ICS 35.240
CCS L70

团　体　标　准

T/CESA XXXX—202X

区块链和分布式记账技术 面向智能合约的数据处理语言技术要求

Blockchain and distributed ledger technology —— Technical requirements of data processing language for smart contract

（征求意见稿）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

已授权的专利证明材料为专利证书复印件或扉页，已公开但尚未授权的专利申请证明材料为专利公开通知书复印件或扉页，未公开的专利申请的证明材料为专利申请号和申请日期。

202X-XX- XX 发布　　202X-XX- XX 实施

中国电子工业标准化技术协会　发布

目 次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由蚂蚁区块链科技（上海）有限公司提出。

本文件由中国电子工业标准化技术协会区块链工作委员会归口。

本文件起草单位：　。

本文件主要起草人：　。

区块链和分布式记账技术 面向智能合约的数据处理语言技术要求

1 范围

本文件给出了面向区块链智能合约数据处理的统一底层数据模型和数据处理语言要求。
本文件适用于指导区块链平台开发者在区块链平台上进行面向智能合约的便捷开发。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

世界状态 world state
用以描述在某一个时间点上，区块链系统内所有账户相关信息的完整视图。
注：“相关信息”包括账户活跃状态、账户持有人、账户余额等。

3.2

当前世界状态 current world state
当前时刻（区块链最新块高）的世界状态。

3.3

历史世界状态 historical world state
世界状态的所有历史版本。
注：同一个状态数据会有多个版本，按照块高顺序组织。

3.4

数据表 data table
保存区块链数据的网格虚拟表。

3.5

系统表 system table
区块链系统中原始的、不可更改的，用于保存区块数据的虚拟表。

3.6

业务表 business table

区块链系统中，与业务相关的、可通过智能合约进行读写操作的，用于保存区块数据的虚拟表。

4 缩略语

下列缩略语适用于本文件。

BNF 巴科斯范式（Backus-Naur Form）

DDL 数据定义语言（Data Definition Language）

DLT 分布式记账技术（Distributed Ledger Technology）

DML 数据操作语言（Data Manipulation Language）

5 总体架构

面向智能合约的数据处理过程包括，对区块链系统数据重新规范整理形成数据表，建立统一的数据模型，并根据统一的数据模型定义数据处理语言。其中，数据处理语言包括数据定义语言和数据操作语言。总体架构参见图1。

