Oracle 区块链表

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-21

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：
400-699-8888

* 活动最终解释权归甲骨文公司所有

Oracle 区块链表

甲骨文技术公益课 - 数据库专场

2023 年 8 月 18 日 11:00

线上直播

范宏伟

Oracle区块链产品

可管理的服务以建立区块链驱动的解决方案

Ready-To-Build

Oracle Blockchain Platform
甲骨文区块链平台

区块链平台
云服务

区块链平台
企业版

Oracle数据库加密安全管理

Ready-To-Use

区块链表 Blockchain Table

打包的区块链应用
加速区块链采用

Ready-To-Go

SaaS Applications
软件及服务应用

Oracle 智能跟踪和溯源

为什么要使用 Oracle 区块链表

问题：常见漏洞和安全挑战

内部攻击
Inside hacker

非法修改
Illicit changes

虚假身份
False Identity

开源区块链

的复杂性

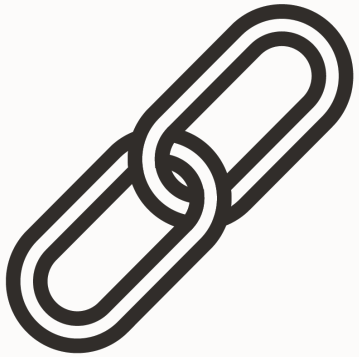
对重要记录的非法修改可能会带来可怕的后果 - 资产损失、法律风险、声誉损失等

