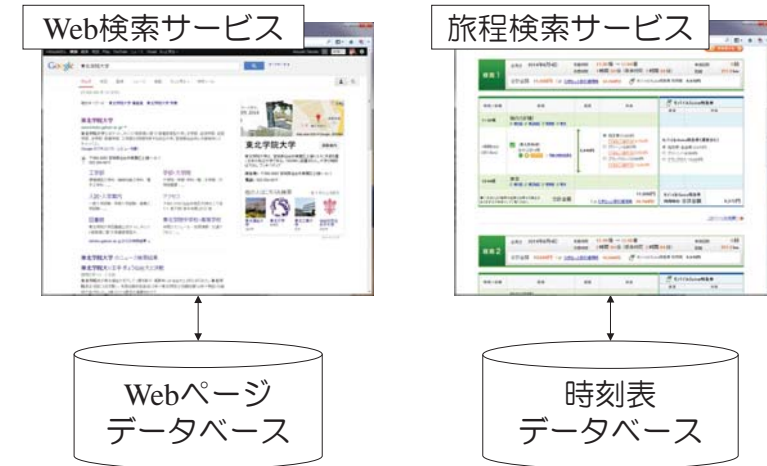


コンピュータ科学III

担当：武田敦志 <takeda@cs.tohoku-gakuin.ac.jp>
http://takeda.cs.tohoku-gakuin.ac.jp/

データベース管理システムとSQL(1)

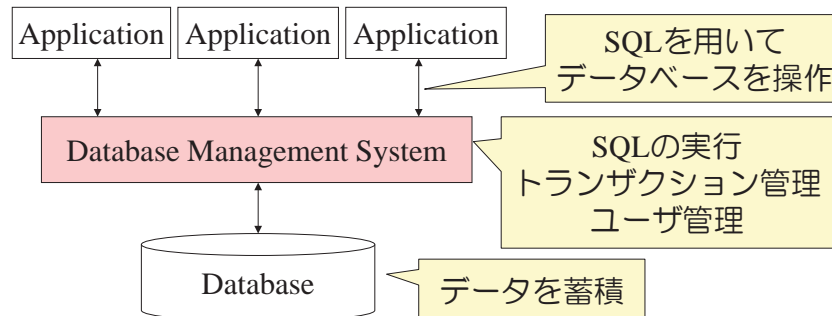
■データベース利用例



データベース管理システムとSQL(2)

■データベース管理システム

データベースの操作を行うためのソフトウェア
⇒ 関係データベースの場合はSQLを用いて操作する



データベース管理システムとSQL(3)

■SQL

データベースを操作するための「問い合わせ言語」

```
INSERT INTO student (student_id, name, dep_id)
VALUES ('7', 'tamaki', '3');
```

student_id	name	dep_id
1	hunaki	2
2	hanaoka	2
3	kishida	2
4	okamoto	1
5	moriyama	1
6	sonoda	1



student_id	name	dep_id
1	hunaki	2
2	hanaoka	2
3	kishida	2
4	okamoto	1
5	moriyama	1
6	sonoda	1
7	tamaki	3

データベース管理システムとSQL (4)

■データベースの操作

検索
`SELECT student_id, dep_id FROM student
WHERE dep_id = '1';`

追加
`INSERT INTO student (student_id, name, dep_id)
VALUES ('7', 'tamaki', '3');`

更新
`UPDATE student SET dep_id = '1'
WHERE student_id = '2';`

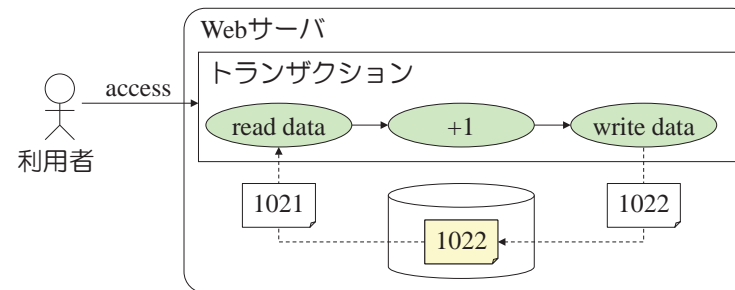
削除
`DELETE FROM student WHERE student_id = '3';`

page 5

トランザクション管理(1)

