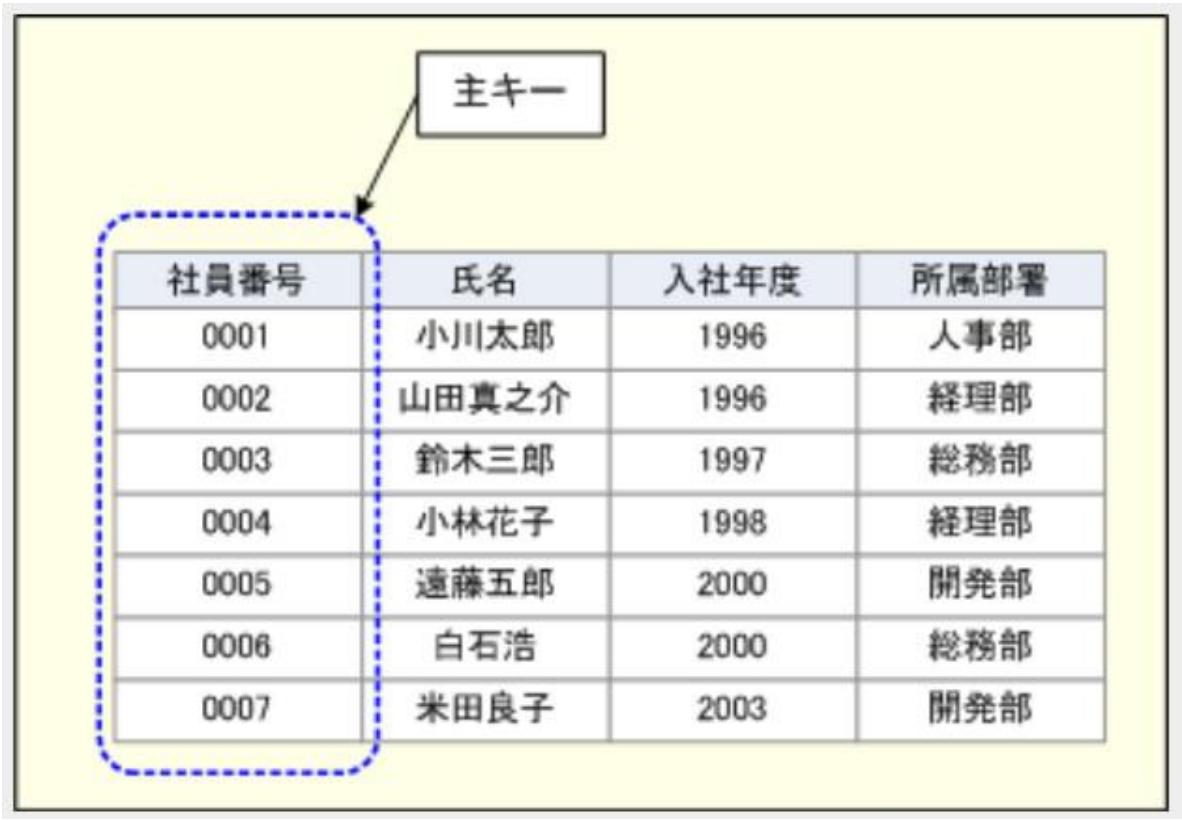
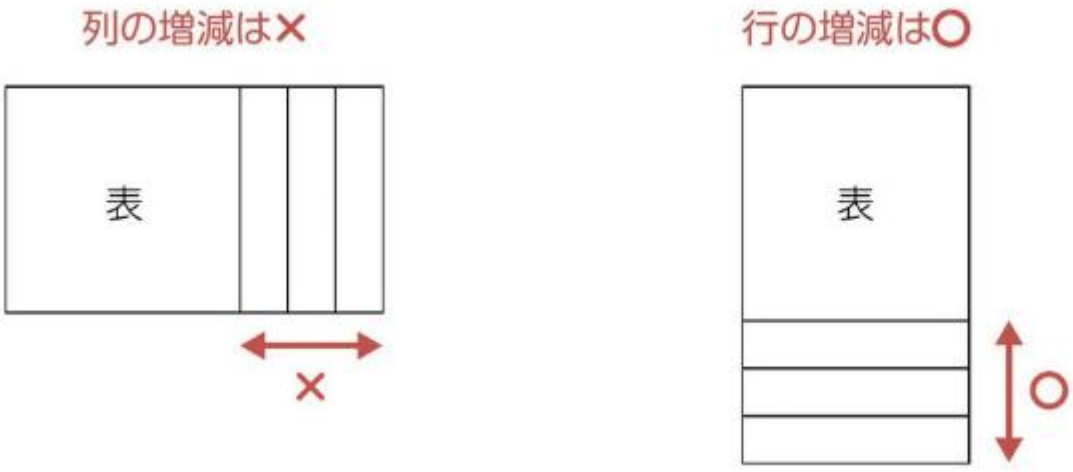
-
-
-
-


MariaDB®
Vs
MySQL
Vs
PostgreSQL

-
-
-
-

-


主キー
社員番号
氏名
入社年度
所属部署
0001
小川太郎
1996
人事部
0002
山田真之介
1996
経理部
0003
鈴木三郎
1997
総務部
0004
小林花子
1998
経理部
0005
遠藤五郎
2000
開発部
0006
白石浩
2000
総務部
0007
米田良子
2003
開発部


列の増減は×
表
×
行の増減は○
表
○

繰返し要素
（行が結合している部分）

非正規形

受注No[*4]	日付	顧客ID	顧客名	商品ID	商品名	数量	単価
1001	8/30	002	B社	A	鉛筆	50	100
				B	定規	20	50
1002	3/7	001	A社	A	鉛筆	20	100
				C	ペン	10	200

第１正規形

受注No	日付	顧客ID	顧客名	商品ID	商品名	数量	単価
1001	8/30	002	B社	A	鉛筆	50	100
1001	8/30	002	B社	B	定規	20	50
1002	3/7	001	A社	A	鉛筆	20	100
1002	3/7	001	A社	C	ペン	10	200