为什么要使用 Oracle 区块链表

解决方案：加密安全数据管理

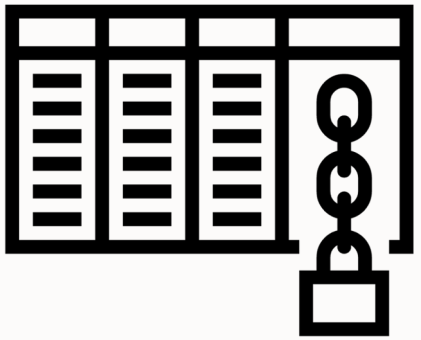
No security bypass

密码链
Cryptographic chain



最终用户
数字签名

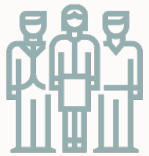
易于部署



数据库加密安全

- 现有的数据安全技术专注于将非法访问拒之门外

Passwords



Privileges

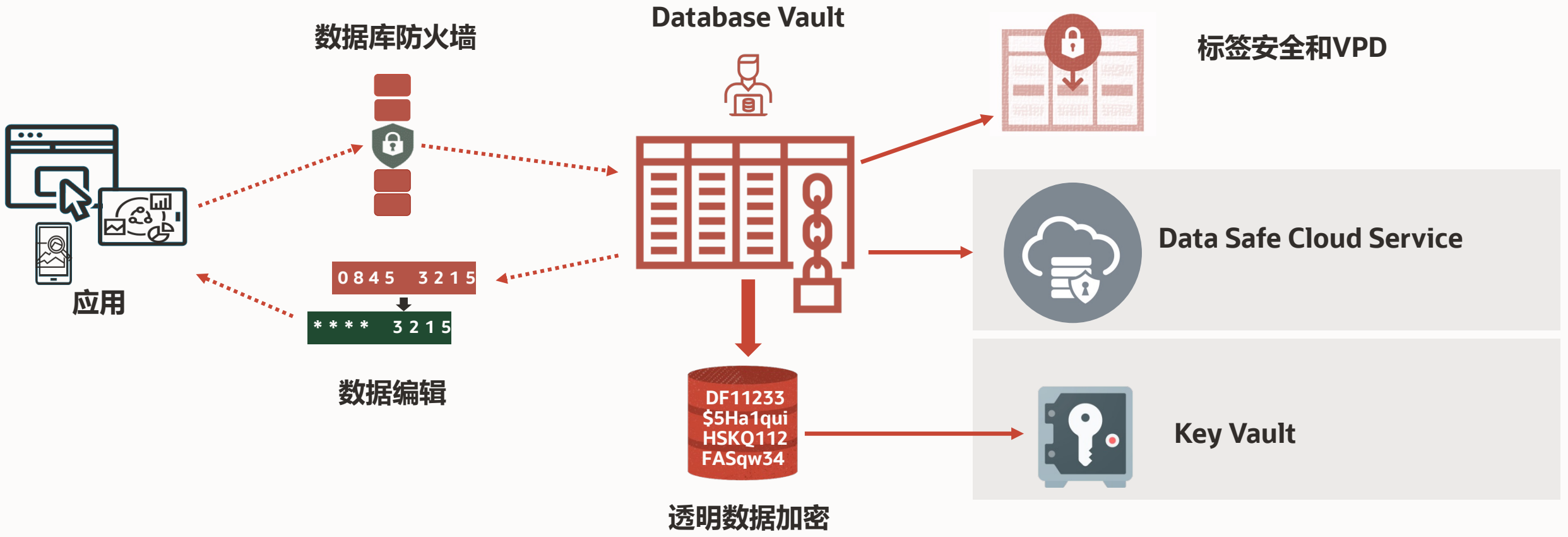
Encryption

Firewalls

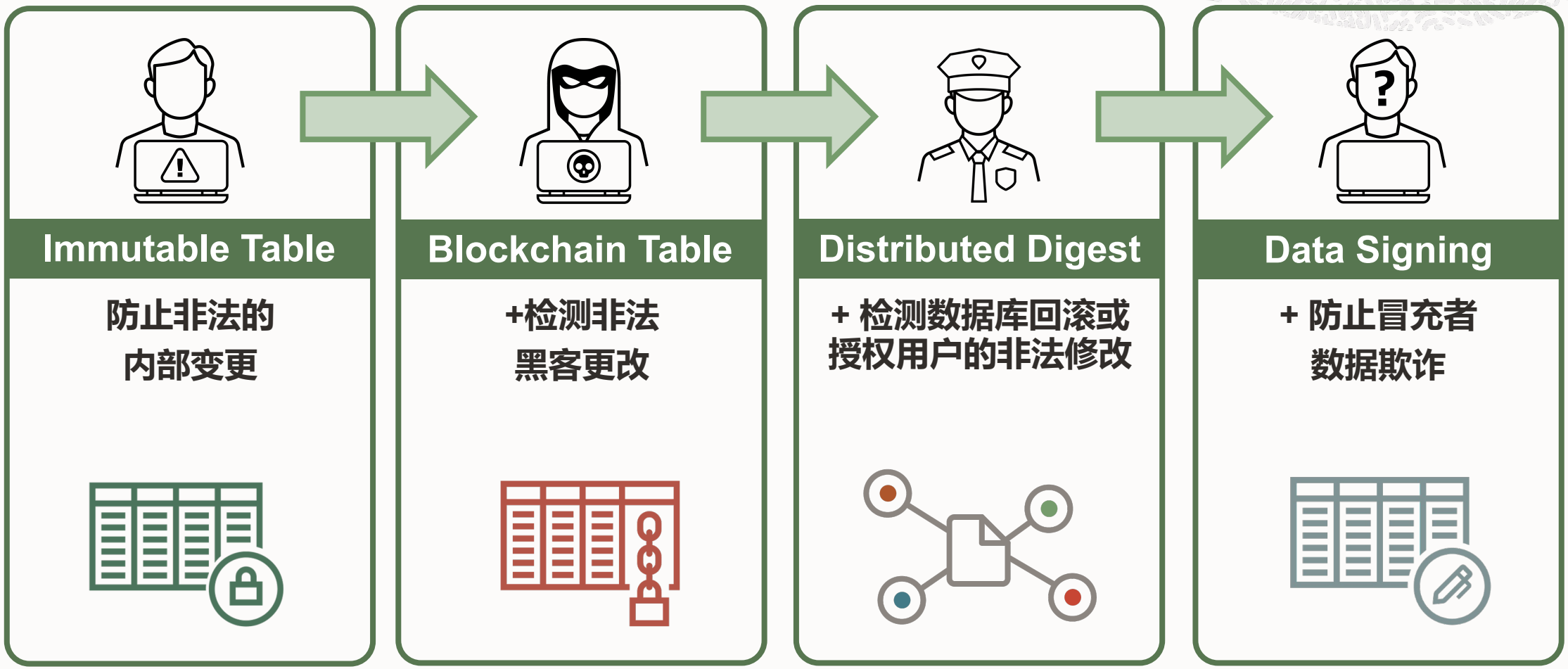
- 区块链增加了另一层数据安全
  - 保护数据不被非法修改或删除

- 使用加密摘要（**cryptographic digests**）和签名（**signatures**）实现加密安全数据管理

区块链增强了 Oracle 独特的高安全架构



Oracle 提供针对多种类型非法更改的保护



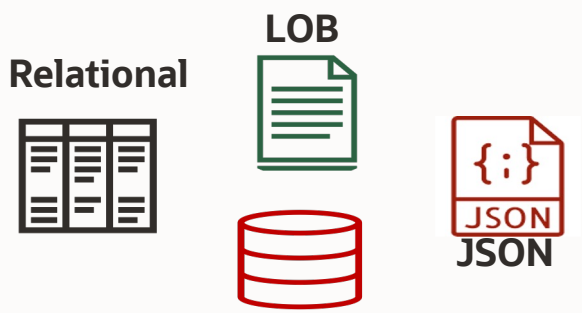
可以有针对性的逐步采用 Oracle 加密安全技术

Oracle不可变表（Immutable Tables）

```
CREATE IMMUTABLE
TABLE trade_ledger (…);
```

TRADE LEDGER