数据模型包括：

a) 区块数据，按照不同数据类型分别整理为区块表，交易表和回执表；

b) 状态数据，包括当前世界状态和历史世界状态。

注1：区块表、交易表、回执表均为系统表，表属性为只读；

注2：当前状态数据和历史状态数据分别组织为多张业务表，表属性为读写。

数据处理语言包括：

a) 数据定义语言，包括新建数据表、更改数据表、删除数据表和查询数据表；

b) 数据操作语言，包括插入数据、更新数据、删除数据和查询数据。

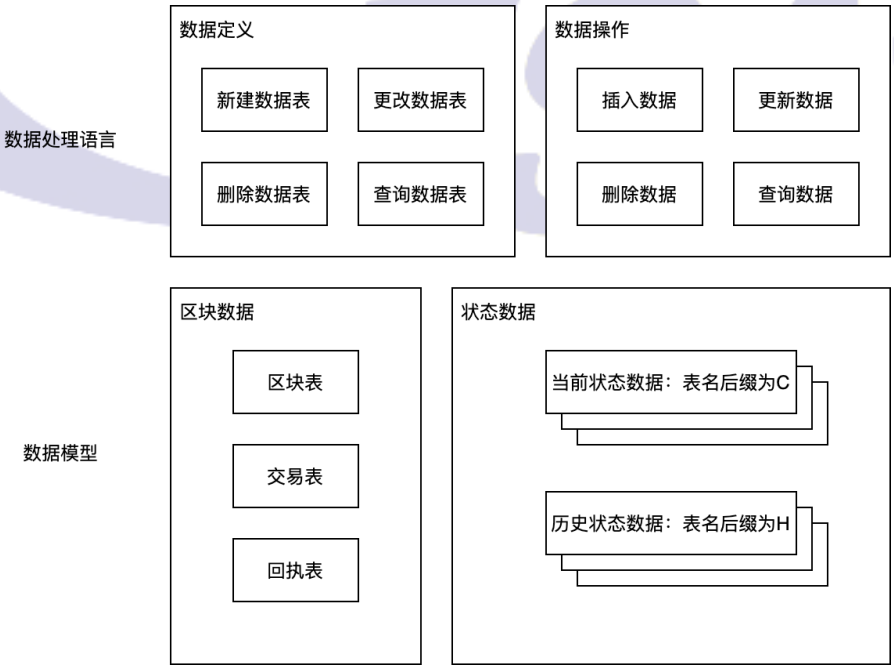


图1 总体架构图

6 统一数据模型

6.1 概述

为向智能合约提供统一的访问接口，区块链平台需提供统一的数据模型，并将区块数据、状态数据通过统一的组织方式组织为统一模型的数据，方便实现各种数据之间的交互。不同类型的区块链数据逻辑关系见图2。

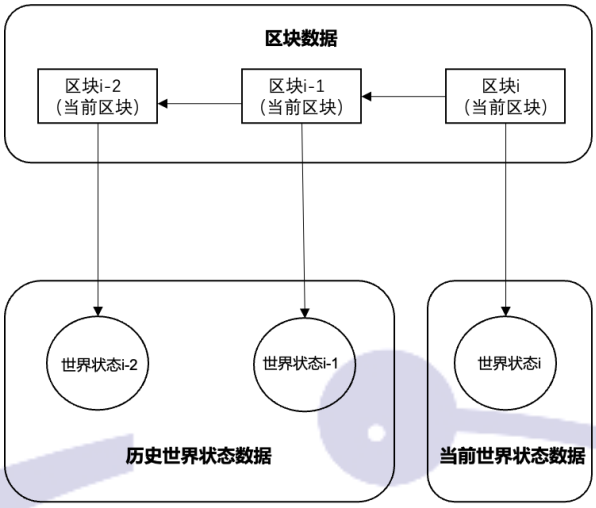


图2 不同类型区块链数据逻辑关系图

6.2 区块数据

区块数据按照不同维度，可分为区块相关的数据、交易相关的数据和回执相关的数据等。为构建统一的数据模型，区块数据采用关系数据表的模型向智能合约展示，方便用户跨表进行灵活处理。

