Immutable Tables in Oracle Database 19c and 21c
(https://oracle-base.com/articles/21c/immutable-tables-21c)

An immutable table is a tamper-proof, insert-only table with an associated table-level and row-level retention period. They are similar to blockchain tables, but the rows are not chained using cryptographic hashes.

Blockchain tables were introduced in 21c and backported to 19.10. Immutable tables were introduced to Oracle 21.3 and 19.11 at the same time, so it could be considered a 19c and 21c new feature.

When learning about immutable tables, be careful not to set excessively long retention periods, or you will have to w ait a long time to drop your test tables.

Immutable Tables in Oracle Database 19c and 21c
Blockchain Tables in Oracle Database 21c
Blockchain Tables in Oracle Database 21c

Prerequisites
The COMPATIBLE parameter must be set to the correct value before we can use an immutable table.

```
conn / as sysdba

# 19c
alter system set compatible='19.11.0' scope=spfile;

# 21c
alter system set compatible='21.0.0' scope=spfile;

shutdown immediate;
startup;
```
Oracle recommend caution when increasing the COMPATIBLE parameter. If in doubt, don't touch it.

Create an Immutable Table
In addition to adding the IMMUTABLE keyword to the CREATE TABLE command, there are two immutable claus es.

The NO DROP clause determines how long the table is protected from being dropped. If the table has no rows it can still be dropped. Unlike the initial releases of blockchain tables, the NO DROP clause also prevents the table being d ropped via a DROP USER ... CASCADE command.

```
NO DROP [ UNTIL number DAYS IDLE ]
```
NO DROP : The table can't be dropped. Be careful about using this setting during testing.
NO DROP UNTIL number DAYS IDLE : The table can't dropped until there have been no new rows inserted for the specified number of days. You may prefer to use 0 or 1 as the number of days during testing this functionality.
The NO DELETE clause determines the retention period. How long each row will be protected from deletion.

```
NO DELETE { [ LOCKED ] | (UNTIL number DAYS AFTER INSERT [ LOCKED ]) }
```
NO DELETE : Each row is retained forever. The absence of the LOCKED keyword implies the setting can be chang

ed with the ALTER TABLE command, but it can't. Retention periods can only be increased.
NO DELETE LOCKED : Same as NO DELETE.
NO DELETE UNTIL number DAYS AFTER INSERT : Each row is protected from deletion for the specified numb er of days, but this setting can be increased using the ALTER TABLE command. Minimum 16 days.
NO DELETE UNTIL number DAYS AFTER INSERT LOCKED : Each row is protected from deletion for the speci fied number of days, and this setting can't be changed using the ALTER TABLE command. Minimum 16 days.
Putting it all together gives us something like the following.

```
--drop table it_t1 purge;

create immutable table it_t1 (
  id           number,
  fruit        varchar2(20),
  quantity     number,
  created_date  date,
  constraint it_t1_pk primary key (id)
)
no drop until 0 days idle
no delete until 16 days after insert;
```

Checking the USER_TAB_COLS view shows us several invisible columns have been added to our column list. The hidden columns are the same as those of a blockchain table, but unlike blockchain tables, only the ORABCTAB_CR EATION_TIME$ and ORABCTAB_USER_NUMBER$ columns are populated. The rest of the columns are set to n ull. The hidden columns are described here.

```
set linesize 120 pagesize 50
column column_name format a30
column data_type format a27
column hidden_column format a13

select internal_column_id,
       column_name,
       data_type,
       data_length,
       hidden_column
FROM   user_tab_cols
WHERE  table_name = 'IT_T1'
ORDER BY internal_column_id;

INTERNAL_COLUMN_ID COLUMN_NAME                    DATA_TYPE                   DATA_LENGTH HIDDEN_
COLUMN
------------------ ------------------------------ --------------------------- ----------- -------------
                 1 ID                             NUMBER                               22 NO
                 2 FRUIT                          VARCHAR2                             25 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
```

```
14 rows selected.

SQL>
```
The {CDB|DBA|ALL|USER}_IMMUTABLE_TABLES views display information about immutable tables. It's a vi ew over the SYS.IMMUTABLE_TABLE$ table.

```
column row_retention format a13
column row_retention_locked format a20
column table_inactivity_retention format a26

SELECT row_retention,
       row_retention_locked,
       table_inactivity_retention
FROM   user_immutable_tables
WHERE  table_name = 'IT_T1';

ROW_RETENTION ROW_RETENTION_LOCKED TABLE_INACTIVITY_RETENTION
------------- -------------------- --------------------------
           16 NO                                            0

SQL>
```
Alter an Immutable Table
The documentation suggests the NO DROP clause can be altered using the ALTER TABLE command, as long as th e retention period is not reduced. At the time of writing this doesn't seem to work in 19c for tables that were initially created with NO DROP UNTIL 0 DAYS IDLE, as all values of days return an error. We currently have a retention p eriod of 0 days for the table. In the following example we try to change it to 100 days, which gives an error. The co mmand is syntactically correct, so I assume this is a bug in this release update. This command now works in 19.12 a nd 21.3.

```
alter table it_t1 no drop until 100 days idle;

Error report -
ORA-05732: retention value cannot be lowered

SQL>
```
This command will work on tables created with NO DROP UNTIL 1 DAYS IDLE or higher.