ID	User	Value
1	Tom	500
2	Carol	176
3	Wang	500
4	Eve	25

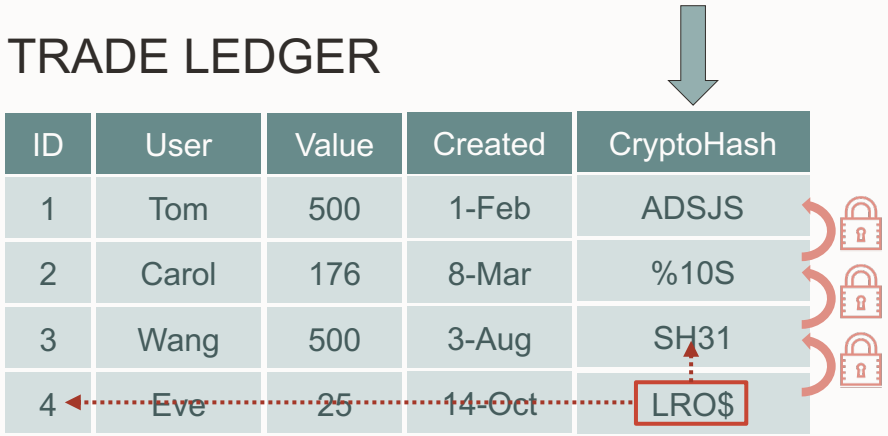


- 只需在表创建语句中添加“Immutable”即可创建仅插入的不可变表
  - 允许关系数据和 JSON 或 LOB 文档
  - 不限于账本，可以存储参考数据
- 操作使用与任何其他表类似，但不可变表**不能**：
  - 更新行（但可以插入行的新版本）
  - 删除行（除非它们已过期）
  - 重命名列（但允许添加/删除）
  - 将不可变表转换为可更新表，反之亦然
  - 修改数据库字典中的表元数据
- 使用不可变表不需要更改应用程序

Oracle区块链表（Blockchain Tables）

```
CREATE BLOCKCHAIN
TABLE trade_ledger (…);
```

TRADE LEDGER



ID	User	Value	Created	CryptoHash
1	Tom	500	1-Feb	ADSJS
2	Carol	176	8-Mar	%10S
3	Wang	500	3-Aug	SH31
4	Eve	25	14-Oct	LRO$

- 区块链表是一个不可变表，它自动以加密方式将新行链接到现有行
  - 只需将“区块链”添加到建表语句中即可
- 新插入的行具有基于新行内容计算的加密哈希值
  - 加上前一行的加密哈希值
- 对行链中数据的任何修改都会破坏加密链