■トランザクション管理が必要なアプリケーションの例

Webページのアクセスカウンタ

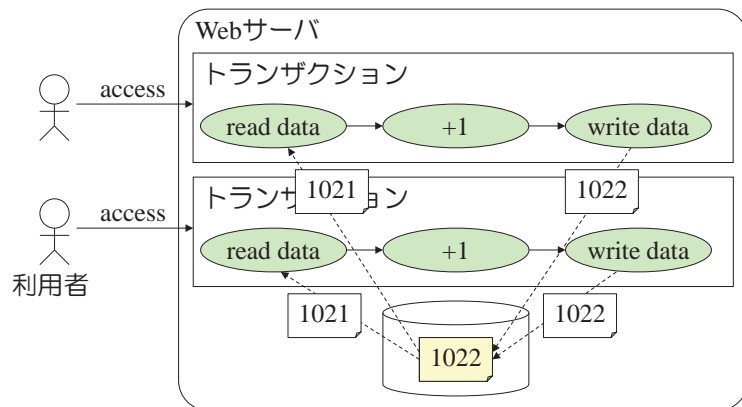


トランザクション = 切り離し不可能な一連の処理

page 6

トランザクション管理(2)

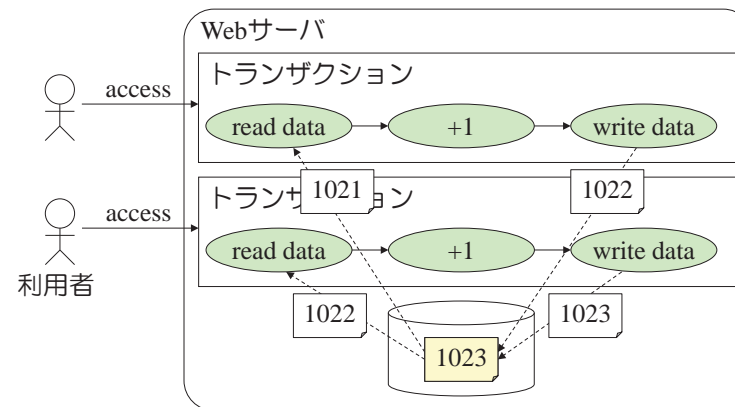
■トランザクション管理が必要となる例 (失敗例)



page 7

トランザクション管理(3)

■トランザクション管理が必要となる例 (成功例)

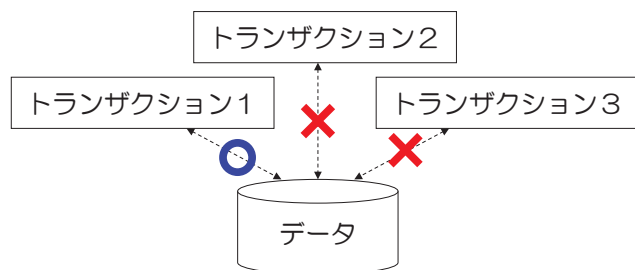


page 8

トランザクション管理(4)

■排他制御

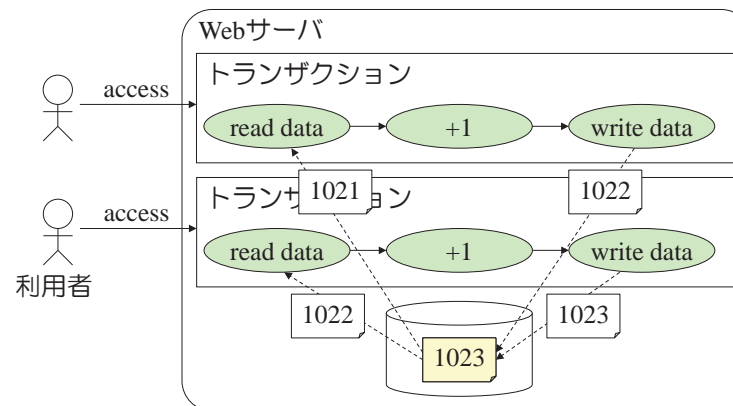
あるリソース（データ）に対する
同時アクセス（トランザクション）数を制限する



page 9

トランザクション管理(5)

