

# MySQL8.0を使って ブロックチェーンを 実装する

第9回 関西DB勉強会

@masayuki14

# 目次



- ✓ 自己紹介
- ✓ 概要
- ✓ 技術要素の解説
- ✓ デモ
- ✓ MySQL8.0の新機能

# 目次



# 自己紹介

# 自己紹介

✓ もりさきまさゆき

✓ @masayuki14



**Follow me !!**

# 自己紹介



## ✓ 主夫

✓ 妻1人 (フルタイム) 幼児2人

✓ フリーランス (Web系パートタイム)

## ✓ コミュニティ

✓ はんなりPython (第3金曜開催@京都)

✓ OSS Gate (11/24 ワークショップ@大阪)

## ✓ スプーキーズアンバサダー

# スプーキーズ@京都



- ✓ Web系システム
- ✓ ソーシャルゲーム
- ✓ 中規模DB運用
  - ✓ MySQL, PerconaServer, MariaDB



**Webエンジニア積極採用中！！**

# スプーキーズ@京都



勉強会 テクテクテック

✓ picoCTF2018 問題の解説会

✓ 2018/11/14 (水)

✓ DB勉強会 運営でのあれこれ

✓ 2019/01/16 (水)



# 目次



# 概要



# 概要



“

MySQL8.0を使って  
簡略化したブロックチェーンを  
実装した

”

