar2(30),
   ddl_date          date,
   event_type        varchar2(30),
   object_type       varchar2(30),
   owner             varchar2(30),
   object_name       varchar2(30)
);
create or replace trigger
  DDL_trigger
AFTER DDL ON DATABASE
BEGIN
    insert into ddl_audit_trail (
       user_id,
       ddl_date,
       event_type,
       object_type,
       owner,
       object_name
    )
    VALUES
    (
       ora_login_user,
       sysdate,
       ora_sysevent,
       ora_dict_obj_type,
       ora_dict_obj_owner,
       ora_dict_obj_name
    );
END;
```

在 SQL Server 中，你要使用迹函数（trace function）,而在 Sybase 中，你就得使用本地函数。只要你愿意，你都可以析取信息、生成报告、创建基准。你可能还需要外部的审计工具。这些工具不仅收集你所关心的信息，而且还提供一些高级功能，用以实现报告、警告及制定基准等。

第三类审计是比较数据库的设计快照，它并不能给你 DDL 活动的详细审计，其重要性远不如其它两类审计，但它易于实施。如果你并不实施一种真正的审计基础架构，那你可以将这种审计作为一种临时解决方案。这种方案基于定期收集数据库设计的完整定义，并将其与以前的设计规划相比较。你甚至可以使用 diff 之类的小工具，因为在这种方法中你要做的一切就是确定是否发生了变化。虽然这种方法极易实施，但是在发生变化时，你却无法跟踪到底是谁做的变更，以及何时、为什么发生变更。此外，如果某人恶意地做了变更，并且利用了它，然后又变回到原先的状态，只要整个过程用时很短，你就无法发现。因而，这个选择有时在配置管理计划中可能是够的，但在一个以安全或合规需求为目标的项目中，这常常是不够的。

审计数据库错误

审计由数据库返回的错误也是很重要的，它是你应实施的首个审计日志之一。从安全的观点来看，这尤其重要。例如，在许多情况下，攻击者会进行很多尝试直至得逞。攻击者可以使用基于 UNION 的攻击，他需要猜测数据表的正确栏数，直到他得到正确的数字，数据库将会不断地返回一个错误代码，表明 SELECT 语句所选择的栏数不匹配。如果你记录了所有的错误，就可以确认这种攻击并做出响应。失败的登录是需要进行记录和监视的错误之一，即使你并没审计数据库的登录。最后，任何失败的提升特权的试图操作都表明攻击正在发生。

从质量的观点看，错误审计也很重要，它符合合规要求。我们应该确认并修复漏洞和应用程序错误，而记录 SQL 错误通常是确认这些问题的一种简单方法。因而，即使你关心的是安全问题，将这种信息提供给应用程序的所有者也很有意义，因为谁都不愿意运行存在着问题的代码。幸运的话，这些错误甚至会向你指出那些影响响应时间和可用性的问题。

数据库厂商都支持详细的错误审计，你可以参考具体的数据库环境指南。在 Oracle 中，你可以使用系统触发器：

```
create table error_audit
(
   user_id              varchar2(30),
   session_id           number(8),
   host                 varchar2(30),
   error_date           date,
   error                varchar2(100)
);
```

下一步，创建一个可用于新登录的触发器：