■排他制御を行ったアクセスカウンタ



page 10

トランザクション管理(6)

■データ保護のために考える必要があること

- ◆ 原子性 すべてのトランザクションは完全に成功する or 完全に失敗する
- ◆ 整合性 作成・削除・更新されたデータには矛盾がない
- ◆ 隔離性 各トランザクションは独立に実行される
- ◆ 耐久性 成功したトランザクションの実行結果はその後発生する障害等の影響を受けない

page 11

データベース設計の概要(1)

■データベースの設計

データベースでは構造化されたデータを扱う

⇒ 追加・検索・更新・削除を高速に行うため

データベースでデータを扱う前に、格納するデータの構造(Schema)を決定する必要がある。

⇒ データベースの設計

page 12

データベース設計の概要(2)

■データベースの設計手順

- ▶ 現実世界のデータを解析する



- ▶ 概念モデルを設計する



- ▶ 論理モデルを設計する



- ▶ 論理モデルのチューニングを行う

page 13

データベース設計の概要(3)

■よいデータベース設計とは

現実世界でのデータ関係を上手く反映した設計

- ◆ データ関係が理解しやすい
⇒ データ関係を忠実・簡潔に記述する
- ◆ データの矛盾が発生しない
⇒ データの重複・不一致などが発生しない

データの操作が簡単な設計

- ◆ 挿入・検索・更新・削除が高速に出来る
- ◆ データの矛盾が発生しにくい

page 14

ERモデル(1)

■ER (Entity-Relationship Model) とは

『実体』と『関連』でデータ構成を記述するモデル

実体： 対象とする事象を表したもの
性質を表す複数の属性を持つ

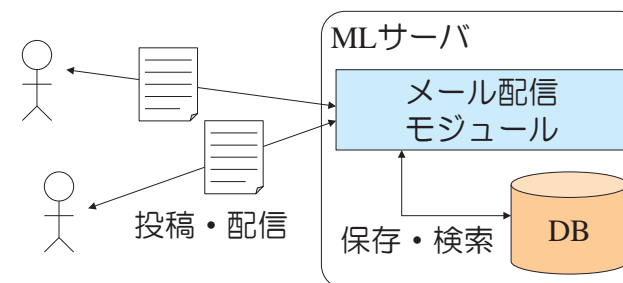
関連： 実体間の相互関係を表したもの
関連の性質を表すための属性を持つ

page 15

ERモデル(2)

■具体例（例題）

メールリスト管理データベース



page 16

ERモデル(3)

■具体例（実体と関連）

◆ 実体

ML利用者 - member

MLコミュニティ - community

投稿メール - message

◆ 関連

コミュニティ参加者 (member ↔ community)

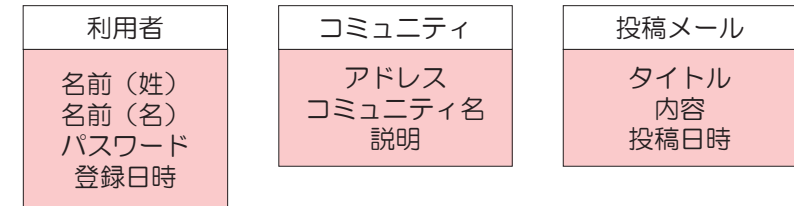
投稿先 (message ↔ community)

page 17

ERモデル(4)

