database, nor does it describe how data is stored externally. Rather, external table metadata describes how the external table layer must *present* data to the database.

A `CREATE TABLE ... ORGANIZATION EXTERNAL` statement has two parts. The external table definition describes the column types. This definition is like a view that enables SQL to query external data without loading it into the database. The second part of the statement maps the external data to the columns.

External tables are read-only unless created with `CREATE TABLE AS SELECT` with the `ORACLE_DATAPUMP` access driver. Restrictions for external tables include no support for indexed columns and column objects.

See Also:

- *Oracle Database Utilities* to learn about external tables
- *Oracle Database Administrator's Guide* to learn about managing external tables, external connections, and directory objects
- *Oracle Database SQL Language Reference* for information about creating and querying external tables

Overview of Blockchain Tables

21 A **blockchain table** is an append-only table designed for centralized blockchain applications.

In Oracle Blockchain Table, peers are database users who trust the database to maintain a tamper-resistant ledger. The ledger is implemented as a blockchain table, which is defined and managed by the application. Existing applications can protect against fraud without requiring a new infrastructure or programming model. Although transaction throughput is lower than for a standard table, performance for a blockchain table is better than for a decentralized blockchain.

A blockchain table is append-only because the only permitted DML are `INSERT` commands. The table disallows `UPDATE`, `DELETE`, `MERGE`, `TRUNCATE`, and direct-path loads. Database transactions can span blockchain tables and standard tables. For example, a single transaction can insert rows into a standard table and two different blockchain tables.

Row Chains

In a blockchain table, a **row chain** is a series of rows linked together with a hashing scheme.

A row chain is identified by unique combination of database instance ID and chain ID. A row in a blockchain table belongs to exactly one row chain. A single table supports multiple row chains.

Note:

A chained row in a standard table is orthogonal to a row chain in a blockchain table. Only the word "chain" is the same.

Every row in a chain has a unique sequence number. The database sequences the rows using an SHA2-512 hash computation on the rows of each chain. The hash for every inserted row is derived from the hash value of the previously inserted row in the chain and the **row content** of the inserted row.

Row Content

21 The **row content** is a contiguous sequence of bytes containing the column data of the row and the hash value of the previous row in the chain.

When you create a blockchain table, the database creates several hidden columns. For example, you might create the blockchain table `bank_ledger` with the columns `bank` and `deposit`:

```
CREATE BLOCKCHAIN TABLE bank_ledger (bank VARCHAR2 (128), deposit
NUMBER)
  NO DROP UNTIL 31 DAYS IDLE
  NO DELETE UNTIL 31 DAYS AFTER INSERT
  HASHING USING "SHA2_512" VERSION "v1";
```

The database automatically creates hidden columns with the prefix `ORABCTAB`: `ORABCTAB_INST_ID$`, `ORABCTAB_CHAIN_ID$`, `ORABCTAB_SEQ_NUM$`, and others. These hidden columns, which you cannot alter or manage, implement the anti-tampering algorithm. This algorithm avoids deadlocks by acquiring unique, table-level locks in a specific order at commit time. 21

Note:

Row content for blockchain tables is stored in standard data blocks. In this release of Oracle Database, blockchain tables do not support table clusters.

The instance ID, chain ID, and sequence number uniquely identify a row. Each row has a platform-independent SHA2-512 hash that is stored in hidden column `ORABCTAB_HASH$`. The hash is based on the content of the inserted row and the hash of the previous row in the chain.

The data format for the column value of a row consists of bytes from the column metadata and content. The column metadata is a 20-byte structure that describes characteristics such as position in the table, data type, null status, and byte length. The column content is the set of bytes representing the value in a row. For example, the ASCII representation of the value `Chase` is `43 68 61 73 65`. You can use the `DUMP` function in SQL to obtain both column metadata and content.