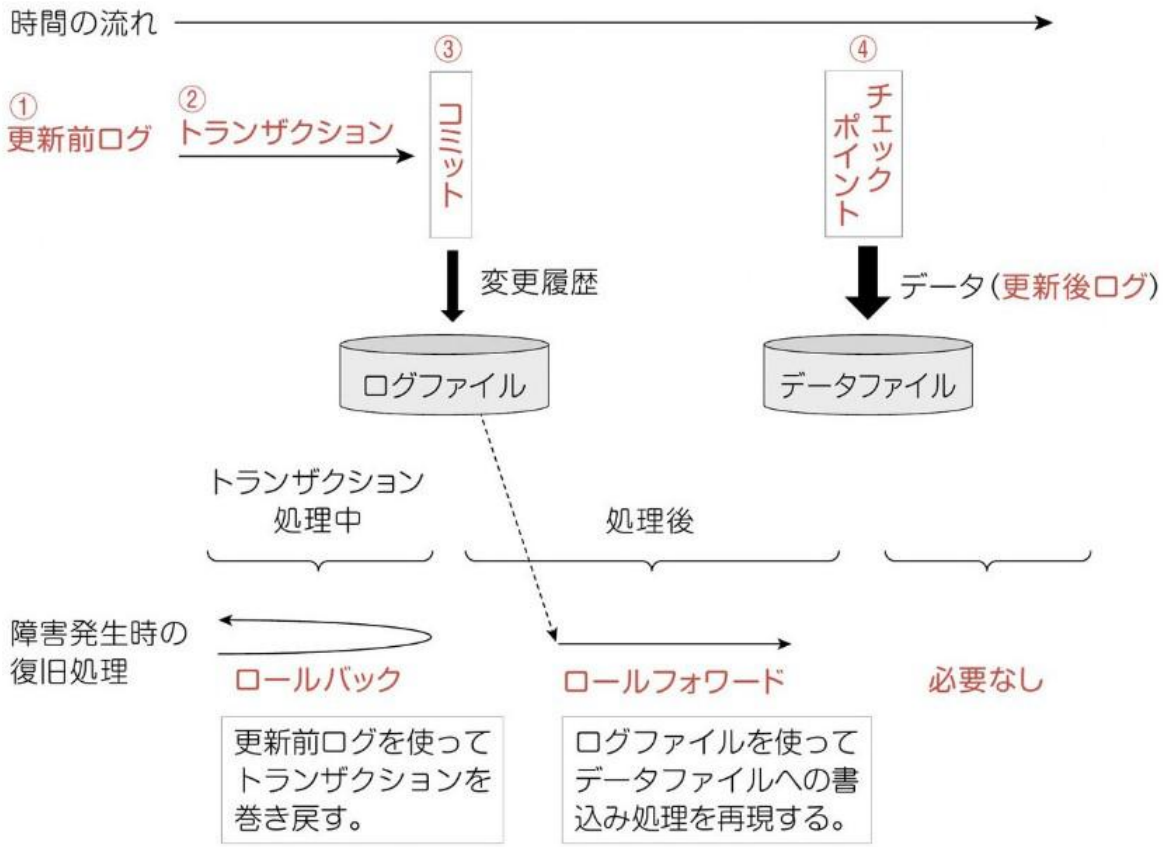
第２正規形

受注No	日付	顧客ID	顧客名
1001	8/30	002	B社
1002	3/7	001	A社

商品ID	商品名	単価
A	鉛筆	100
B	定規	50
C	ペン	200

受注No	商品ID	数量
1001	A	50
1001	B	20
1002	A	20
1002	C	10

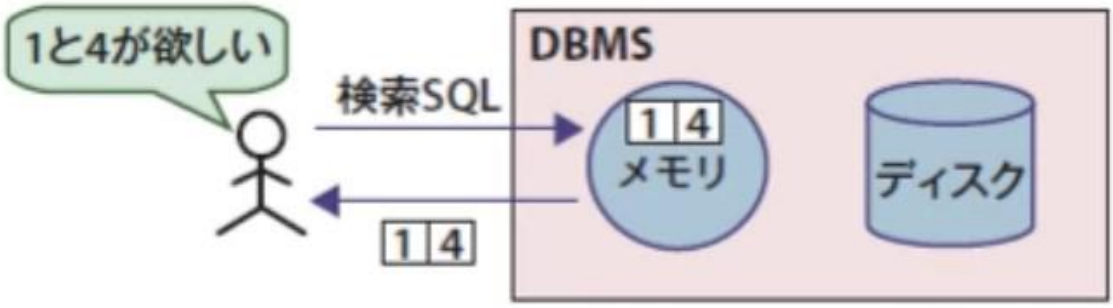
-
-
-
-



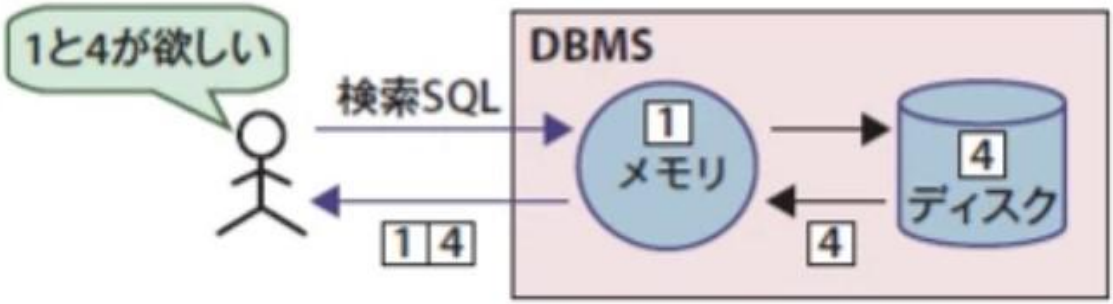

```php
try{
    // データベースと接続。
    $db = getDb();

    // 例外処理を有効化。
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // トランザクションを開始。
    $db->beginTransaction();
    // いくつかのSQLが実行される。※もし失敗した場合、ERRMODE_EXCEPTIONを実行。
    $db->exec("INSERT INTO movie(title, price) VALUES('ハリポタ', 2000)")
    $db->exec("INSERT INTO movie(title, price) VALUES('シスター', 2000)")

    // トランザクション内の一連のステートメントが成功したら、トランザクションをコミット。
    $db->commit();

} catch{
    // 例外が発生したらロールバックし、エラーメッセージを出力。
    $db->rollBack();
    print "失敗しました。:{$e->getMessage()}"
}
```