# 概要



## やったこと

- ✓ ブロックチェーンをテーブルで表現
- ✓ MySQL Shell スクリプト実装
  - ✓ ブロック追加などのデータ操作
  - ✓ マイニング処理

# 目次



## 技術要素の解説

# 技術要素の解説



使っているもの

- ✓ MySQL8.0
- ✓ MySQL Shell
- ✓ JavaScript

# 技術要素の解説



## MySQL8.0

- ✓ ブロックのモデルをテーブルで表現
- ✓ 取引データはJSON

# 技術要素の解説



## MySQL Shell

- ✓ MySQLのCLIクライアント
- ✓ DBアクセスのインタフェース
- ✓ JavaScriptでDBを操作が可能

# 技術要素の解説



## JavaScript

- ✓ テーブル操作
- ✓ マイニング処理
- ✓ MySQL Shellで実行

# ブロックチェーン



- ✓ 分散型ネットワーク
- ✓ 合意形成
- ✓ スマートコントラクト

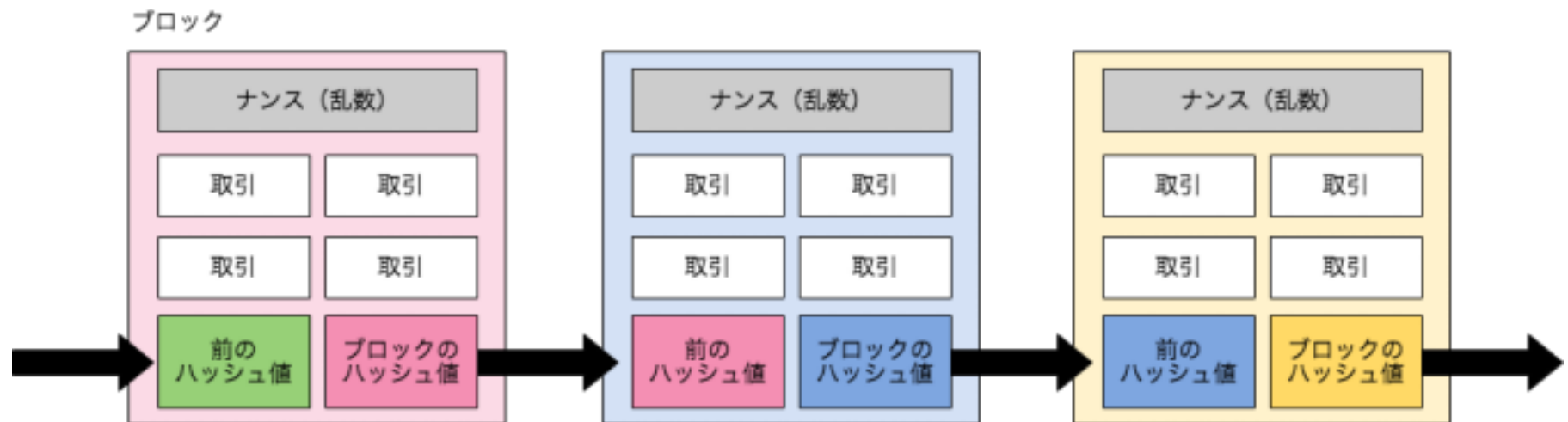
**データ構造だけにフォーカス！**



# ブロックチェーン



## ブロックチェーンの基本的な構造



# ブロックチェーン



## 構成要素

- ✓ ナンス(乱数)
- ✓ 取引(トランザクション)
- ✓ 前のブロックのHash値

# マイニングとは



- ✓ ブロックのHash値に制約
- ✓ 制約を満たすナンス(乱数)を探すこと

# マイニングとは



## Hash値の先頭に0を10個連続させる


```
value = hash( nonce, transaction, prev_hash )
```

```
# => 00000000004b2a76b9719d911017c592
```

```
# md5 32桁
```

## ナンス を探せ！

# 実行環境



## MySQLでブロックチェーンを表現

---

- ✓ ブロック -> Record
- ✓ ブロックチェーン -> Table
- ✓ Hash関数 -> MD5
- ✓ Hash制約 -> 先頭に0が4つ連続

# 実行環境



```
CREATE TABLE `block` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `transaction` json DEFAULT NULL,  
  `nonce` int(11) DEFAULT NULL,  
  `prev_hash` varchar(32) DEFAULT NULL,  
  `hash` varchar(32) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB
```

# 実行環境



## 例えばこんな Transaction

```
[  
  {"name": "miyake", "date": "2018-04-01",  
    "report": "今日は良い天気でした。"},  
  {"name": "kataoka", "date": "2018-04-01",  
    "report": "チョコレートが美味しかった。"},  
  {"name": "tamamura", "date": "2018-04-01",  
    "report": "電車が遅れて最悪だった。"}  
]
```

# 目次



# デモ



# デモ



```
$ mysqlsh -u root -p
```

# デモ



// テーブル取得

```
table = session  
    .getSchema('blockchain')  
    .getTable('block')
```

// ヘルプが表示される

```
table.help()  
// まだデータがないのでEmpty  
table.select()
```

// オブジェクトでInsertできる

```
table.insert({  
    transaction: [{'name': 'morisaki'}],  
    prev_hash: ''  
})
```

# デモ

*// INSERTしたデータが参照できる*  
table.select()

*// where句の指定など*  
record = table.select()  
    .where('id = :id')  
    .bind('id', 1)  
    .execute().fetchOne()

# デモ

```
// SQLモードに切り替え  
\sql
```

```
use blockchain;  
select * From block;  
show create table block;
```

# マイニング処理



```
mining = function (id) {  
  
    schema = session.getSchema('blockchain')  
    table = schema.getTable('block')  
    record = table.select().where('id = :id')  
        .bind('id', id).execute().fetchOne()  
  
    nonce = 0  
    while (true) {  
        hash = md5(record.transaction + nonce + record.prev_hash)  
        // 条件に一致するHash値になれば終了  
        if (hash.match(/^0000/)) {  
            table.update()  
                .set('hash', hash).set('nonce', nonce)  
                .where('id = :id').bind('id', record.id)  
                .execute()  
            break  
        }  
        nonce++  
    }  
}
```

# デモ



```
mining(1)
```

```
table.select()
```

```
record = table.select().where('id = 1')  
    .execute().fetchOne()
```

```
record.hash
```

```
record.getField('nonce')
```

```
JSON.parse(record.transaction)
```

# デモ



```
id2hash = function (id) {  
  schema = session.getSchema('blockchain')  
  table = schema.getTable('block')  
  record = table.select()  
    .where('id = :id').bind('id', id)  
    .execute().fetchOne()  
  return record.hash  
}  
  
format = function(prevId, transaction) {  
  return {  
    prev_hash: id2hash(prevId),  
    transaction: transaction  
  }  
}
```

