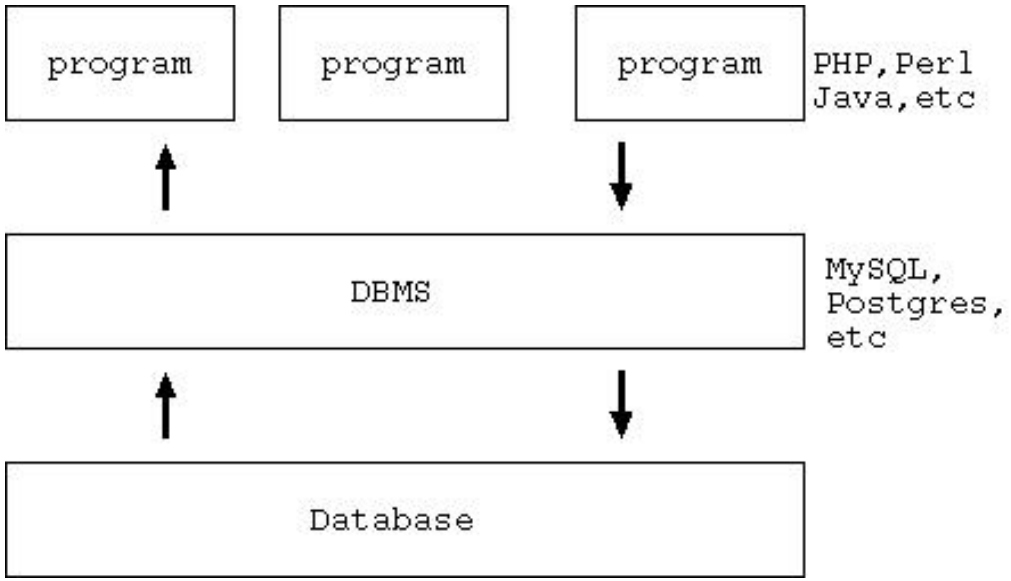
データベースシステム
第9回

理工学部情報科学科
松澤　智史

本日の内容

- DBMS(DataBase Management System)
  - インデックス



インデックスとは

MySQLでの説明

インデックスは、カラムが特定の値をもつレコードの迅速な検索に使用されます。インデックスがないと、MySQL がレコードを見つけるために、最初のレコードから開始し、テーブル全体を読み取るとが必要になります。テーブルが大きくなると、これにコストがかかります。クエリ対象のカラムにインデックスがあると、MySQL は全てのデータを探すことなく、データファイルの途中にあるシーク対象ポジションを迅速に取得することができます。テーブルに 1000 レコードある場合、シーケンシャルに読み取る場合と比較して少なくとも 100 倍は高速化できます。1000 レコードのほとんどすべてにアクセスする必要がある場合は、ディスクシークが最小になるため、シーケンシャルに読むほうが速くなることに注意してください。

- インデックスを作成すると、検索速度が向上する
- 全てのテーブルにインデックスを作成すると
  書き込み速度の低下や、データの保守の為に必要な容量が増えてしまう
- 検索機能は向上するがデータ容量が肥大化する

インデックスの確認方法

```
mysql> use test_db;
Database changed
mysql> show index from student;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| student |          0 | PRIMARY  |            1 | id          | A         |           4 |     NULL |   NULL |      | BTREE      |
| student |          0 | email    |            1 | email       | A         |           4 |     NULL |   NULL | YES  | BTREE      |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
2 rows in set (0.01 sec)

mysql> use world;
Database changed
mysql> show index from city
    -> ;
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| city  |          0 | PRIMARY     |            1 | ID          | A         |        4188 |     NULL |   NULL |      | BTREE      |
| city  |          1 | CountryCode |            1 | CountryCode | A         |         232 |     NULL |   NULL |      | BTREE      |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
2 rows in set (0.01 sec)
```

mysql> show index from テーブル名

Primary keyなど，ユニークな列には，
すでにインデックスが作られている

インデックスの作成

```
mysql> alter table city add index test_index(id,Name);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from city;
+-------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table | Non_unique | Key_name   | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+-------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| city  |          0 | PRIMARY    |            1 | ID          | A         |        4188 |     NULL |   NULL |      | BTREE      |
| city  |          1 | CountryCode|            1 | CountryCode | A         |         232 |     NULL |   NULL |      | BTREE      |
| city  |          1 | test_index |            1 | ID          | A         |        4079 |     NULL |   NULL |      | BTREE      |
| city  |          1 | test_index |            2 | Name        | A         |        4079 |     NULL |   NULL |      | BTREE      |
+-------+------------+------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
4 rows in set (0.01 sec)

mysql>
```

mysql> ALTER TABLE テーブル名 ADD INDEX インデックス名(列名);

mysql> CREATE INDEX インデックス名 ON テーブル名(列名);