-

•

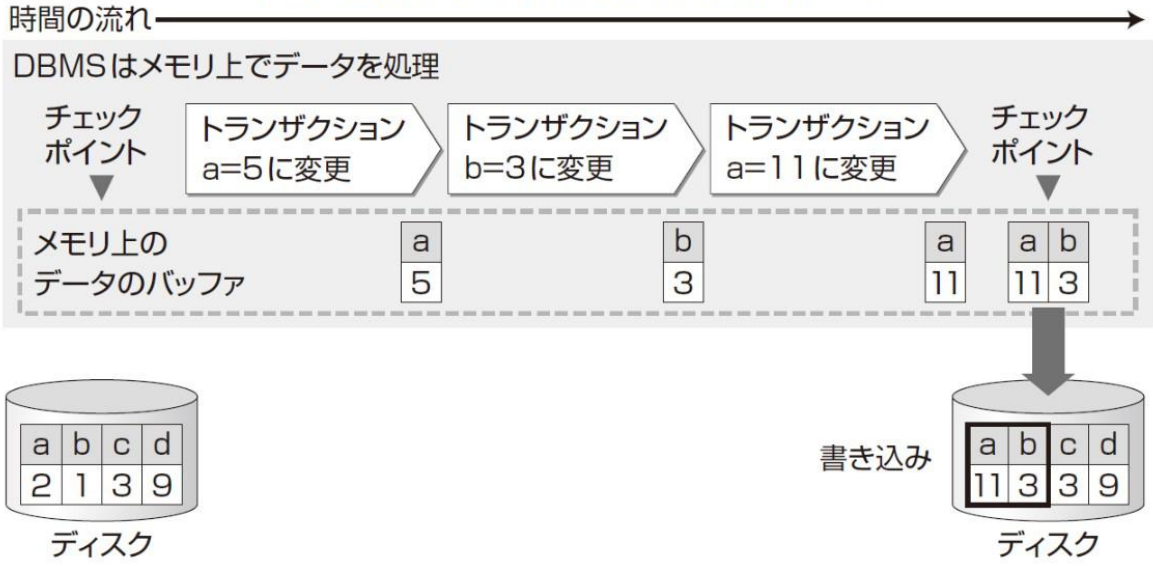


メモリに存在するデータだけで結果が返せると非常に速い

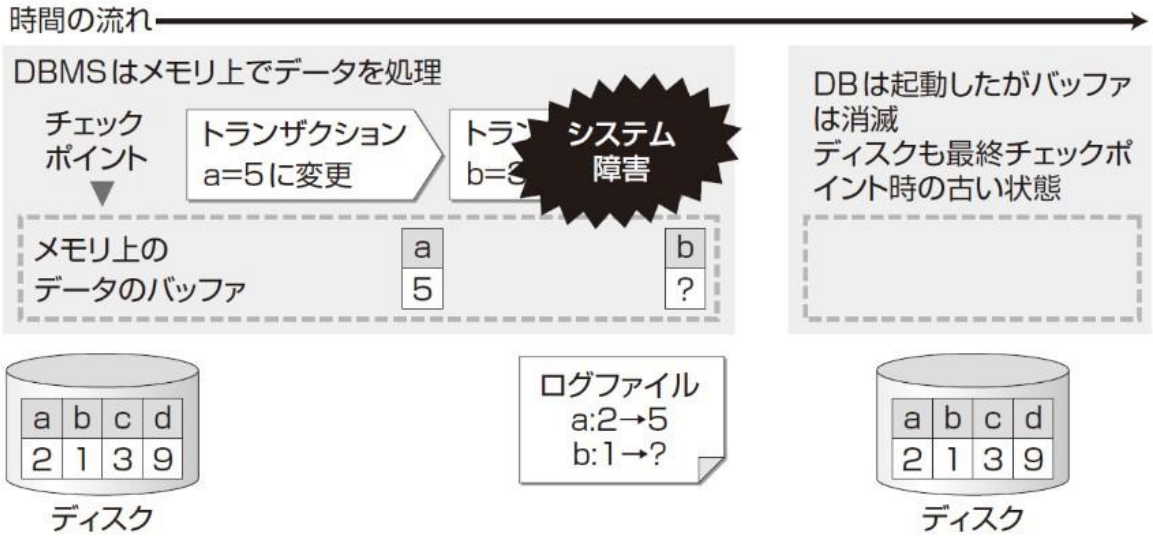


メモリにデータがなくて、ディスクまで検索しなければならないと遅い

チェックポイントでのディスク書き込みイメージ



システム障害からの回復イメージ



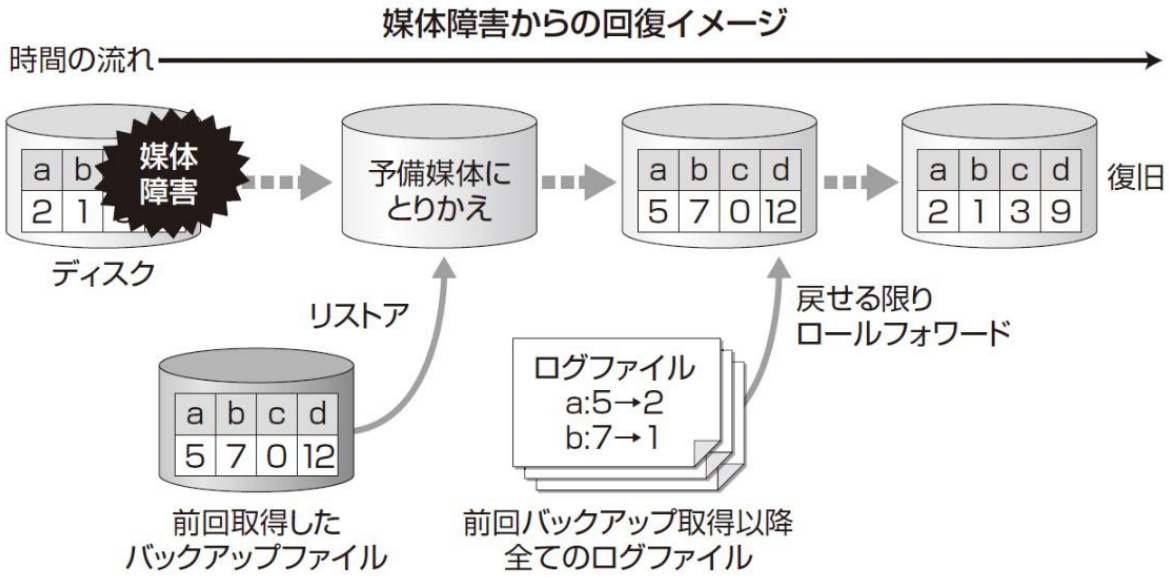
ログファイルを利用して，可能な限り復旧

a：障害発生前にトランザクション完了（トランザクション後の値が５とわかっている）
　→ロールフォワードにより５に書き換え

b：トランザクション実行中に障害発生（何の値にすべきかログに残っていない）
　→ロールバックにより，障害発生前の値に戻す。

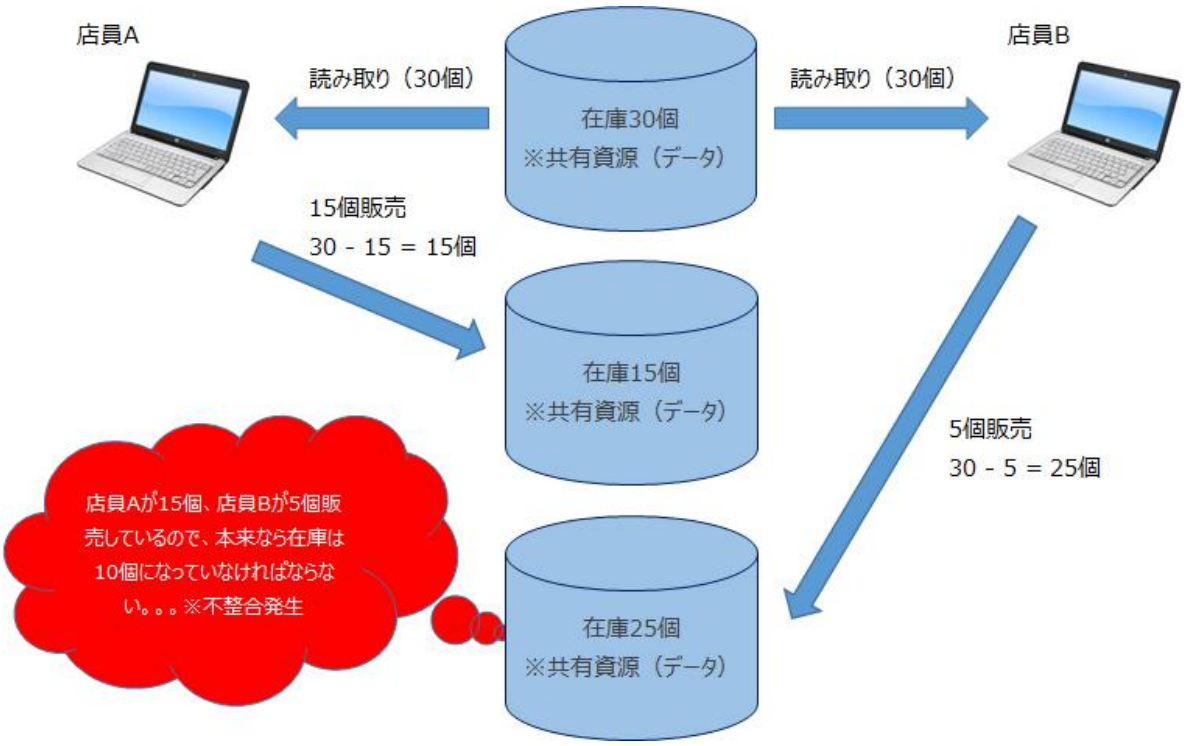
•

•

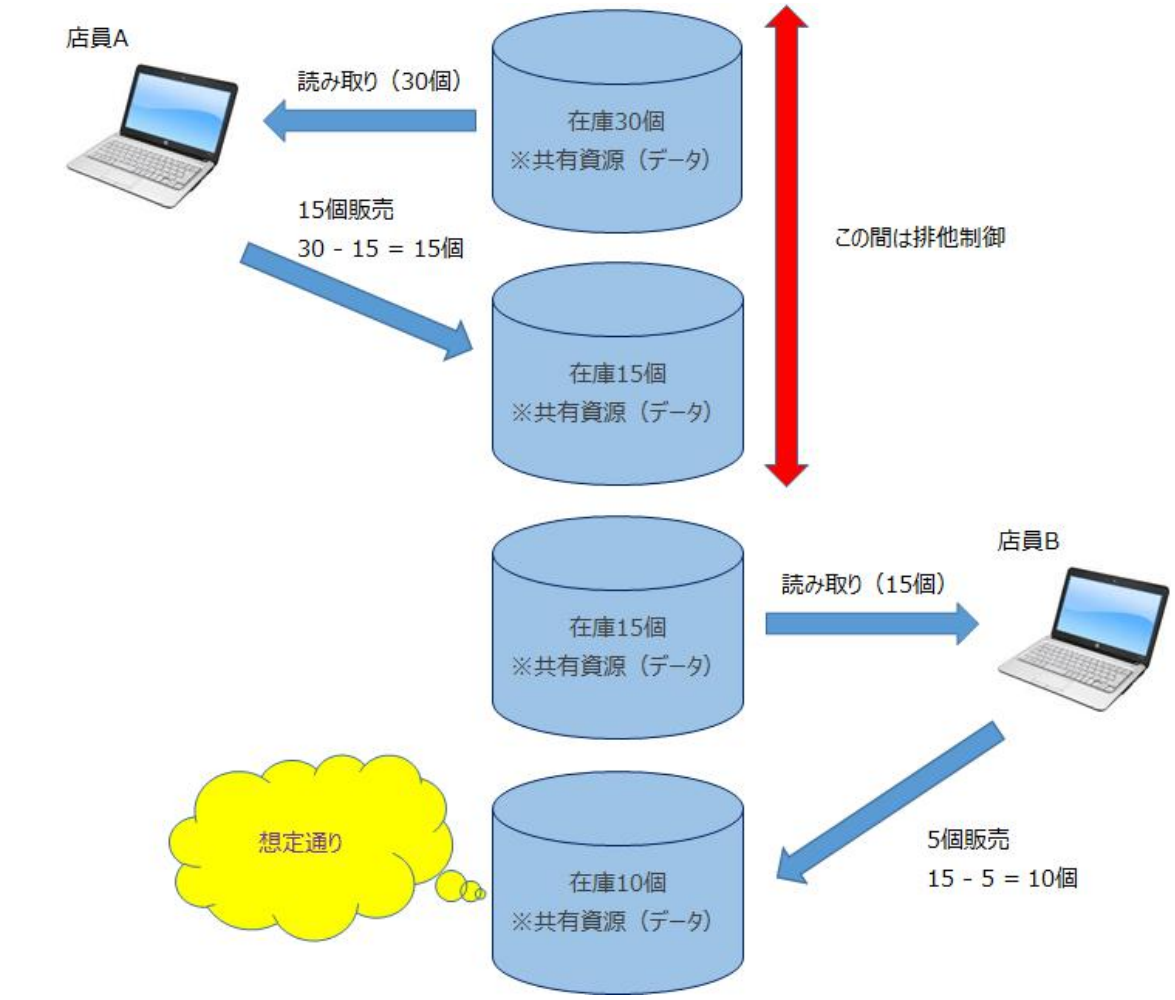


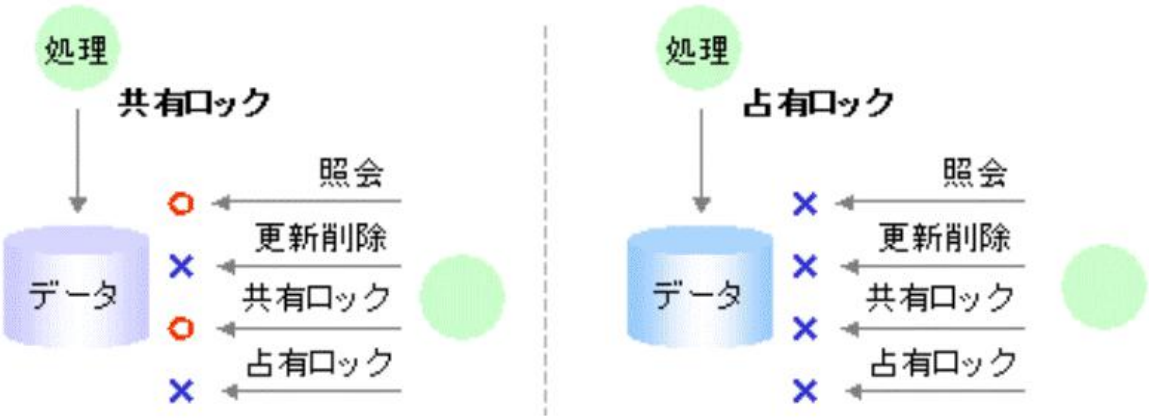

```
-- --------------------------------------------------------