Regardless of the current drop delay setting, an attempt to switch to the maximum value of NO DROP causes an OR A-00600 error in 19c. This is still broken in 19.12, but this command does work properly on 21.3.

```
alter table it_t1 no drop;

Error starting at line : 1 in command -
alter table it_t1 no drop
Error report -
ORA-00600: internal error code, arguments: [atbbctable_1], [0], [], [], [], [], [], [], [], [], [], []
```
This is a problem, as I would expect most people to want to play it safe by starting with a zero day delay, then uppin g the value later once they are happy with their setup. Starting on day one with a NO DROP seems very risky, as the only way to remove the table is to drop the whole schema.

Assuming it was not defined as locked, the NO DELETE clause can be modified using the ALTER TABLE comman d, as long as the retention period is not reduced. We currently have a row retention period of 16 days. In the example below we increase that value to 32. When we subsequently attempt to lower the value to 16 it gives an error.

```
-- Increase to 32 days.
alter table it_t1 no delete until 32 days after insert;

Table IT_T1 altered.

SQL>


-- Decrease to 16 days (fail).
alter table it_t1 no delete until 16 days after insert;

Error report -
ORA-05732: retention value cannot be lowered

SQL>
```

In the current release, attempting to set the row retention to NO DELETE, which is an increase in the retention perio d, results in an ORA-00600 error. I assume this is a bug in the current release update. This is still broken in 19.12, bu t works correctly in 21.3.

```
alter table it_t1 no delete;

Error report -
ORA-00600: internal error code, arguments: [atbbctable_1], [0], [], [], [], [], [], [], [], [], [], []
```

Blocked DML and DDL Operations

As you would expect for an insert-only table, all DML and DDL operations that would result in row data being amen ded or deleted are prevented for an immutable table.

The following example shows a successful insert, then some unsuccessful DML statements.

```
-- INSERT
insert into it_t1 (id, fruit, quantity, created_date ) values (1, 'apple', 20, sysdate);

1 row inserted.

SQL> commit;

Commit complete.

SQL>


-- UPDATE
update it_t1 set quantity = 10 where id = 1;

Error report -
SQL Error: ORA-05715: operation not allowed on the blockchain or immutable table

SQL>


-- DELETE
delete from it_t1 where id = 1;

Error report -
SQL Error: ORA-05715: operation not allowed on the blockchain or immutable table
```

```
SQL>
```

Some DDL statements that could alter the contents of the data are also prevented. Here is an example of the TRUNC ATE statement.

```
truncate table it_t1;

Error report -
ORA-05715: operation not allowed on the blockchain or immutable table

SQL>
```

Extending existing columns is fine, but adding new columns or dropping existing columns is not allowed.

```
-- Extend column.
alter table it_t1 modify (fruit varchar2(25));

Table IT_T1 altered.

SQL>


-- Add column
alter table it_t1 add (additional_info varchar2(50));

Error report -
ORA-05715: operation not allowed on the blockchain or immutable table

SQL>


-- Drop column.
alter table it_t1 drop column quantity;

Error report -
ORA-05715: operation not allowed on the blockchain or immutable table

SQL>
```

DBMS_IMMUTABLE_TABLE Package

The DBMS_IMMUTABLE_TABLE package is used for maintenance of immutable tables.

The DELETE_EXPIRED_ROWS procedure removes any rows that are beyond the retention period. They can't be re moved using a normal DELETE statement.

```
set serveroutput on
declare
  l_rows  number;
begin
  dbms_immutable_table.delete_expired_rows(
    schema_name            => 'testuser1',
    table_name             => 'it_t1',
    before_timestamp       => null,
    number_of_rows_deleted => l_rows);

  dbms_output.put_line('Rows Deleted=' || l_rows);
end;
```

```
/
Rows Deleted=0

PL/SQL procedure successfully completed.

SQL>
```

Alternatively, we can limit the deletion by date. The rows will only be deleted if they are outside the retention period , and match the date criteria.

```
set serveroutput on
declare
  l_rows  number;
begin
  dbms_immutable_table.delete_expired_rows(
    schema_name           => 'testuser1',
    table_name            => 'it_t1',
    before_timestamp      => systimestamp - 60,
    number_of_rows_deleted => l_rows);

  dbms_output.put_line('Rows Deleted=' || l_rows);
end;
/
Rows Deleted=0

PL/SQL procedure successfully completed.

SQL>
```

Considerations

There are a number of things to consider when using immutable tables.

The implementation of immutable tables feels kind-of buggy prior to 21.3. Even the 19.12 patch of 19c continues to have some of these issues. There are some features that don't work as documented, resulting in error messages that are inaccurate, or aren't trapped properly.
Immutable tables can be indexed and partitioned in the normal manner.
There are a number of general restrictions associated with blockchain tables, described here.
I guess the main question should be, why would you use an immutable table?

If you need an insert-only tamper proof table in your application generally, this could be the solution.
If you want the added security of cryptographic hashes, you might want to consider blockchain tables.
For more information see:

Managing Immutable Tables
DBMS_IMMUTABLE_TABLE
ALL_IMMUTABLE_TABLES
CREATE TABLE
Immutable Tables in Oracle Database 19c and 21c
Blockchain Tables in Oracle Database 21c
Blockchain Tables in Oracle Database 21c
Hope this helps. Regards Tim...