インデックスの削除

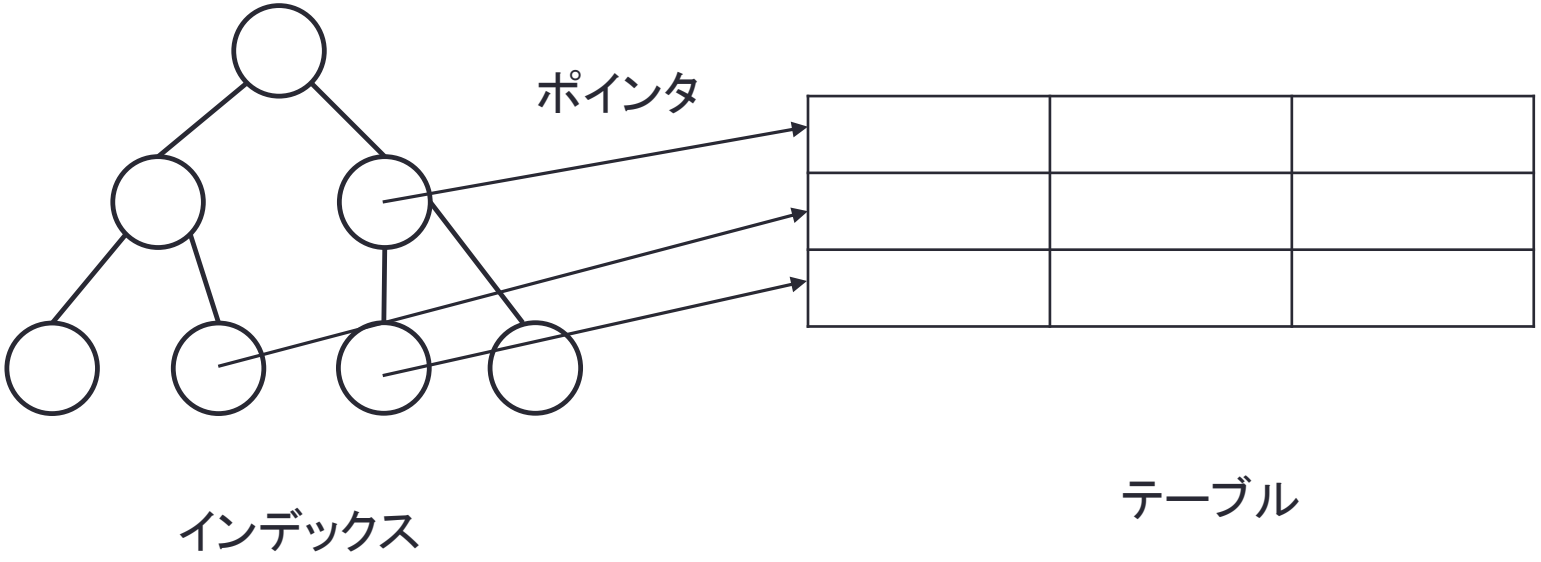
```
mysql> drop index test_index on city;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from city;
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| city  |          0 | PRIMARY     |            1 | ID          | A         |        4188 |     NULL |   NULL |      | BTREE      |
| city  |          1 | CountryCode |            1 | CountryCode | A         |         232 |     NULL |   NULL |      | BTREE      |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
2 rows in set (0.00 sec)
```

mysql> ALTER TABLE テーブル名 DROP INDEX インデックス名;

mysql> DROP INDEX インデックス名 ON テーブル名;