```
create or replace trigger
  audit_errors_trigger
AFTER SERVERERROR ON DATABASE
BEGIN
insert into error_audit values(
   user,
   sys_context('USERENV','SESSIONID'),
   sys_context('USERENV','HOST'),
   sysdate,
   dbms_standard.server_error(1)
);
COMMIT;
END;
```

在 SQL Server 中，你可以使用审计特性或跟踪特性。如果你选择使用跟踪特性，你需要使用 sp_trace_event 来建立与错误有关的正确事件。这包括下表所示的事件 ID：

事件 ID	事件类型	描述
16	注意	收集所有的注意事件，如客户中断请求，或客户的连接中断。
21	事件日志（EventLog）	错误事件被记录在错误日志中。
22	事件日志（EventLog）	事件已被记录在应用程序日志中
33	例外（Exception）	服务器中发生了意外。
55	哈希警告	哈希操作可能遇到了问题
61	OLEDB 错误	发生了 OLE DB 错误

多种 DB2 事件监视器与错误审计相关，并且你必须使用这些事件类型。对于所需要的每一种类型，你都应当过滤那些与错误相关的记录。例如，你应当为拒绝访问（ACCESS DENIED）记录选择检查（CHECKING）记录，并在 VALIDATE 类型中查看 AUTHENTICATE_PASSWORD 和 VALIDATE_USER 事件。

虽然在有些环境中，记录错误和审计错误是可能的，但外部的审计系统更加普遍。如果你监视所有进入的 SQL 请求和响应，跟踪和报告所有的错误是很简单的，且不会给数据库带来任何额外负担。可以使用任何一种标准集来报告错误，这种信息对于制定基准是很益的。

如果你的应用环境不太完善，制定基准就显得很重要。不是每一个数据库和应用环境都十全十美，在多数环境中，有些应用程序会产生数据库错误，即使是在生产过程中也会如此。事实上，应用程序所产生的错误是重复性的：在大体相同的地方发生了相同的错误，这类错误往往是由缺陷引起的，且这些缺陷并不会

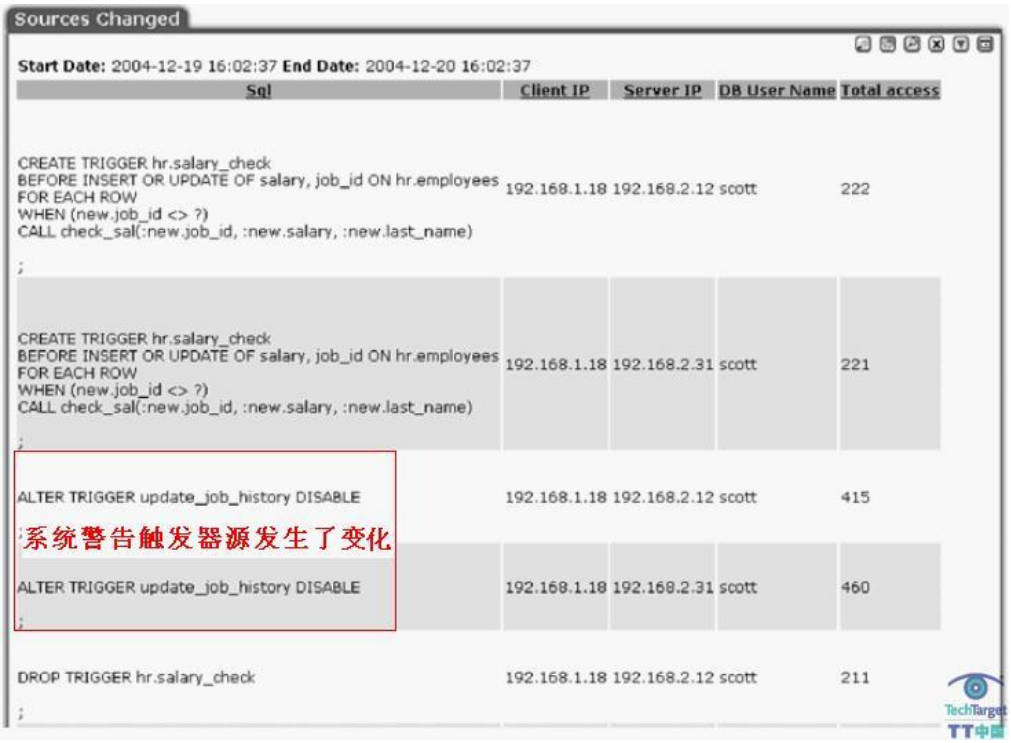
发生改变。如果制定了错误基准，却突然发现其它方面所产生的错误，或者你看到了完全不同的错误代码，那么你该调查发生了什么问题。

存储过程和触发器的来源变更审计

由于数据库木马使用了灵活但却完全过程化的编程语言，要隐藏恶意代码是很容易的。因此，你应采取最佳方法来审计这些结构的所有变更。

有几种方法可实现这种审计。最简单的方法是基于配置控制的，可以通过定期（如每天）从数据库中检索代码并将其与前一段时期所检索的代码相比较来实施这种审计。通过使用一些工具和脚本程序（如 diff），这种方法相对易于实施。

第二种选择是使用一个外部的数据库安全和审计系统。这种系统可以实时地对创建或修改命令向管理员发出警告，并且很轻松地就生成可以详细描述各种变化的报告，下图显示的就是触发器的变更：



第三个选择是使用内置的数据库特性。例如，在 SQL Server 中，你可以使用 Recompile 事件来跟踪对存储过程的变更：

事件ID	事件类型	描述
37	SP：Recompile	指明存储过程被重新编译

在多数数据库环境中，这种特性是通过 DDL 审计来支持的，虽然从命令中析取源代码并以一种易于审计的方式维持代码并不容易。

审计特权、用户、登录定义和其它安全特性的变更

这类审计对于数据库的审计来说是必须的；你必须对数据库安全和特权的任何变更维持一套完整的审计记录。数据库管理着安全、许可、变更的复杂规划，在安全问题中的首要规则是，必须审计安全形势的任何变更。考虑审计下面的这些变更：

1、增加和删除用户、登录、角色等

2、登录和用户(角色)之间的映射关系发生变更

3、特权变更（无论是由用户还是由角色引起的）

4、口令变更