-- Host:                         xxxxx
-- Server version:               10.1.38-MariaDB - mariadb.org binary
distribution
-- Server OS:                    Win64
-- HeidiSQL Version:             10.2.0.5611
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

# データベース作成
-- Dumping database structure for kizukeba_pronami_php
CREATE DATABASE IF NOT EXISTS `kizukeba_pronami_php` /*!40100 DEFAULT CHARACTER
SET utf8 COLLATE utf8_unicode_ci */;
USE `kizukeba_pronami_php`;

# テーブルのデータ型を指定
```

```
-- Dumping structure for table kizukeba_pronami_php.mst_staff
CREATE TABLE IF NOT EXISTS `mst_staff` (
  `code` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(15) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`code`)
) ENGINE=InnoDB AUTO_INCREMENT=22 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

# データを作成
-- Dumping data for table kizukeba_pronami_php.mst_staff: ~8 rows
(approximately)
/*!40000 ALTER TABLE `mst_staff` DISABLE KEYS */;
INSERT INTO `mst_staff` (`code`, `name`, `password`) VALUES
    (1, '秦基博', 'xxxxxxx'),
    (2, '藤原基央', 'xxxxxxx');
/*!40000 ALTER TABLE `mst_staff` ENABLE KEYS */;

/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1,
@OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```



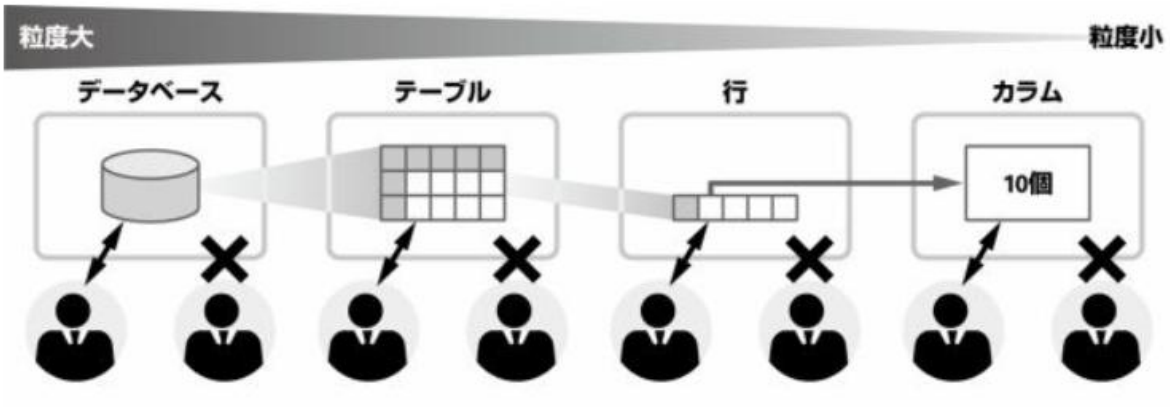
-


店員A
読み取り（30個）
在庫30個
※共有資源（データ）
15個販売
30 - 15 = 15個
この間は排他制御
在庫15個
※共有資源（データ）
店員B
読み取り（15個）
在庫15個
※共有資源（データ）
想定通り
5個販売
15 - 5 = 10個
在庫10個
※共有資源（データ）


処理
共有ロック
照会
更新削除
データ
共有ロック
占有ロック
処理
占有ロック
照会
更新削除
データ
共有ロック
占有ロック

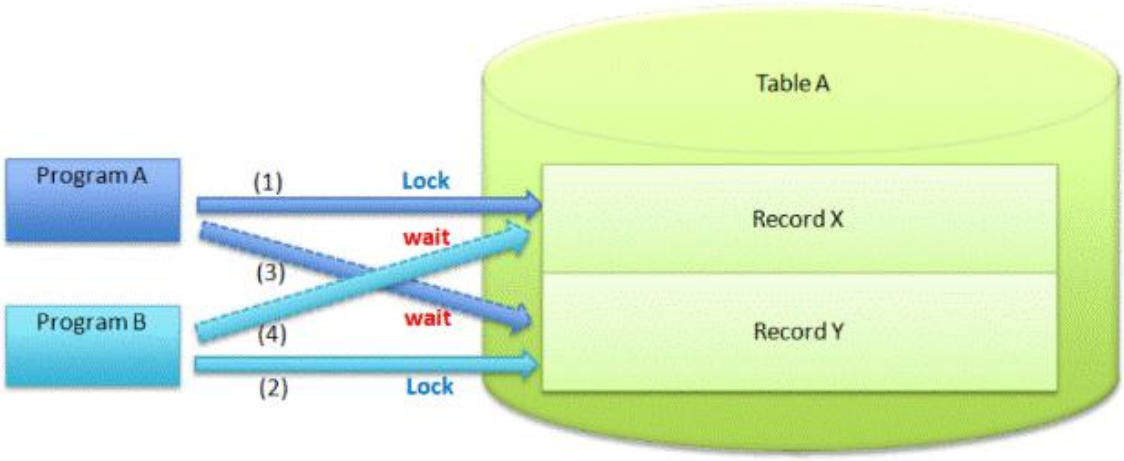
-

-


粒度大
粒度小
データベース
テーブル
行
カラム
10個

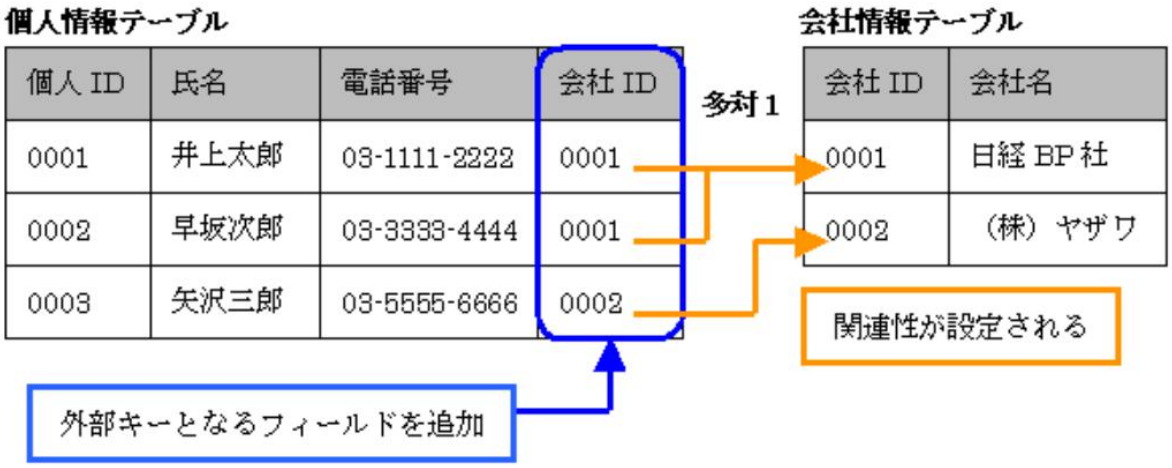

ロックの粒度
大（荒）
小（細）
DB全体
表
行
ブロック
同時実行数（レスポンス）
増大
減少
管理コスト（オーバヘッド）
減少
増大

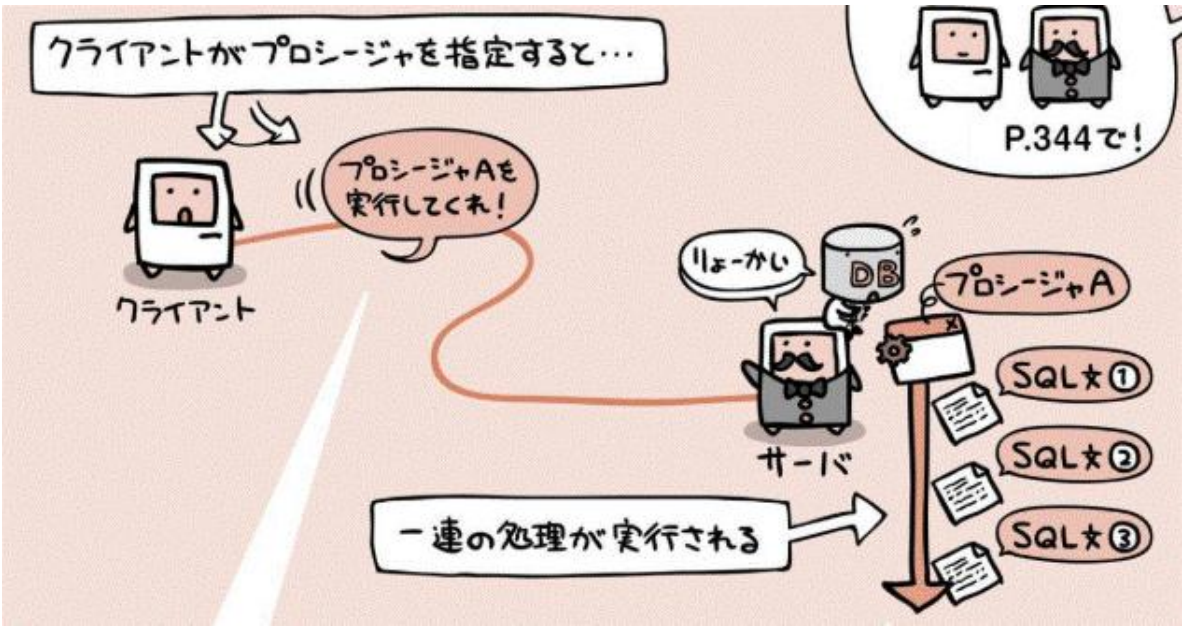
・
・
・
・