インデックスの仕組み

- データベースのレコードを取得する際に利用する別のデータ
- 検索に利用する列の値とそのレコードへのポインタで構成

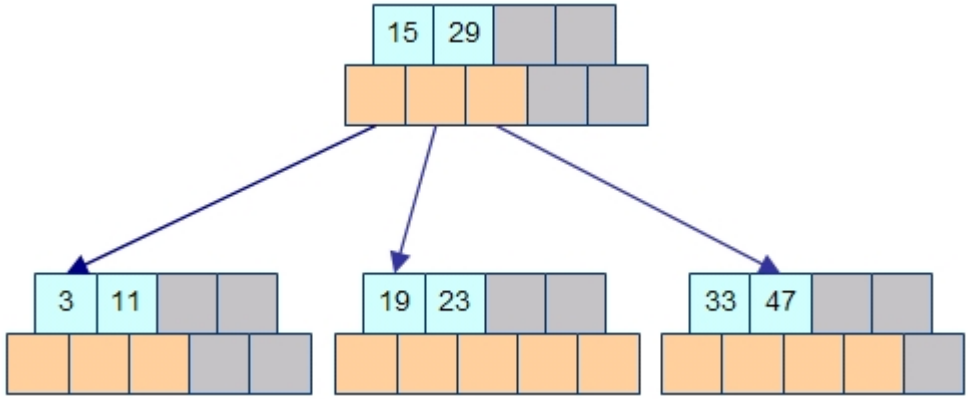


インデックスの構造の種類

- B木(バランス木)
  - MySQLのデフォルトのエンジンではB木のみ使用可
- ビットマップ
  - Oracleなどに実装
- ハッシュ
  - MySQLでも使用可

B木(バランス木)構造インデックス

- N分木ともいう，N=2の時は二分木
- 根(root)から葉(leaf)までの高さがすべて同じバランス木



- 主キーなどのインデックスに使用
- 大小比較がしやすいので範囲探索(Between)も可能
- 木を探索してアクセスするため，複数のインデックスの同時使用不可

ビットマップ構造インデックス

- 取り得る値の数が少なく，複雑な検索が行われる際に利用

MA	0	0	0	1	0	1	0
IS	1	1	0	0	0	0	1
BS	0	0	1	0	1	0	0

- データ検索時，1に該当するポインタをたどることで高速にアクセス可能

- AND，ORの条件指定などの組み合わせて使用できる

ハッシュ構造インデックス

- ハッシュ関数を使ってハッシュ値を計算し，
  その値からデータの格納位置を求める方法

- あらかじめデータ数などが決まっている場合，
  衝突が起きにくい場合に利用

ユニーク・非ユニーク インデックス

- ユニークインデックス
  - インデックス値に対するレコードが一つに定まる
  - 主キーや値が重複しない列のインデックスはユニークインデックスとなる
- 非ユニークインデックス
  - インデックス値に対するレコードが複数ある
  - 外部キーなどのインデックスは非ユニークインデックスとなる

```
mysql> use test_db;
Database changed
mysql> show index from student;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| student |          0 | PRIMARY  |            1 | id          | A         |           4 |     NULL |   NULL |      | BTREE      |
| student |          0 | email    |            1 | email       | A         |           4 |     NULL |   NULL | YES  | BTREE      |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
2 rows in set (0.01 sec)

mysql> use world;
Database changed
mysql> show index from city
    -> ;
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| Table | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
| city  |          0 | PRIMARY     |            1 | ID          | A         |        4188 |     NULL |   NULL |      | BTREE      |
| city  |          1 | CountryCode |            1 | CountryCode | A         |         232 |     NULL |   NULL |      | BTREE      |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
2 rows in set (0.01 sec)
```

インデックスがユニークなら0，重複ありなら1

インデックスの使用確認

- EXPLAIN
  - SQLの解析を行う
  - SQL文の前にEXPLAINを入れる

```
mysql> explain select * from city;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | city  | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 4188 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

  - TYPEの結果を見る
    - const ： PRIMARYKEYかユニークインデックスによる検索アクセス　最速
    - eq_ref ：JOINにおいてPRIMARYKEYかユニークインデックスが利用される時のアクセスタイプ
    - ref ：ユニークでないインデックスを使って等価検索を行った時のタイプ
    - index ： インデックス全体をスキャン　遅い
    - ALL ：テーブル全体のスキャンでインデックスが使用されていない　最も遅い