# デモ



*// id => hash 指定IDのHash値を返す*

```
id2hash(1)
```

```
format(1, [{"name": "mysql"}, {"name": "masayuki", "date": "2018-10-14"}])
```

```
table.insert(format(1, [{"name": "yoshida", "date": "2018-10-15"}]))
```

```
mining(2)
```

```
table.insert(format(2, [{"name": "tanaka", "date": "2018-10-16"}]))
```

```
mining(3)
```

```
table.insert(format(3, [{"name": "okamura", "date": "2018-10-17"}]))
```

```
mining(4)
```



# ファイルの実行



-f, --file=file オプションで実行

```
$ mysqlsh -u root -p -f batch.js
```

# CTE 共通テーブル式



- ✓ MySQL8.0でCTEをサポート
- ✓ 再帰的にJOINが実行できる

# CTE 共通テーブル式



```
with recursive
blockchain (id, hash, path) as (
  select id, hash, cast(id as char(100))
  from block
  where id = 1
union all
  select b.id, b.hash, concat(bc.path, '->', b.id)
  from blockchain bc join block b
  on bc.hash = b.prev_hash
)
select * from blockchain;
```

# CTE 共通テーブル式



id	hash	path
1	00003e18bd64f8ca6a962cb960b65f77	1
2	0000ec3e66368ca655715136995d1701	1->2
3	00008c18fc3f5e38ba0319ce27525a22	1->2->3
4	00000015c2b4d2395443e5eb94245794	1->2->3->4

# JSON\_TABLE()



- ✓ MySQL8.0で新規追加
- ✓ JSONを表に変換できる
- ✓ 他のテーブルとJOINできる

# JSON\_TABLE()



transaction を見やすくする

```
select id, trans.*
from block, JSON_TABLE(
    `transaction`,
    '$[*]'
    columns (
        `name` varchar(32) path '$.name',
        `date` date path '$.date',
        `report` varchar(128) path '$.report'
    )
) trans;
```

# JSON\_TABLE()



id	name	date	report
1	miyake	2018-04-01	今日は良い天気でした。
1	kataoka	2018-04-01	チョコレートが美味しかった。
1	tamamura	2018-04-01	電車が遅れて最悪だった。
2	miyake	2018-04-02	夕立がありすごい雨でした。
2	kataoka	2018-04-02	ドーナツならチョコレートがかかっている欲しい。
2	tamamura	2018-04-02	前を歩く人の傘が刺さりそうで腹がたった。
3	miyake	2018-04-03	月が綺麗ですね。
3	kataoka	2018-04-03	紅茶ならダージリンが好みだ。チョコレートを添えて。
3	tamamura	2018-04-03	前を歩く学生が歩道に広がって邪魔だった。
3	kawai	2018-04-03	ストレッチをしたら気持ちよく寝れました。
4	miyake	2018-04-04	セミが鳴いていたので夏を見つけた気分です。
4	kataoka	2018-04-04	シフォンケーキにはチョコクリームがとても合う。
4	tamamura	2018-04-04	電車の中でハンバーガー食うなよ。臭うだろ。
4	kawai	2018-04-04	はじめてランニングハイを感じられました。

# JSON\_TABLE()



```
with trans (id, name, date, report)
as (
  select id, t.*
  from block, JSON_TABLE(
    `transaction`,
    '$[*]'
    columns (
      `name` varchar(32) path '$.name',
      `date` date path '$.date',
      `report` varchar(128) path '$.report'
    )
  ) t
)
select * from trans join block using(id);
```



# まとめ



- ✓ MySQLShellでDBプログラミング
  - ✓ JavaScript
- ✓ MySQL8.0新機能が便利
  - ✓ CTE
  - ✓ JSON\_TABLE()

# 参考文献

- ✓ MySQL8.0 公式ドキュメント
- ✓ MySQLセッションスライド
  - ✓ <https://www.mysql.com/jp/news-and-events/seminar/downloads.html>
- ✓ 詳解MySQL 5.7 (書籍)