区块相关的数据组织为一张独立的数据表，用以描述区块的元数据，一个数据行代表一个特定块高的区块。要求包括：

a) 数据表为系统表，表模式由区块链平台定义，可命名为“Blocks”；

b) 主关键字为块高，类型为无符号的 64 位整数，不应为 NULL；

c) 其他字段描述该区块的元信息，具体定义可根据区块链平台的具体设计而定。

交易相关的数据组织为一张独立的数据表，用以描述交易的元数据，一个数据行代表一个特定哈希的交易。要求包括：

a) 数据表为系统表，表模式由区块链平台定义，可命名为“Transactions”；

b) 主关键字为交易哈希，类型为 32 字节的字符串，不应为 NULL；

c) 其他字段描述该交易的元信息，具体定义可根据区块链平台的具体设计而定。

回执相关的数据组织为一张独立的数据表，用以描述交易回执数据，一个数据行代表一个特定哈希的交易回执。要求包括：

a) 数据表为系统表，表模式由区块链平台定义，可命名为“Receipts”；

b) 主关键字为交易哈希，类型为 32 字节的字符串，不应为 NULL；

c) 其他字段描述该回执的具体信息，具体定义可根据区块链平台的具体设计而定。

注：系统表是由区块区块链平台定义并且写入数据的，其模式及数据对于智能合约来说均为只读，智能合约不能新建或者修改系统表的模式，也不能对表里的数据进行写操作。

6.3 世界状态数据

为构建统一的数据模型，世界状态数据应采用关系数据表的模型向智能合约展示。关系数据表的模型包括状态数据原有的主关键字和历史版本信息，以便区分同一条数据的不同版本。

当前世界状态组织为多张数据表，用以描述业务数据的最新版本。要求包括：

a) 数据表为业务表，表模式由合约开发者通过 DDL（见 6.2）定义，宜为用户定义的表名添加后缀“C”，代表“Current”；

b) 所有业务数据列由合约开发者定义，表示当前块高的业务数据。

注 1：表模式可读可写，智能合约可对表模式进行新建、更新、删除、及查询操作；

注 2：表数据可读可写，智能合约可以对表数据进行插入、更新、删除、及查询操作。

历史世界状态组织为多张数据表，用以描述业务数据的历史版本。要求包括：

a) 数据表为业务表，表模式由合约开发者通过 DDL（见 6.2）定义，宜为用户定义的表名添加后缀“H”，代表“Historical”。

b) 所有业务数据列由合约开发者定义，表示特定历史版本的业务数据。

c) 每个表实例对应一个具体的块高，块高信息由区块链平台从当前交易的上下文信息当中自动提取，智能合约无需指定。

注 1：表模式只读，智能合约只能对表模式进行查询操作；

注 2：表数据只读，智能合约只能对表数据进行查询操作。

7 合约数据处理语言

7.1 概述

面向区块链智能合约，针对统一的数据模型，区块链系统应提供统一的数据处理语言，以便进行区块链系统进行快速、便捷的开发等操作。

本文件采用BNF范式对数据处理语言进行定义。采用以下形式组织：

a) 功能描述：对语句用途简短描述；

b) 语法格式：对语法的 BNF 范式定义；

c) 语法规则：BNF 范式无法表示的，但而该语句应当满足的附加语法限制。

此外，本文件采用BNF范式作为语法表示，具有如下的扩展：

a) 方括号（[]）表示可选元素；

b) 省略号（…）表示可重复一次或多次的元素；

c) 花括号（{}）表示元素的组合序列。

本文件采用大写和小写字母来区分常量和变量。大写字母表示字面量/常量，如CREATE TABLE代表“CREATE TABLE”字符串本身。规则名/变量使用尖括号包小写字母来表示，如<table definition>代表表定义的规则。逗号，圆括号也是字面量，代表他们自己本身。

7.2 数据定义语言

7.2.1 概述

数据定义语言实现统一数据模型中新建数据表、更改数据表、删除数据表和查询数据表的操作。以下内容给出了新建数据表、更改数据表、删除数据表和查询数据表的语句功能描述、语法格式和语法规则。

7.2.2 新建数据表

7.2.2.1 功能描述

“新建数据表”语句用于定义数据表。该操作仅针对业务表，系统表由区块链平台创建，不可更改。

7.2.2.2 语法格式

<table definition> ::= CREATE TABLE <table name> <table element list>
解释：表定义<table definition>，包含表名<table name>以及表元素清单<table element list>.

<table element list> ::= (<table element>[{,<table element>}…])
解释：表元素清单<table element list>，由一个或者多个表元素<table element>组成。

<table element> ::= <column definition>|<table constraint definition>
解释：表元素<table element>，可以是一个列定义<column definition>，或者是一个表约束定义<table constraint definition>

<column definition> ::= <column name><data type>[<default clause>][<column constraint definition>…]
解释：列定义<column definition>，应包含列名<column name>以及数据类型<default clause>，选择性包含缺省值子句<default clause>以及列约束定义<column constraint definition>，其中列约束定义可以是一个或者多个。

<data type> ::= <character string type>|<numeric type>
解释：数据类型<data type>表示当前数据列的类型，可以是字符串类型<character string type>，或者是数值类型<numeric type>.