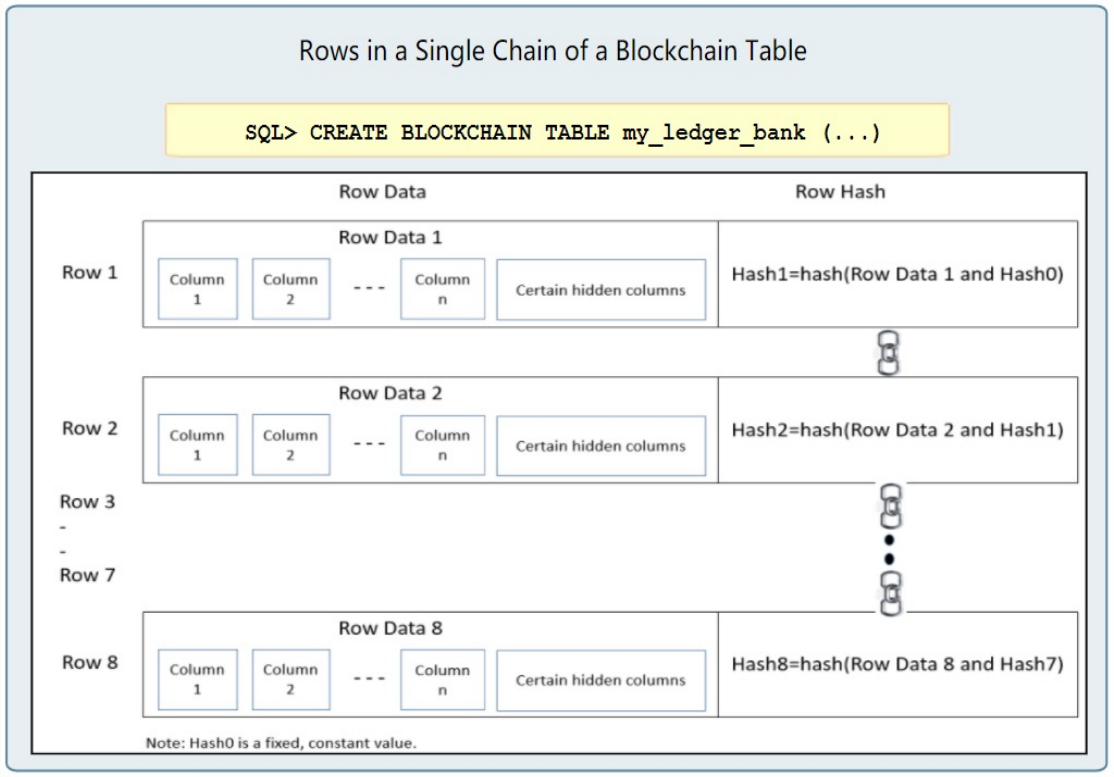
不可变表 vs. 区块链表

不可变表	区块链表
不可变表可防止有访问权限的内部人员进行恶意的未经授权的更改。	除了可防止有访问权限的内部人员进行恶意的未经授权的更改,区块链表还可以做: • 检测绕过 Oracle 数据库软件进行的未经授权的更改 • 检测最终用户假冒行为以及在未经用户授权的情况下以用户名插入数据的行为 • 防止数据篡改并确保数据实际插入到表中
没有内部行记录的链接	除第一行之外的每一行都通过使用加密哈希链接到前一行。 行的哈希值是根据该行数据和链中前一行的哈希值计算的。 对行的任何修改都会破坏链，从而表明该行被篡改。
插入行不需要在提交时进行额外的处理。	在提交时，需要额外的处理时间来链接行。

区块链表是在 21c 中引入的，然后向后移植到 19.10；

不可变表同时被引入到 Oracle 21.3 和 19.11 中，compatible 参数设置 19.11.0 之上

Blockchain Table 原理



- 行数据由用户列和某些隐藏列组成
- 一行的哈希值是基于行数据和链中前一行的散列值来计算的
- 单个事务可以将行插入多个区块链表中
- 由单个交易插入的区块链表中的行被添加到同一个链中，它们在链上的位置遵循它们插入区块链表的顺序
- 当多个用户同时将行插入区块链表中的同一链中时，添加行的顺序取决于插入这些行的事务的提交顺序

区块链表的“隐藏列”

列名	类型	描述
ORABCTAB_INST_ID$	NUMBER (22)	插入行的数据库实例的实例ID
ORABCTAB_CHAIN_ID$	NUMBER (22)	数据库实例中插入行的链的链ID（链ID的有效值为0到31）
ORABCTAB_SEQ_NUM$	NUMBER(22)	链上的行的序列号，插入区块链表链中的每一行都分配了一个以1开头的唯一序列号。一行的序列号比链中前一行的顺序号高1。使用此列可以检测到缺少的行。实例ID、链ID和序列号的组合唯一标识区块链表中的一行。
ORABCTAB_CREATION_TIME$	TIMESTAMP WITH TIME ZONE	创建行时的时间（UTC格式）
ORABCTAB_USER_NUMBER$	NUMBER (22)	插入行的数据库用户的用户ID
ORABCTAB_HASH$	RAW(2000)	行的哈希值。哈希值是基于该行的行内容和链中前一行的哈希值来计算的。
ORABCTAB_SIGNATURE$	RAW(2000)	行的用户签名。签名是使用行的哈希值计算的
ORABCTAB_SIGNATURE_ALG$	NUMBER(22)	用于生成已签名行的用户签名的签名算法
ORABCTAB_SIGNATURE_CERT$	RAW(16)	与已签名行上的签名关联的证书的GUID。

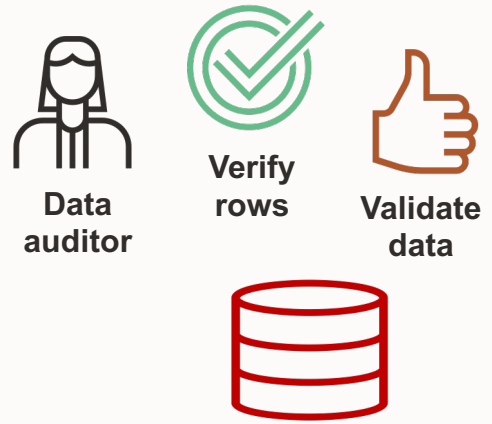
区块链表验证

DBMS_BLOCKCHAIN_TABLE.
VERIFY_ROWS()

TRADE LEDGER

ID	User	Value	Created	CryptoHash
1	Tom	500	1-Feb	ADSJS
2	Carol	176	8-Mar	%10S
3	Wang	500	3-Aug	SH31
4	Eve	25	14-Oct	LRO$