```
mysql> explain select * from city where id=1;
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | E
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-
|  1 | SIMPLE      | city  | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | N
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from city where id>100;
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Ex
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-
|  1 | SIMPLE      | city  | NULL       | range | PRIMARY       | PRIMARY | 4       | NULL | 2094 |   100.00 | Us
+----+-------------+-------+------------+-------+---------------+---------+---------+------+------+----------+-
1 row in set, 1 warning (0.00 sec)
```

ほぼ最速の検索

```
mysql> use test_db;
Database changed
mysql> explain select * from student natural join test;
+----+-------------+---------+------------+--------+---------------+---------+---------+-------------------+
| id | select_type | table   | partitions | type   | possible_keys | key     | key_len | ref               |
+----+-------------+---------+------------+--------+---------------+---------+---------+-------------------+
|  1 | SIMPLE      | student | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL              |
|  1 | SIMPLE      | test    | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | test_db.student.id |
+----+-------------+---------+------------+--------+---------------+---------+---------+-------------------+
2 rows in set, 1 warning (0.00 sec)

mysql> explain select * from student where name='Alice';
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |    25.00 |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+
1 row in set, 1 warning (0.00 sec)
```

自然結合

インデックスが作られていない列

```
mysql> explain select * from student where name='Alice';
+----+-------------+---------+------------+------+---------------+------+---------+------+------+-------
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtere
+----+-------------+---------+------------+------+---------------+------+---------+------+------+-------
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |    25.0
+----+-------------+---------+------------+------+---------------+------+---------+------+------+-------
1 row in set, 1 warning (0.00 sec)

mysql> alter table student add index test_index(name);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select * from student where name='Alice';
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+
| id | select_type | table   | partitions | type | possible_keys | key        | key_len | ref   | rows |
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+
|  1 | SIMPLE      | student | NULL       | ref  | test_index    | test_index | 60      | const |    1 |
+----+-------------+---------+------------+------+---------------+------------+---------+-------+------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

Name列でインデックスを作成しておくと，Typeがrefとなる
作られたインデックス（B木）からAliceに該当するレコードを探すので，
rowsも1となり，高速に結果を出せる

※数行程度での速度向上は誤差の範囲

select_typeは副問合せ（サブクエリ）があると変化

インデックスを利用する例(MySQL)

- フィールド値を定数と比較する時
  (WHERE name = “Alice”)
- フィールド値全体でJOINする時
  (WHERE a.name = b.name)
- フィールド値の範囲を求める時

- ORDER BY, GROUP BY

- MIN(), MAX() 関数

- LIKEで文字列の先頭が固定な時

インデックスを利用しない例

- LIKEがワイルドカードで始まる時
- DB全体を読んだ方が早いとMySQLが判断した時
- 通常，インデックスはORDER BYには使われない
- WHERE と ORDER BYのフィールドが違う時にはどちらかしか使われない

インデックス設計

- 抑えておくべきポイント
  - インデックスを作成するとSELECTで
    検索時のスピードはあがる(ことが多い)
  - INSERT, DELETE, UPDATE等のデータの再構成(追加削除変更)
    ではインデックスの再構成が必要なため処理が遅くなる
- 一つのテーブルにインデックスを設定しすぎない
  - データの再構成時影響が大きくなる
- データ量が少ないときは設定しない方が良い場合もある
  - データ量が少ない時はインデックスからアクセスするより直接テーブル
    にアクセスした方が効率が良いケースもある

実験データの作成(テーブルの作成)

```
use test_db;
DROP TABLE IF EXISTS user;
CREATE TABLE user(
  id integer not null auto_increment,
  name varchar(255) not null,
  email varchar(255) not null,
  PRIMARY KEY(id)
)
```

test_table.sqlに記載

実験データの作成（PROCEDUREの作成）

```
DELIMITER //
DROP PROCEDURE testIn;
CREATE PROCEDURE testIn(max INT)
  BEGIN
  DECLARE c INT Default 1 ;
    simple_loop: LOOP
      INSERT INTO user (name, email) VALUES
(CONCAT('student',c), CONCAT('student',c,'@tus.ac.jp'));
      SET c=c+1;
      IF c=max THEN
        LEAVE simple_loop;
       END IF;
    END LOOP simple_loop;
END //
DELIMITER ;
```