5、在服务器、数据库、语句或对象水平上对安全属性的变更

由于数据库内的安全模式是一个入口，所以必须审计对特权和许可的任何变更。攻击者会经常提升其特权水平，而且在管理员提供了错误的许可、授权后也经常出现错误。因而，对可能影响数据库安全状况的所有变更都进行完整的审计，就如同将监视摄像机放置在大楼的入口处一样，必须审计登录凭证是否发生变化。

安全许可的变更对数据库极其重要，仅依赖于每天的比较是不够的，应该选择实时变更通知,即实时审计那些没有在生产环境中提前规划的变更。这意味着你应当使用一种外部的数据库安全和审计系统，或者使用内置的数据库机制所生成

的审计线索来构建实时的警告。

如果你打算自己实施这种系统，就需要捕获相关事件，然后构建警告框架。例如，下表显示了可用于 SQL Server 的相关事件。

事件ID	事件类型	描述
102	审计语句GDR	在 SQL Server 中任何用户在每次发布 GRANT、DENY、REVOKE语句许可时，发生此事件。
103	审计对象GDR	在 SQL Server 中任何用户在每次发布 GRANT、DENY、REVOKE对象许可时，发生此事件。
104	审计添加或丢弃登录	在添加/清除一次 SQL Server 登录时（sp_addlogin、sp_droplogin），发生此事件。
105	审计登录GDR	在添加或清除一次 Windows 登录权限时（sp_grantlogin、sp_revokelogin、sp_denylogin），会发生此事件。
106	审计登录的属性变更	在一次登录的属性等被修改时（sp_defaultdb、sp_defaultlanguage），发生此事件。
107	审计登录的口令变更	在SQL Server登录的口令发生变更时，发生此事件。
108	审计将登录添加到服务器角色中的活动	在向固定的服务器角色增加或清除登录时（sp_addsrvrolemember 和 sp_dropsrvrolemember），发生此事件。
109	审计添加数据库用户的活动	在将一次登录作为数据库用户（Windows或 SQL Server）进行增加或清除时（sp_grantdbaccess, sp_revokedbaccess, sp_adduser, sp_dropuser）发生此事件。
110	审计将成员添加到数据库中的活动	在将一次登录作为一个数据库用户添加（固定的或用户定义的）到或从一个数据库中清除时（sp_addrolemember, sp_droprolemember, sp_changegroup），发生此事件。
111	审计增加/丢弃角色的活动	在将一次登录作为一个数据库用户添加到或从一个数据库中清除时（sp_addrole 和 sp_droprole），发生此事件。
112	应用程序角色的口令发生变更	在一个应用程序角色的口令发生变更时，发生此事件。
113	审计语句的许可	在使用语句许可（如CREATE TABLE）时，发生此事件。
114	审计对象的许可	在使用对象许可（如SELECT）时，不管成功与否，都发生此事件。

在 DB2 中，SECMAINT 是六类审计之一，在授权和撤消对象或数据库特权时，或者在许可和撤消 DBADM 权限时，会生成记录。在修改数据库管理员的安全配置参数（SYSADM_GROUP、SYSCTRL_GROUP、 SYSMAINT_GROUP）时，也会生成记录。下表列示了一些可能的 SECMAINT 的特权或授权：

DB2 SECMAINT事件	描述
Control table	控制对一个表格或视图的特权许可或撤消
ALTER TABLE	修改表格的特权授权或取消
ALTER TABLE with GRANT	通过准许的特权，准许修改或撤消修改表格或视图的特权授权。
DELETE TABLE	授与或取消丢弃表格或视图的特权
DELETE TABLE WITH GRANT	通过准许的特权，准许清除或撤消清除表或视图的特权。
TABLE INDEX	准许或撤消索引的特权
TABLE INDEX WITH GRANT	通过准许的特权，准许或撤消索引的特权