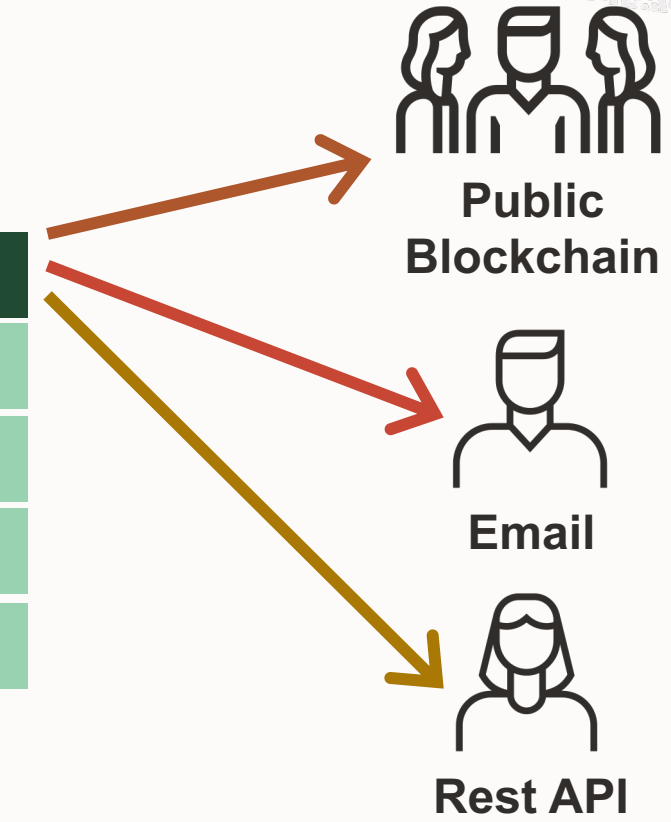
- 可由用户验证：
  - 调用 DBMS_BLOCKCHAIN_TABLE.VERIFY_ROWS()
- 可以独立验证，而不依赖于数据库
  - 在数据库外部运行的开源library可用于读取链式数据并验证加密摘要
- 独立的数据审计员可以验证数据



分布式加密摘要

TRADE LEDGER

ID	User	Value	Created	CryptoDigest
1	Tom	500	1-Feb	ADSJS
2	Carol	176	8-Mar	%10S
3	Wang	500	3-Aug	SH31
4	Eve	25	14-Oct	LRO$



数据签名

DBMS_BLOCKCHAIN_TABLE.SIGN_ROW

TRADE LEDGER

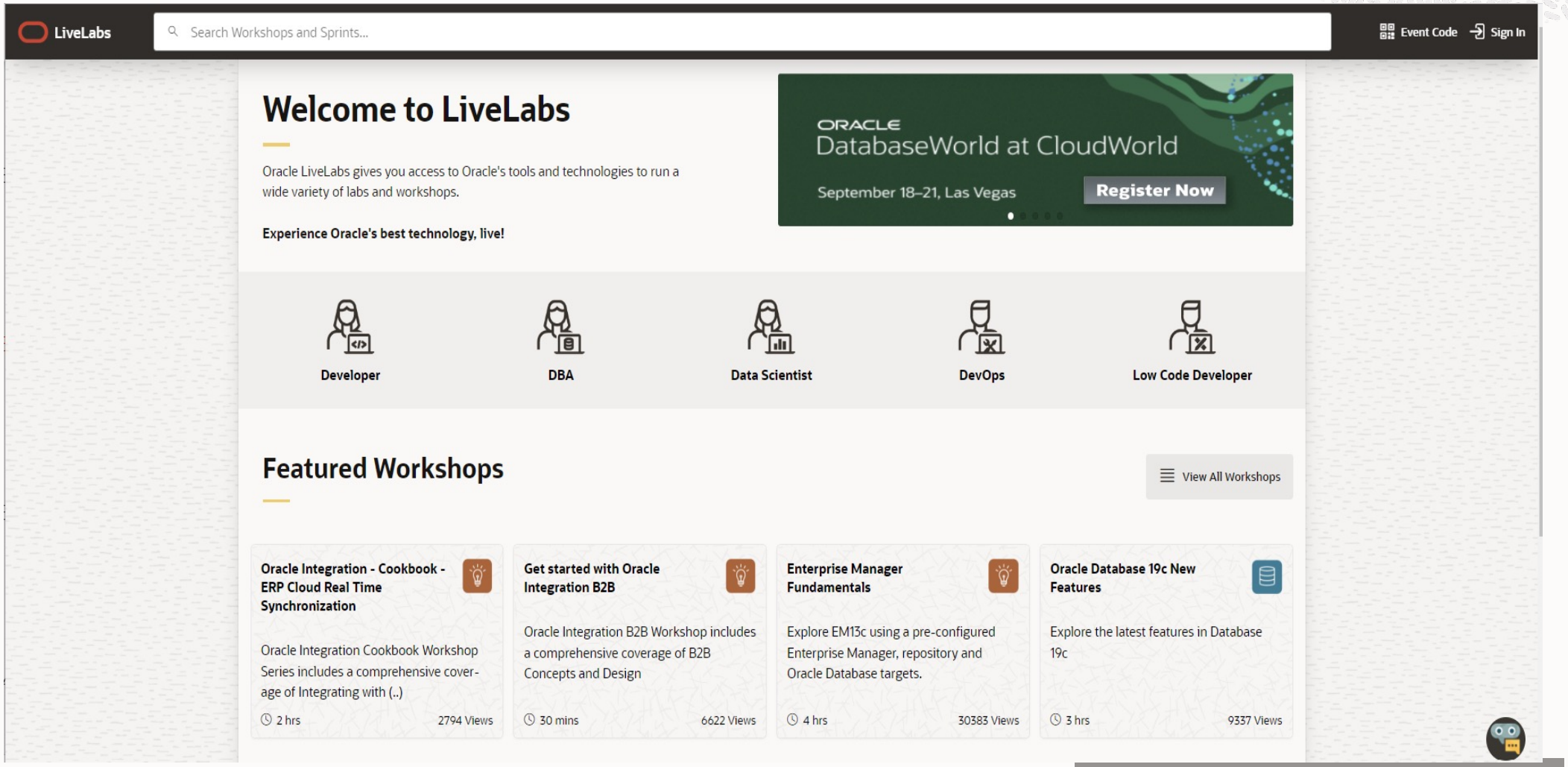
ID	User	Value	Created	CryptoDigest	Signature	User_Cert_ID
1	Tom	500	1-Feb	ADSJS	RT#E	GRTE
2	Carol	176	8-Mar	%10S	GI(!	SOQP
3	Wang	500	3-Aug	SH31	HV*P	OPRT
4	Eve	25	14-Oct	LRO$	N@P%	LCZI