<default clause> ::= DEFAULT {<literal>|NULL}
解释：缺省值子句<default clause>用于定义当前数据列的缺省值，该值可以是一个字面值<literal>或者NULL.

<column constraint definition> ::= NOT NULL|<unique specification>
解释：列约束定义<column constraint definition>用于描述当前数据列的相关限制条件，如非空性，唯一性等。

<unique specification> ::= UNIQUE|PRIMARY KEY
解释：唯一性说明<unique specification>用于描述索引的唯一性，可能的取值为UNIQUE（唯一性索引）或者PRIMARY KEY（主索引）。如果没有指定，缺省为非唯一性索引。

<table constraint definition> ::= [<constraint name definition>][<unique specification>](<column name list>)
解释：表约束定义<table constraint definition>用于描述当前数据表的相关限制条件，本版本仅包括索引的声明。包含可选的约束名定义<constraint name definition>，可选的唯一性说明<unique specification>，以及列名清单<column name list>。

<constraint name definition> ::= CONSTRAINT <constraint name>

解释：约束名定义<constraint name definition>，由关键字CONSTRAINT以及约束名称<constraint name>组成。

<column name list> ::= <column name>[{,<column name>}…]
解释：列名清单<column name list>由一个或者多个列名<column name>组成。

7.2.2.3 语法规则

“新建数据表”语句的语法规则包括：

a) 不同<table definition>当中的<table name>互不相同；
b) <table definition>应至少包含一个<column definition>；
c) 由<table definition>定义的表的描述，包含<table name>和由各个<column definition>指明的各列的描述，第 i 个列描述由第 i 个<columne definition>给出；
d) 在一个<table definition>里面，不同<column definition>当中的<column name>互不相同；
e) 由<column definition>定义的列的描述，包含<column name>和由<data type>指明的数据类型。

7.2.3 更改数据表

7.2.3.1 功能描述

“更改数据表”语句用于修改一个数据表的定义。该操作仅针对业务表，系统表由区块链平台创建，不允许智能合约更改。

7.2.3.2 语法格式

<alter table statement> ::= ALTER TABLE <table name> <alter table action list>
解释：表修改语句<alter table statement>，需要指定表名<table name>和表修改行为清单。

<alter table action list> ::= <alter table action>[{, <alter table action>}…]
解释：表修改行为清单<alter table action list>，由一个或多个表修改行为<alter table action>组成。

<alter table action> ::= <add column definition>|<drop column definition>|<add table constraint definition>|<drop table constraint definition>
解释：表修改行为<alter table action>，可能是增加列定义<add column definition>，删除列定义<drop column definition>，增加表约束定义<add table constraint definition>，或者删除表约束定义<drop table constraint definition>.

<add column definition> ::= ADD COLUMN <column definition>
解释：增加列定义<add column definition>，包含关键字ADD COLUMN以及列定义<column definition>本身，<column definition>的格式和规则见前面章节。

<drop column definition> ::= DROP COLUMN <column name>

解释：删除列定义<drop column definition>，包含关键字DROP COLUMN以及列名<column name>.

<add table constraint definition> ::= ADD <table constraint definition>

解释：增加表约束定义<add table constraint definition>，包含关键字ADD以及表约束定义<table constraint definition>本身。<table constraint definition>的格式和规则见前面章节。

<drop table constraint definition> ::= DROP <constraint name>

解释：删除表约束定义<drop table constraint definition>，包含关键字DROP以及表约名<constraint name>.

7.2.3.3 语法规则

“更改数据表”语句的语法规则包括：

a) 令 T 为<table name>所指定的数据表；
b) T 应是之前定义过的；
c) <column name>所指定的数据列应是 T 的一个列，但不能是唯一的列；
d) <constraint name>所指定的约束应是 T 的一个约束。

7.2.4 删除数据表

7.2.4.1 功能描述

“删除数据表”语句用于删除整个数据表。该操作仅针对业务表，系统表由区块链平台创建，不允许智能合约删除。

7.2.4.2 语法格式

<drop table statement> ::= DROP TABLE <table name>

解释：表删除语句<drop table statement>，包含关键字"DROP TABLE"以及所要删除的表名称<table name>.