■具体例（ER図・その1）

実体の作成
属性値の設定

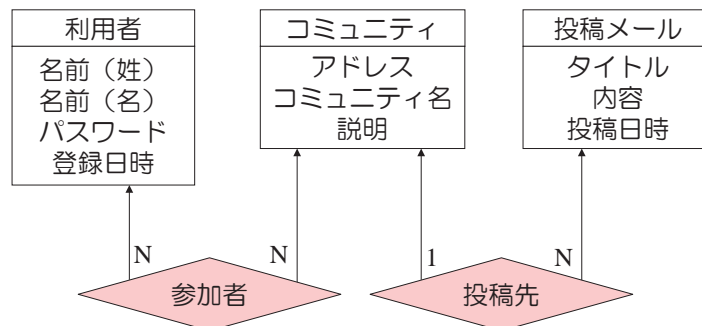


page 18

ERモデル(5)

■具体例（ER図・その2）

関連（Relation）を設定

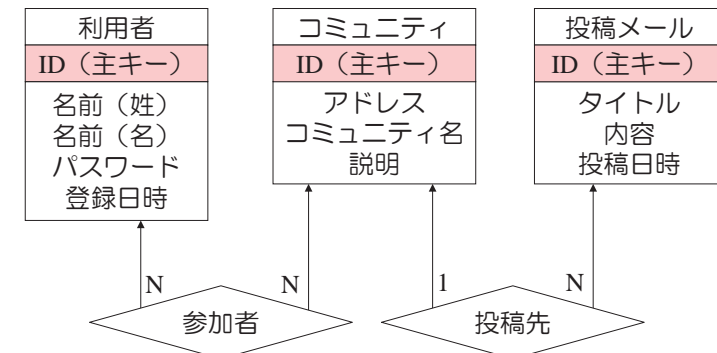


page 19

ERモデル(6)

■具体例（ER図・その3）

主キーを設定

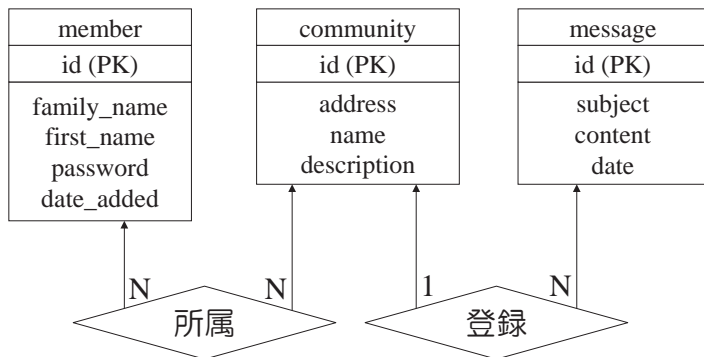


page 20

ERモデル(7)

■具体例 (ER図・その4)

データベース登録用の名称 (アルファベット) に変更

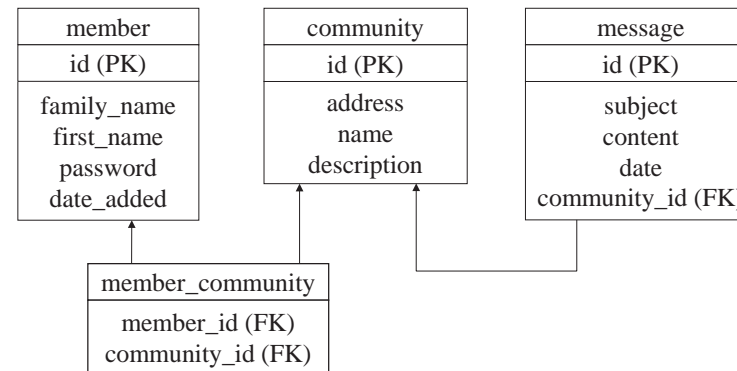


page 21

ERモデル(8)

■具体例 (ER図・その5)