Row signature

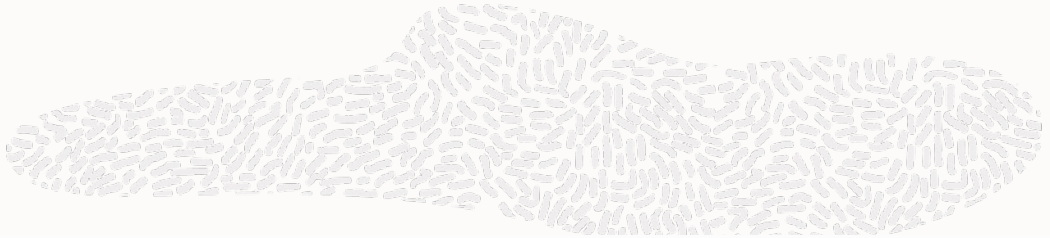
User certificate ID

动手实验：Oracle LiveLabs

https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs/home



区块链表实验



A centralized ledger model reduces administrative overheads of setting a up a decentralized ledger network, leads to a relatively lower latency compared to decentralized ledgers, enhances developer productivity, reduces the time to market, and leads to significant savings for the organization. Database users can continue to use the same tools and practices that they would use for other database application development.

Estimated Time: 30 minutes

Objectives

In this lab, you will:

- Create the blockchain table
- Insert and delete rows
- Drop the blockchain table
- Check the validity of rows in the blockchain table

Prerequisites

- An Oracle Account
- SSH Keys
- Create a DBCS VM Database
- 21c Setup

LAB 1：创建一张Blockchain Table
LAB 2：插入数据
LAB 3：删除数据
LAB 3：drop区块链表
LAB 5： BlockChain数据验证

创建一张Blockchain Table

```
SQL> create blockchain table bct_t1 (
  2    id             number,
  3    fruit          varchar2(20),
  4    quantity       number,
  5    created_date   date,
  6    constraint bct_t1_pk primary key (id)
  7  )
  8  no drop until 0 days idle
  9  no delete until 16 days after insert
 10  hashing using "SHA2_512" version "v1";

表已创建。
```

```
SQL> create blockchain table bct_t3 (
  2    id             number,
  3    fruit          varchar2(20),
  4    quantity       number,
  5    created_date   date,
  6    constraint bct_t2_pk primary key (id)
  7  )
  8  no drop until 1 days idle
  9  no delete until 6 days after insert
 10  hashing using "SHA2_512" version "v1";
create blockchain table bct_t3 (
*
第 1 行出现错误:
ORA-05741: 最低保留时间太低，应至少为 16 天
```