TABLE INSERT	授与或撤消对表或视图进行插入的特权
TABLE INSERT with GRANT	通过准许的特权，准许或撤消对表的插入
TABLE SELECT	准许或撤消对表的选择
TABLE SELECT WITH GRANT	通过准许的特权，准许或撤消对表的选择
TABLE UPDATE	准许或撤消对表的更新特权。

在这种情况下，你可以建立一组希望用于跟踪的命令。如下图所示。

然后将规则（相关规则显示在下图中）添加到策略中，在使用这种命令时向安全管理人员发出警告。策略中的规则确保了安全管理人员能收到关于这种命令的警告，但即使没有规则，你仍能获得完整的审计线索，这种线索包括用户组中发生的任何命令。

与前面几种审计不同，对链接、同义词或昵称的审计以及对创建副本过程的审计都表明，定期析出和比较数据已经足够了。虽然你有三个选择：比较快照，

使用数据库的内部审计机制，使用外部审计和安全系统。但实际上，使用 diff 这个小工具就足以应对。这种情形下，你只需一段能够查询这些定义的脚本，并将这些定义放在一个可用于与未来的某天进行比较的文件中。

如果你喜欢使用内部的数据库审计机制或使用外部的审计系统，就得将这些审计线索建立在对象和命令上。在多数数据库环境中，没有专门的副本和链接的审计功能。不过，你仍有许多特定的审计对象和命令。例如，你可以使用与副本有关的 SQL Server 对象。

审计敏感数据的变更

对数据操纵语言（DML）的审计是另一种常见需求，特别是在财务信息的正确性成为主要问题时。

相关的审计需求包括，全面记录每一次 DML 活动的新值和旧值。例如，你可能需要为雇员表中存储年金的那一“列”创建审计跟踪线索。在这种情况下，你有两种不同的要求。第一个要求是完全记录对这些值的任何更新，对于每次更新，要记录每个执行更新的用户，使用的客户端和应用程序，以及更新时间和真实的 SQL 语句。第二个要求是记录上述所有信息，以及记录更新前和后的值。不过情况并非总是一样的，因为通过使用下面的命令，笔者可以将 50%的年金归为己有：UPDATE EMP SET BONUS = BONUS * 1.5。

尽管 DML 的审计跟踪并记录新值和旧值是重要的审计类型。但在使用此技术时仍需小心，要有选择地实施这种审计。有时，人们过分热衷于这种审计，因为简单方便，就为每一次 DML 操作都启用了这种审计。这在技术上是可能的，但可能会产生巨大的数据量，应该确保你的审计基础架构可以管理这种负荷，特别是在你的审计包括新值和旧值时。例如，单位每天发生大量业务，为了简单起见，每笔业务都仅更新一次值，每个数据库有多个表，每个表有若干个值需要更新，每个表有大量记录。一年后，你会发现你的审计数据库是超过数据库自身的 35 倍。

因此，在你准备 DML 审计线索时，应当谨慎地选择需要审计的对象和命令。

例如，你可以决定为一个数据库的表集来创建审计，为某些登录名或账户创建审计等。更深入的选择是，你可以选择为哪些表及表中的哪些列来维护新值和旧值。

还可以通过三种主要方法来支持 DML 审计：使用数据库自身的功能、使用一个外部的审计系统，或使用触发器。

所有数据库都提供了实施 DML 活动审计的方法。例如，在 Oracle 中，你可以使用基于 redo 日志的“日志矿工（log miner）”工具。因为 redo 日志会捕获所有的 DML 活动（包括新值和旧值），“日志矿工（log miner）”可以析取这种信息，并使其可用。在 SQL Server 中，你可以使用 DOP 跟踪事件：

事件ID	事件类型	描述
28	DOP事件	在执行SELECT、INSERT或UPDATE语句之前发生此事件。

第二种方法，即外部的数据库审计系统，支持基于过滤标准的 DML 审计，包括数据库对象、用户、应用程序等。这种系统有助于捕获并压缩信息，并便于报告工具的使用，即使数据量很大也不会产生太多问题。