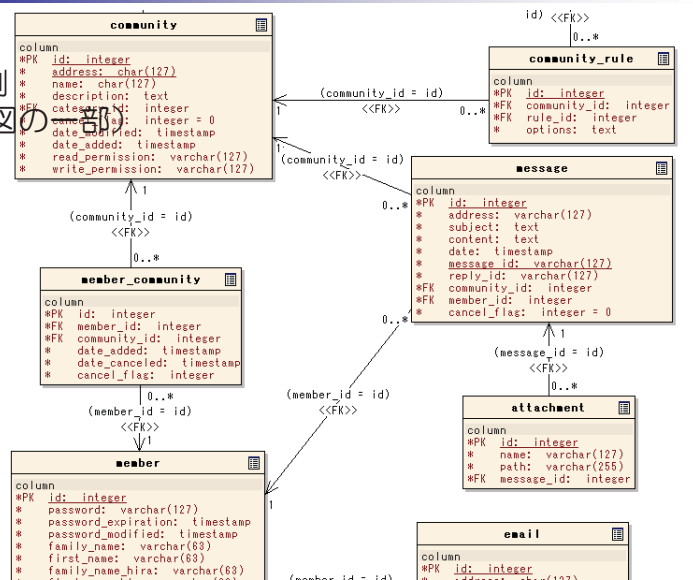
関連を外部キーで表現



page 22

ERモデル(9)

■具体例 (完成図の一部)



page 23

関連モデルへの変換(1)

■関連モデルへの変換方法

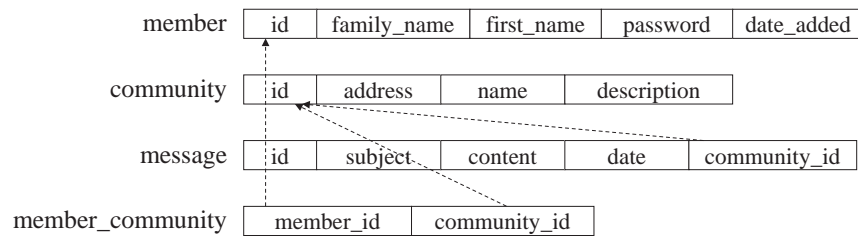
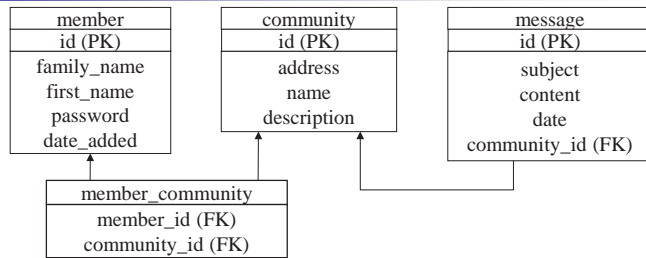
- ◆ 実体 ⇒ テーブル
- ◆ 属性 ⇒ カラム
- ◆ 主キー ⇒ 主キー
- ◆ 外部キー ⇒ 外部キー

⇒ 単純なマッピングで変換は可能

page 24

関連モデルへの変換(2)

■具体例



page 25

関連モデルへの変換(3)

■具体例 (SQL文の生成)

```
CREATE TABLE message (  
  id integer NOT NULL,  
  subject text NOT NULL,  
  content text NOT NULL,  
  date timestamp NOT NULL,  
  community_id integer NOT NULL,  
);
```

```
ALTER TABLE message ADD CONSTRAINT PK_message  
  PRIMARY KEY (id);
```

```
ALTER TABLE message  
  ADD CONSTRAINT UQ_message_id UNIQUE (id);
```

```
ALTER TABLE message ADD CONSTRAINT FK_message_community  
  FOREIGN KEY (community_id) REFERENCES community (id);
```

page 26

この講義で取り上げていないこと

■トランザクション

- 排他制御の種類
- デッドロック
- トランザクション障害の種類

■データベースの設計

- 正規化
- モデルのチューニング

■データベースシステム

- 分散データベース

page 27