7.4.2.3 语法规则

“删除数据表”语句的语法规则包括：

a) 令 T 为<table name>所指定的数据表；
b) T 应是之前定义过的。

7.2.5 查询数据表

7.2.5.1 功能描述

“查询数据表”语句用于列出所有数据表的名称。该操作对业务表和系统表均有效。

7.2.5.2 语法格式

<show tables statement> ::= SHOW TABLES

解释：表查询语句<show tables statement>只包含关键字“SHOW TABLES”。

7.3 数据操作语言

7.3.1 概述

数据操作语言能够实现统一数据模型中插入数据、更新数据、删除数据、查询数据操作。以下内容给出了插入数据、更新数据、删除数据、查询数据的语句功能描述、语法格式和语法规则。

7.3.2 插入数据

7.3.2.1 功能描述

“插入数据”语句用于在数据表中插入新的数据行集合。该操作仅针对业务表，系统表由区块链平台填充数据。

7.3.2.2 语法格式

<insert statement> ::= INSERT INTO <table name> <insert columns and source>

解释：插入语句<insert statement>，将特定的数据<insert columns and source>，插入到名为<table name>的数据表。

<insert columns and source> ::= <from constructor> | <from default>

解释：插入数据源<insert columns and source>，可能来自于构造器<from constructor>，或者是缺省值<from default>。

<from constructor> ::= [(<insert column list>)] VALUES <row value expression list>

解释：数据源构造器<from constructor>，可以指定要插入的列<insert column list>，应指定要插入的数据清单。

<insert column list> ::= <column name>[{,<column name>}…]

解释：插入列的清单<insert column list>，由一个或者多个列名<column name>组成，列名之间使用逗号分隔。

<row value expression list> ::= <row value expression>[{,<row value expression>}…]

解释：行值表达式清单<row value expression list>，由一个或者多个行值表达式<row value expression>组成。

<row value expression> ::= [ROW](<row value element>[{,<row value element>}…])

解释：行值表达式<row value expression>，由一个或者多个行值元素<row value element>组成。

<row value element> ::= <value expression>|NULL|DEFAULT

解释：行值元素<row value element>，可能是一个值表达式，字符串“NULL”，或者字符串“DEFAULT”.

<from default> ::= DEFAULT VALUES

解释：缺省数据源<from default>，由字符串“DEFAULT VALUES”表示。

7.3.2.3 语法规则

“插入数据” 语句的语法规则包括：

a) 令 T 表示<table name>标识的表，<insert column list>中的每一个<column name>应标识 T 的一个列，且同一个列不应标识多于一次。<insert column list>的省略代表一个隐式声明，该声明包含 T 中所有列的升序序列；
b) 对于任意一个<row value expression>，其中<row value element>的个数应等于<insert column list>中的<column name>的个数；
c) 如果<row value expression>的第 i 项不是 NULL 或者 DEFAULT，则其数据类型如下：
   1) 如果<insert column list>第 i 项所对应的列的数据类型是长度为 L 的字符串，则对应插入值类型应为长度小于或等于 L 的字符串；
   2) 如果<insert column list>第 i 项所对应的列的数据类型是精确数值，则对应插入值类型应为精确数值。

7.3.3 更新数据

7.3.3.1 功能描述

“更新数据”语句用于在数据表中更新指定的数据行集合。该操作仅针对业务表，系统表由区块链平台填充数据，不允许智能合约进行修改。

7.3.3.2 语法格式

<update statement> ::= UPDATE <table name> SET <set clause list>[WHERE <search condition>]

解释：更新语句<update statement>，在名为<table name>的数据表当中，针对所有满足<search condition>条件的数据行，实施赋值从句清单<set clause list>所指定的更新操作。当WHERE子句省略时，更新表中所有数据行。