```
// 注文テーブル作成
CREATE TABLE order_data (

    // PRIMARY KEY制約
    order_id INT(10) PRIMARY KEY COMMENT '注文ID',

    // NOT NULL制約
    order_kbn INT(3) NOT NULL COMMENT '注文区分',
    system_create_date_time DATETIME NOT NULL COMMENT 'システム登録日時',
    system_update_date_time DATETIME NOT NULL COMMENT 'システム更新日時',
    delete_flg INT(1) DEFAULT 0 NOT NULL COMMENT '0：通常、1：削除済',

    // 参照制約キー
    FOREIGN KEY order_kbn REFERENCES order_kbn_data
)
```

-



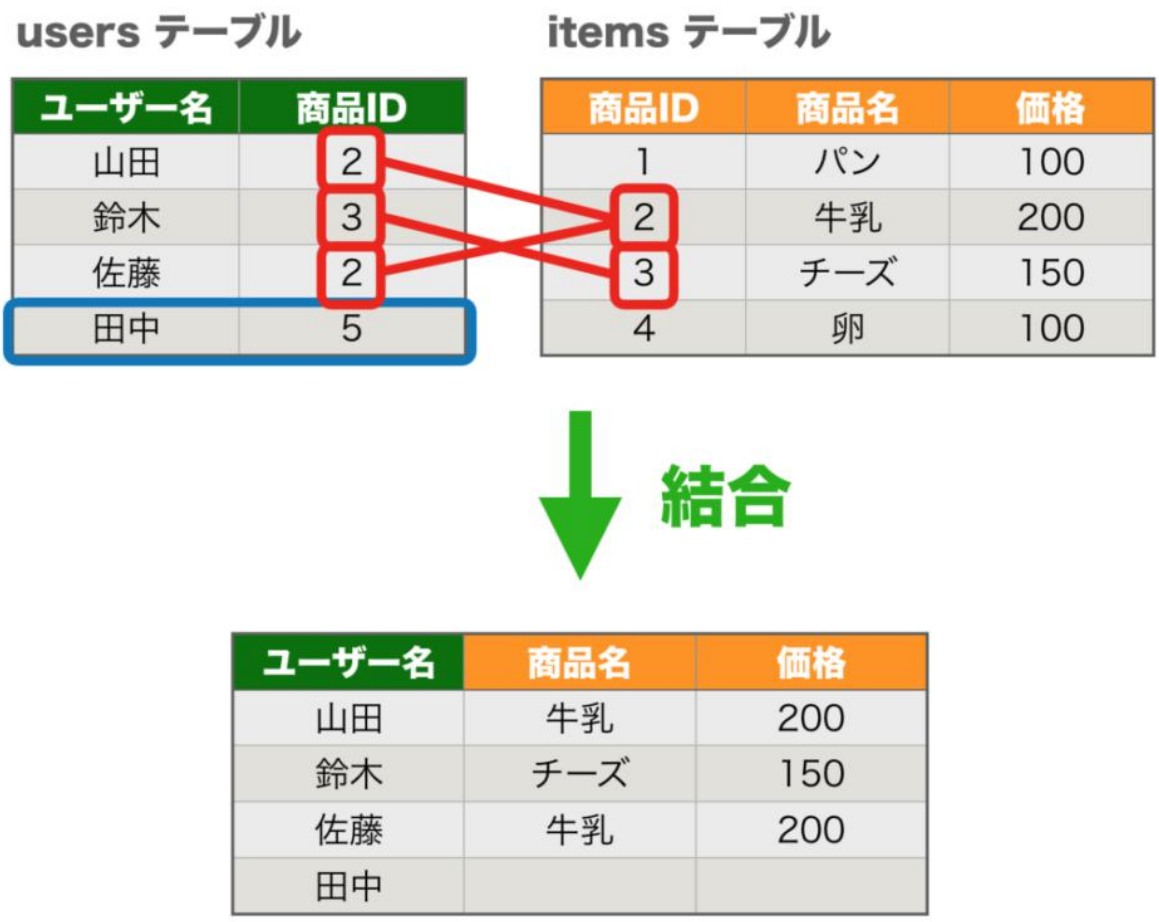