第三种选择是简单地使用自己定制的触发器。这种选择从技术上而言，不如其它的选择，但是如果你从事的并不是一个大规模的审计项目的一部分，而仅仅需要为几个对象创建一个 DML 审计线索，不妨增加触发器，将信息写到一个特定的审计表中，这种最简单而快捷的方法也许可以帮你顺利进入下一个项目。

审计关于私密问题的 SELECT 语句

SELECT 语句过去并不是审计线索的重点，但最近对私密问题的重视已经改变了这种情况。例如，如果你需要确保客户、合伙人、雇员的机密信息不会从数据库中泄露，就得重视审计 SELECT 语句。你尤其需要显示出 SELECT 语句来自哪里（IP 地址、应用程序）、谁选择了数据（用户名）、选择了哪些数据等。在 DML 的审计中，SELECT 语句的审计对于整个数据库而言并不现实，你需要重视有重要意义的必要问题。

第一步是根据 SELECT 线索审计区分出哪些数据重要。例如，人的姓氏不太算机密，但姓氏与身份证号结合在一起就绝对是机密。在数据分类阶段，你应当定义机密信息存在哪里（对象名及列名），以及哪些信息组合是机密信息。

创建 SELECT 审计线索通常要比其它审计类型更为困难。很明显，在这里，快照并不是一个可行的选择，触发器也不行，所以你只能使用数据库跟踪或外部的审计系统。你还可以选择使用定制日志来构建视图，但这样做要求的工作量太大。在使用内部的数据库特性时，你的选择更为有限。例如，即使你可以跟踪 SELECT（例如，下图所示:在 SQL Server 中使用 DOP 事件），这也往往是不现实的，因为你要收集太多的信息，并需要应用过滤器。

事件ID	事件类型	描述
28	DOP事件	在执行SELECT、INSERT或UPDATE语句之前发生此事件。

因而，在你需要执行 SELECT 审计时，最佳选择通常是使用一个外部的数据库审计系统。注意，并非所有的方法都支持 SELECT 审计。例如，一个基于业务日志的方案在使用 SELECT 审计线索时就没有什么作用。

对审计对象的定义变更实施审计

必须审计对审计对象的定义所做的任何更改。如果你有一个可以监控办公楼的监控设备，不要忘了监视其所指向的方向。否则，入侵者可以将监控设备指向墙壁，或者用纸粘在镜头上，然后再从事其它不法活动。同样地，如果你不审计对审计对象、审计线索的定义所做的变更，攻击者就可以改变审计对象的定义从而从事不法活动。

你可以使用内置的数据库特性或使用外部的数据库安全和审计系统。例如，DB2 的 AUDIT 审计，在审计设置被改变或有人访问审计日志时，它就会生成记录。而 SQL Server 拥有下表中所示的跟踪事件：

事件ID	事件类型	描述
117	对审计的变更进行审计	在发生审计修改时发生此事件

如果你选择使用外部的审计系统，务必确保它能够自动完成全部的审计功能。

总结

本次数据库电子书介绍了多种不同的审计线索。为了确保自己的数据库环境的安全，也为了遵循规范或内部要求，你需要实施这些审计。虽然不同的单位有不同需求，但都能够映射到一系列数据库审计功能上。

为了选择适合自己需要的审计类型，你需要选择用于实施审计线索的方法和系统，并做出关于架构的决定。这一点很重要，因为审计并非权宜之计，你的部

署应当经得起时间的考验。

我们的编辑团队

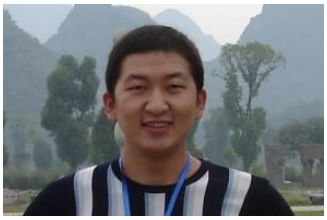
您若有何意见与建议，欢迎与我们的编辑联系。

诚挚感谢以下人员热情参与 TechTarget 中国《数据库电子书》的内容编辑工作！

诚邀更多的数据库专业人士加入我们的内容建设团队！

曾少宁

TechTarget中国特邀技术编辑。软件工程硕士学学位，4年以上软件开发经验，熟悉Oracle、Java以及Linux等领域，曾经任职于juniper等著名企业，目前从事计算机教学工作。



冯昀晖

TechTarget中国特邀技术编辑。资深软件工程师，有超过7年的政府和企业信息化软件解决方案经验，熟悉SQL Server、Oracle等数据库技术，爱好阅读、健身和中国象棋。

孙瑞

TechTarget 中国高级网站编辑，四年网络媒体从业经验。负责“TT 数据库”和“SearchBI”网站的内容建设，熟悉数据库以及商业智能等企业信息化领域，拥有计算机学士学位。