test_procedure.sqlに記載

実験

```
mysql> source test_table.sql;
Database changed
Query OK, 0 rows affected (0.04 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> source test_procedure.sql;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> call testIn(100000);
Query OK, 1 row affected (5 min 25.70 sec)

mysql> select * from user where name='student80000' AND email='student80000@tus.ac.jp';
+-------+-------------+----------------------+
| id    | name        | email                |
+-------+-------------+----------------------+
| 80000 | student80000 | student80000@tus.ac.jp |
+-------+-------------+----------------------+
1 row in set (0.07 sec)

mysql> explain select * from user where name='student80000' AND email='student80000@tus.ac.jp';
+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+--
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows  | filtered | E
+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+--
|  1 | SIMPLE      | user  | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 97612 |     1.00 | U
+----+-------------+-------+------------+------+---------------+------+---------+------+-------+----------+--
1 row in set, 1 warning (0.00 sec)

mysql>
```

実験

```
mysql> show index from user;
+-------+------------+----------+--------------+-------------+-----------+-------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality |
+-------+------------+----------+--------------+-------------+-----------+-------------+
| user  |          0 | PRIMARY  |            1 | id          | A         |       94902 |
+-------+------------+----------+--------------+-------------+-----------+-------------+
1 row in set (0.01 sec)

mysql> alter table user add index index1(name);
Query OK, 0 rows affected (0.50 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table user add index index2(email);
Query OK, 0 rows affected (0.71 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from user;
+-------+------------+----------+--------------+-------------+-----------+-------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality |
+-------+------------+----------+--------------+-------------+-----------+-------------+
| user  |          0 | PRIMARY  |            1 | id          | A         |       94902 |
| user  |          1 | index1   |            1 | name        | A         |       97612 |
| user  |          1 | index2   |            1 | email       | A         |       97612 |
+-------+------------+----------+--------------+-------------+-----------+-------------+
3 rows in set (0.01 sec)

mysql>
```

実験

```
mysql> explain select * from user where name='student80000' AND email='student80000@tus.ac.jp';
+----+-------------+-------+------------+------+---------------+--------+---------+-------+------+
| id | select_type | table | partitions | type | possible_keys | key    | key_len | ref   | rows |
+----+-------------+-------+------------+------+---------------+--------+---------+-------+------+
|  1 | SIMPLE      | user  | NULL       | ref  | index1,index2 | index1 | 1022    | const |    1 |
+----+-------------+-------+------------+------+---------------+--------+---------+-------+------+
1 row in set, 1 warning (0.00 sec)

mysql> select * from user where name='student80000' AND email='student80000@tus.ac.jp';
+-------+--------------+-----------------------+
| id    | name         | email                 |
+-------+--------------+-----------------------+
| 80000 | student80000 | student80000@tus.ac.jp |
+-------+--------------+-----------------------+
1 row in set (0.00 sec)

mysql>
```

劇的に速くなる

この程度の行であればインデックスをnameやemailに作ることで改善できるが，
もっと多い行であれば，nameとemailの複合インデックスを作ることで
さらに改善が見込まれる

インデックス関係のSQL文まとめ

- テーブル作成時にインデックス作成
  - mysql> create table テーブル名(作成対象カラム名 データ型 not null,index インデックス名(作成対象カラム名));
- 既存のテーブルにインデックス作成
  - mysql> create index インデックス名 on テーブル名(作成対象カラム名);
  - mysql> alter table テーブル名 add index インデックス名(対象カラム名);
- インデックスの確認
  - mysql> show index from テーブル名;
- インデックスを使用して検索しているか確認
  - mysql> explain select * from テーブル名 where文～等
- インデックスの削除
  - mysql> drop index インデックス名 on テーブル名;

まとめ

- インデックスは作成するとSELECTの応答速度を向上させる
- インデックスは作成するとテーブルの変更の速度を低下させる
- インデックスの構造はDBMSなどによって異なる手法が存在するが，MySQLではB木が採用されている

質問あればどうぞ