- **不能在CDB创建区块链表**
- ***NO DROP, NO DELETE, HASHING USING, VERSION关键字是必须的***
- **NO DROP UNTIL n DAYS IDLE（n>=0），建议>=16**
- **NO DELETE UNTIL n DAYS AFTER INSERT [LOCKED]，n>=16，只能增加不能修改。如果LOCKED，保留时间不能修改**

查看隐藏列

```
SQL> set linesize 120 pagesize 50
SQL> column column_name format a30
SQL> column data_type format a27
SQL> column hidden_column format a13
SQL>
SQL> select internal_column_id,
  2         column_name,
  3         data_type,
  4         data_length,
  5         hidden_column
  6  FROM   user_tab_cols
  7  WHERE  table_name = 'BCT_T1'
  8  ORDER BY internal_column_id;

INTERNAL_COLUMN_ID COLUMN_NAME                    DATA_TYPE                   DATA_LENGTH HIDDEN_COLUMN
------------------ ------------------------------ --------------------------- ----------- -------------
                 1 ID                             NUMBER                               22 NO
                 2 FRUIT                          VARCHAR2                             20 NO
                 3 QUANTITY                       NUMBER                               22 NO
                 4 CREATED_DATE                   DATE                                  7 NO
                 5 ORABCTAB_INST_ID$              NUMBER                               22 YES
                 6 ORABCTAB_CHAIN_ID$             NUMBER                               22 YES
                 7 ORABCTAB_SEQ_NUM$              NUMBER                               22 YES
                 8 ORABCTAB_CREATION_TIME$        TIMESTAMP(6) WITH TIME ZONE          13 YES
                 9 ORABCTAB_USER_NUMBER$          NUMBER                               22 YES
                10 ORABCTAB_HASH$                 RAW                                2000 YES
                11 ORABCTAB_SIGNATURE$            RAW                                2000 YES
                12 ORABCTAB_SIGNATURE_ALG$        NUMBER                               22 YES
                13 ORABCTAB_SIGNATURE_CERT$       RAW                                  16 YES
                14 ORABCTAB_SPARE$                RAW                                2000 YES

已选择 14 行。
```

```
SQL> SELECT ORABCTAB_CHAIN_ID$ "Chain ID", ORABCTAB_SEQ_NUM$ "Seq Num",
  2             to_char(ORABCTAB_CREATION_TIME$,'dd-Mon-YYYY hh-mi') "Chain date",
  3             ORABCTAB_USER_NUMBER$ "User Num", ORABCTAB_HASH$ "Chain HASH"
  4  FROM   bct_t1;

 Chain ID   Seq Num Chain date          User Num
--------- --------- ------------------- --------
Chain HASH
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------

       25         1 14-8月 -2023 11-30        109
8CBF8D1ABCFFD33428D4289805B8A08B5BABECBE97D118D71E67B12201FC7A87440028C77174DA52D8CD1D613C84601FEFE2AD0CEACF121565A10755A9CCC727
```

查看区块链表基本信息

```
SQL> column table_name format a10
SQL> column row_retention format 9999999999999
SQL> column row_retention_locked format a20
SQL> column table_inactivity_retention format 999999999999999999999999
SQL> column hash_algorithm format a14
SQL> column table_version format a14
SQL>
SQL> select table_name,
  2         row_retention,
  3         row_retention_locked,
  4         table_inactivity_retention,
  5         hash_algorithm
  6         --table_version
  7  from   user_blockchain_tables
  8  where  table_name like 'BCT_T%'
  9  order by 1;

TABLE_NAME  ROW_RETENTION ROW_RETENTION_LOCKED TABLE_INACTIVITY_RETENTION HASH_ALGORITHM
---------- -------------- -------------------- ---------------------------- --------------
BCT_T1                 16 NO                                            0 SHA2_512

SQL>
```

Alter Blockchain Table

```
SQL> alter table bct_t1 no drop until 100 days idle;

表已更改。

SQL> alter table bct_t1 no drop until 99 days idle;
alter table bct_t1 no drop until 99 days idle
*
第 1 行出现错误:
ORA-05732: 无法降低保留时间值


SQL> alter table bct_t1 no drop;

表已更改。
```

```
SQL> alter table bct_t1 no delete until 32 days after insert;

表已更改。

SQL> alter table bct_t1 no delete until 16 days after insert;
alter table bct_t1 no delete until 16 days after insert
*
第 1 行出现错误:
ORA-05732: 无法降低保留时间值


SQL> alter table bct_t1 no delete;

表已更改。
```