```
# PROCEDUREを作成し、データベースへ格納しておく。
CREATE PROCEDURE SelectContact AS
  SELECT CustomerID, CompanyName, ContactName, Phone
  FROM Customers
```

```
# PROCEDUREを実行
EXEC SelectContact
```

```
from ⇒ where ⇒ group by ⇒ having ⇒ select ⇒ order by
```

- `LEFT JOIN`



- `where`

```
select C,  #『カラム』を指定
    from T1, T2, T3  #複数の表を指定
    where R1 = R2 and  #指定したカラムのうち、レコードと別の表のレコードを照合し、フィールドを取得
    R2 = R3  #3つ目の表のレコードとも照合
```

- `inner join on`

```
select C,  #『カラム』を指定
    from T1  #複数の表を指定
    inner join T2
        on T1.C1 = T2.C2  #2つ目の表の『レコード』と照合
    inner join T3
        on T1.C1 = T3.C3  #3つ目の表の『レコード』と照合
```

-

-

```
select sum(C)  #指定したカラムで、『フィールド』の合計を取得
    from T
```

-

```
select avg(C)  #指定したカラムで、『フィールド』の平均値を取得
    from T
```

-

```
select min(C)  #指定したカラムで、『フィールド』の最小値を取得
    from T
```

-

```
select max(C)  #指定したカラムで、『フィールド』の最大値を取得
    from T
```

-

```
select count(C)  #指定したカラムで、『フィールド』の個数を取得
    from T
```

```
select avg(sum(C))  #集合関数を集合関数の中に入れ子状にすることはできない。
    from T
```

```
select count(*)  #指定したカラムで、値無しも含む『フィールド』を取得
    from T
```

```
select count(C)  #指定したカラムで、値無しを除いた『フィールド』を取得

select count(all C)  #上に同じ
```

```
select count(distinct C)  #指定したカラムで、重複した『フィールド』を除く全ての『フィールド』を取得
```

```
SELECT * FROM fruit WHERE name = "みかん" OR name = "りんご";
```

```
SELECT * FROM fruit WHERE name IN("みかん","りんご");
```

`group by` `select`

```
select C
    from T
    group by C  #指定したカラムで、各グループのフィールドを集計
```

`group by` `having`

```
select C
    from T
    group by C
    having count(*) >=2  #集計値が２以上の『フィールド』を取得
```

`group by + having` `where`

```
select C
    from T
    group by C
    having R
```

```
select C
    from T
    where R
    group by C
```

```
select * from T  #Main-query
    where C != (select max(C) from T)  #Sub-query
```

-

```
select * from T
    where C in (xxx, xxx, ...)  #指定したカラムで、指定した値の『フィールド』を取得
```

```
select * from T
    where C not in (R1, R2, ...)  #指定したカラムで、指定した値以外の『フィールド』を取得
```

```
select * from T
    where C not in ( #フィールドを指定の値として用いるｚ
        select C from T where R >= 160)  #指定したカラムで、『フィールド』を取得
```

-

`in`

```
select * from T
    where C = any(xxx, xxx, xxx)
```

```
create view T as
    select * from ...
```

```
select * from T
    where C like '%営業'  #任意の文字（文字無しも含まれる）
```

```
select * from T
    where C like '_営業'  #任意の一文字
```

```
select * from T
    between 1 and 10  #指定したカラムで、1以上10以下の『フィールド』を取得
```

```
$sql = "SELECT * FROM doraemon_characters";
$stmt = $dbh->prepare($sql);
$stmt->execute()


// 全てのデータ行を取得
$data = $stmt->fetchAll();
print_r($data);


// カラム名と値の連想配列として取得できる。
Array
(
    [0] => Array
    (
        [id] => 1
        [name] => のび太
        [gender] => man
        [type] => human
    )
    [1] => Array
    (
        [id] => 2
        [name] => ドラえもん
```

```
        [gender] => man
        [type] => robot
    )
)
```

`getConnection()` `fetchAll()`

```
// select文やパラメータを設定し、fetchAllへメソッドチェーン
getConnection()->executeQuery($query, $params, $types, $qcp)->fetchAll()


// カラム名と値の連想配列として取得できる。
Array  = (
    [kbn_name] => レッド
    [kbn_value] => 2
    [quantity] => 50
)
```

```
// $_POSTを用いて、送信されたpostメソッドのリクエストを受け取り、属性から各値を取得
$staff_name = $_POST['name'];
$staff_pass = $_POST['pass'];


// HTMLとして変数の内容を出力する際、「<」「>」などの特殊文字をエスケープ（無害化）
$staff_name = htmlspecialchars($staff_name, ENT_QUOTES, 'UTF-8');
$staff_pass = htmlspecialchars($staff_pass, ENT_QUOTES, 'UTF-8');


// データベースと接続（イコールの間にスペースを入れるとエラーになる）
$dsn = 'mysql:dbname=kizukeba_pronami_php;
host=kizukebapronamiphp
charaset=UTF-8';
$user = 'root';
$password = '';
$dbh = new PDO($dsn, $user, $password);
$dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);


// データベースに対してCREATE
$sql='INSERT INTO mst_staff (name,password) VALUES (?,?)';
$stmt = $dbh->prepare($sql);


// 配列に値を格納（格納する値の順番と、SQLでの引数の順番は、合わせる必要がある）
$data[] = $staff_name;
$data[] = $staff_pass;
```

```
// SQLを実行
$stmt->execute($data);


// データベースとの接続を切断
$dbh = null;
```

`db:reset`

```
namespace Migration

class ItemQuery
{
    public static function insert()
    {
        return "INSERT INTO item_table VALUES(1, '商品A', 1000, '2019-07-24
07:07:07');"
    }
}
```