<set clause list> ::= <set clause>[{,<set clause>}…]

解释：赋值从句清单<set clause list>，包含一个或多个赋值从句<set clause>，从句之间使用逗号分隔。

<set clause> ::= <column name> = <value expression>|NULL

解释：赋值从句<set clause>，将名为<column name>的列，赋值为<value expression>或者NULL。

7.3.3.3 语法规则

“更新数据”语句的语法规则包括：

a) 令 T 表示<table name>标识的表，T 不应是一个只读的表；
b) 每一个<column name>应标识 T 的一个列,相同的<column name>不应在同一个更新语句<update statement>中多次出现；
c) 对于每一个赋值从句<set clause>，应满足以下条件：
   1) 如果赋值为 NULL，那么<column name>所指定的列应允许为空值；
   2) 如果<column name>所对应的列的数据类型是长度为 L 的字符串，则对应<value expression>的类型应为长度小于或等于 L 的字符串；

3) 如果<column name>所对应的列的数据类型是精确数值，则对应<value expression>de 类型应为精确数值。

7.3.4 删除数据

7.3.4.1 功能描述

“删除数据”语句用于从数据表中删除指定的数据行集合。该操作仅针对对业务表有效，系统表由区块链平台填充数据，不允许智能合约进行删除。

7.3.4.2 语法格式

<delete statement> ::= DELETE FROM <table_name> [WHERE <search_condition>]

解释：删除语句<delete statement>，从名为<table name>的数据表中，删除所有满足条件<search condition>的数据行。

7.3.4.3 语法规则

“删除数据”语句的语法规则：令 T 表示<table name>标识的表，T 不应是一个只读的表。

7.3.5 查询数据

7.3.5.1 功能描述

“查询数据”语句用于从数据表指定的数据行中查询特定的值集合。该操作对业务表和系统表均有效。

7.3.5.2 语法格式

<select statement> ::= SELECT <select list> <table expression>

解释:查询语句<select statement>,从表达式<table expression>所指定的数据表中,获取<select list>所指定的值清单。

<select list> ::= <value expression>[{,<value expression>}…]

解释：查询清单<select list>，包含一个或多个值表达式，表达式之间使用逗号分隔。

<table expression> ::= <from clause>[<where clause>][<group by clause>]

解释:表表达式<table expression>,主要包含<from clause>,可选性包含<where clause>,<group by clause>.

<from clause> ::= FROM <table reference>[{,<table reference>}…]

解释：从句<from clause>，包含一个或多个表引用<table reference>，表引用之间使用逗号分隔。

<table reference> ::= <table name>[AS <correlation name>]

解释：表引用<table reference>，主要包含表名<table name>，可选择性定义相关名<correlation name>.

<where clause> ::= WHERE <search condition>

解释：条件从句<where clause>，主要包含搜索条件<search condition>.

<group by clause> ::= GROUP BY <column reference>[{,<column reference>}…]

解释：分组从句<group by clause>，包含一个或多个列引用<column reference>，列引用之间使用逗号分隔。

<column reference> ::= [<qualified identifier>.]<column name>

解释：列引用<column reference>，主要包含列名<column name>，可选择性包含限定符<qualified identifier>作为前缀。

7.3.5.3 语法规则

“查询数据”语句的语法规则包括：

a) 令 R 表示<table expression>的结果；
b) 查询结果的度等于<select list>的基数；
c) 每个<value expression>内的每个<column reference>应无歧义的引用 R 的一个列；
d) 查询结果的每个列与相应的<value expression>具有相同的数据类型，长度，精度，以及标度；
e) 如果<select list>中的第 i 个<value expression>由单一的<column reference>组成，则结果的第 i 列是一个命名列，它的列名就是<column reference>的<column name>；否则，该列是一个非命名列；
f) 当且仅当查询结果的一个列是命名列，并且只包含非空值，则该列限制为只包含非空值。