DML 和 DDL 操作

```
SQL> insert into bct_t1 (id, fruit, quantity, created_date ) values (1, 'apple', 20, sysdate);

已创建 1 行。

SQL> commit;

提交完成。

SQL> update bct_t1 set quantity = 10 where id = 1;
update bct_t1 set quantity = 10 where id = 1
       *
第 1 行出现错误:
ORA-05715: operation not allowed on the blockchain or immutable table


SQL> delete from bct_t1 where id = 1;
delete from bct_t1 where id = 1
            *
第 1 行出现错误:
ORA-05715: operation not allowed on the blockchain or immutable table


SQL> truncate table bct_t1;
truncate table bct_t1
               *
第 1 行出现错误:
ORA-05715: operation not allowed on the blockchain or immutable table
```

改变表结构

```
SQL> alter table bct_t1 modify (fruit varchar2(25));

表已更改。

SQL> alter table bct_t1 add (additional_info varchar2(50));
alter table bct_t1 add (additional_info varchar2(50))
*
第 1 行出现错误:
ORA-05715: operation not allowed on the blockchain or immutable table


SQL> alter table bct_t1 drop column quantity;
alter table bct_t1 drop column quantity
*
第 1 行出现错误:
ORA-05715: operation not allowed on the blockchain or immutable table
```

DBMS_BLOCKCHAIN_TABLE包

```
SQL> set serveroutput on
SQL> declare
  2    l_rows  number;
  3  begin
  4   dbms_blockchain_table.delete_expired_rows(
  5      schema_name             => 'testuser1',
  6      table_name              => 'bct_t1',
  7      before_timestamp        => null,
  8      number_of_rows_deleted => l_rows);
  9
 10    dbms_output.put_line('Rows Deleted=' || l_rows);
 11  end;
 12  /
Rows Deleted=0

PL/SQL 过程已成功完成。

SQL>
```

```
SQL> set serveroutput on
SQL> declare
  2    l_rows  number;
  3  begin
  4    dbms_blockchain_table.delete_expired_rows(
  5      schema_name             => 'testuser1',
  6      table_name              => 'bct_t1',
  7      before_timestamp        => systimestamp - 60,
  8      number_of_rows_deleted => l_rows);
  9
 10    dbms_output.put_line('Rows Deleted=' || l_rows);
 11  end;
 12  /
Rows Deleted=0

PL/SQL 过程已成功完成。
```

数据验证

```
SQL> set serveroutput on
SQL> declare
  2    l_rows      number;
  3    l_verified  number;
  4  begin
  5    select count(*)
  6    into   l_rows
  7    from   testuser1.bct_t1;
  8
  9    dbms_blockchain_table.verify_rows(
 10      schema_name              => 'testuser1',
 11      table_name               => 'bct_t1',
 12      number_of_rows_verified => l_verified);
 13
 14    dbms_output.put_line('Rows=' || l_rows || '  Verified Rows=' || l_verified);
 15  end;
 16  /
Rows=1  Verified Rows=1

PL/SQL 过程已成功完成。
```

Oracle原生区块链表

由受信任的提供商管理的安全分类帐表，以防止欺诈

- **区块链表是将行组织成多个链的仅插入表。链中的每一行（第一行除外）都使用加密哈希链接到链中的前一行。**
- 区块链表中的行是防篡改的
  - 每行包含一个加密哈希值，该值基于该行中的数据和链中上一行的哈希值
  - 如果某行被篡改，则该行的哈希值会更改，这会导致链中下一行的哈希值发生更改
  - **在插入时，每一行的时间戳也会被记录下来。**
  - 为了增强欺诈保护，可以在一行中添加可选的用户签名
- 用于实施集中式区块链应用程序
  - 区块链表可以进行索引和分区
  - 可以控制是否以及何时从区块链表中删除行
  - 区块链表可以与（常规）表一起用于事务和查询

Oracle区块链表应用场景

政府

通过保护执法证据链并存储政府，公民和商业数据，增强对政府的信任并减少欺诈和腐败

零售和供应链

通过跟踪库存和位置，装运和退货历史记录，信用和退款以及召回信息来提高消费者信任度

公司财务

通过维护公司间发票/ PO分类帐，符合SOX-404的交易和付款历史记录来减少对帐需求

法律服务

保护用于审判，文件公证，财产记录，托管的证据链的完整性

ISV

通过维护客户组织的防篡改交易历史记录，确保数字认证（ISO等）流程的每个步骤的真实性和合规性

金融

通过在区块链表中存储帐户头寸，交易记录，付款和资金转账，保护财务分类帐免受黑客获取DBA凭证的攻击

更多资源:

Blockchain Blog
https://blogs.oracle.com/blockchain/

Blockchain Hands-on-Lab
https://apexapps.oracle.com/pls/apex/dbpm/r/livelabs

Blockchain Documentation
https://docs.oracle.com/en/database/oracle/oracle-database/21/nfcon/oracle-blockchain-table-268779556.html

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

400-699-8888

* 活动最终解释权归甲骨文公司所有

数据库和云讲座群

20-21

甲骨文云技术公众号

技术专家1V1深入交流


ORACLE