附 录 A
（资料性）
区块链底层数据分类

区块链底层数据可分为区块数据和状态数据，其中状态数据分为当前世界状态数据和历史世界状态数据。三种数据的逻辑关系如 A.1 所示。

A.1 区块数据

区块数据是区块链系统的核心数据。一段特定时间之内的交易和回执数据打包组成一个区块，多个区块之间通过哈希纠缠的方式组成线性的链块结构。

A.2 世界状态数据

世界状态数据是区块链系统的扩展数据，用以描述在某一个时间点上，系统内所有账户的相关信息，如账户活跃状态，账户持有人，账户余额等。当前世界状态是当前时刻（区块链最新块高）的世界状态，是区块链交易数据读写的缺省视图。

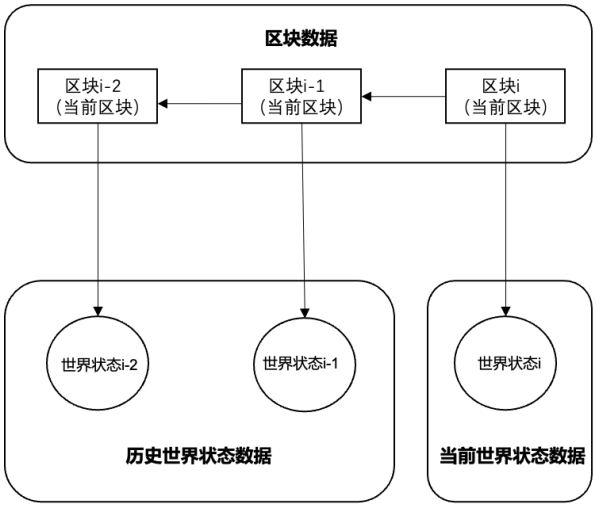
区块链具有数据历史可追踪的特点。对于同一条数据，需要在区块区块链平台层面保存其完整历史的各个版本，为了能够在数据模型里面统一处理各个版本的状态数据，数据表的每一行应具备历史版本的相关信息。

A.3 历史世界状态数据

历史世界状态包含世界状态的所有历史版本，同一个状态数据会有多个版本，按照块高顺序组织。

三种数据逻辑关系如图所示。区块数据分为多个区块，从最新的当前区块开始，每个区块通过哈希指向自己的前驱区块，整体形成一个链式结构。同时，每个区块包含对应世界状态的根哈希。其中，当前区块指向的世界状态称为当前世界状态。其他的世界状态整体称为历史世界状态。

注：不同区块对应的世界状态在物理存储上是有重叠的，这里为了简化，表示为各自独立的逻辑视图。



图A.1 三种数据逻辑关系图

附 录 B
（资料性）
数据处理语言实例

本附录提供数据操作语言的实例，以便于加深对正文的理解。

B.1 表定义

```
CREATE TABLE table 1 (
    id INT PRIMARY KEY,
    name VARCHAR(100) DEFAULT “”,
    CONSTRAINT id_name UNIQUE (id, name)
)
```

解释：定义一个名为table_1的数据表，包含两个数据列。第一列名为id，类型为整形，作为整个表的主关键字。第二列名为name，类型为字符串，缺省为空串。两个列构成一个叫作id_name的唯一性约束。

B.2 插入数据

```
INSERT INTO table_1 (column_1, column_2) VALUES ROW(1, “a”), ROW(2, “b”),ROW(3, “c”)
```

解释：向名为“table_1”的数据表插入数据，提供两个列(“column_1”和“column_2”)的插入值，其他列采用缺省值。总共插入三行数据，分别为(1, “a”), (2, “b”)和(3, “c”)。

```
INSERT INTO table_2 DEFAULT VALUES
```

解释：向名为“table_2”的数据表插入一行数据，所有列的插入值为缺省值。

参 考 文 献

[1] GB/T 12991.1-2008 信息技术 数据库语言SQL 第1部分：框架
[2] GB/T 42752-2023 区块链和分布式记账技术 参考架构