

# 応用プログラミングII

## 第3回

岩下・柴田

# 本日の内容

1. データベースとは
2. NetBeansによるデータベースの扱い
  - i. データベースの起動
  - ii. データベース・テーブルの作成
  - iii. テーブルへのレコード追加
3. SQL文
  - i. SELECT文によるデータ抽出
  - ii. UPDATE文によるデータ変更
  - iii. INSERT文によるデータ追加
  - iv. DELETE文によるデータ削除
4. バックアップ

# 1. データベースとは

# データベースとは

- データを保存して効率よく利用する
- リレーショナルデータベース
  - データを2次元の表の集まりとして扱う
  - データ構造とプログラムが独立していて、データの再利用が楽
- SQL(問い合わせ言語)を使ってデータベースにアクセス可能

# 今回作成するデータベース

- 学生情報を管理する「名簿」のデータベース
- データベース内で、「学部の情報」と「学生の情報」の表を管理したい  
→ テーブル

## 【名簿データベース】

学部の情報(学部テーブル)

学部ID	学部名
1	コンピュータサイエンス
2	応用生物
3	メディア

学生の情報(学生情報テーブル)

学生番号	氏名	学部ID	学年
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1

# テーブル

- テーブル

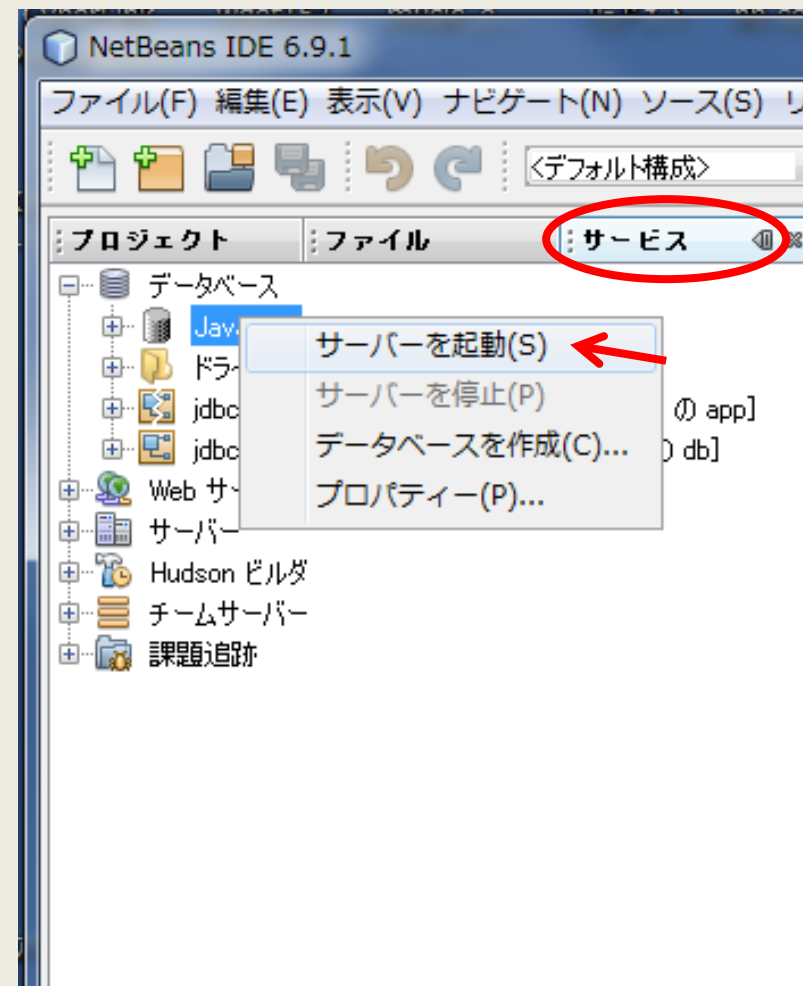
- 行(レコード): 各データ
- 列(フィールド): 項目
- 主キー(primary key): レコードを一意に特定できるフィールド

学生番号	氏名	学部ID	学年
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1

## 2. NetBeansによるデータベースの扱い

## 2-i. NetBeansにおけるデータベースの起動

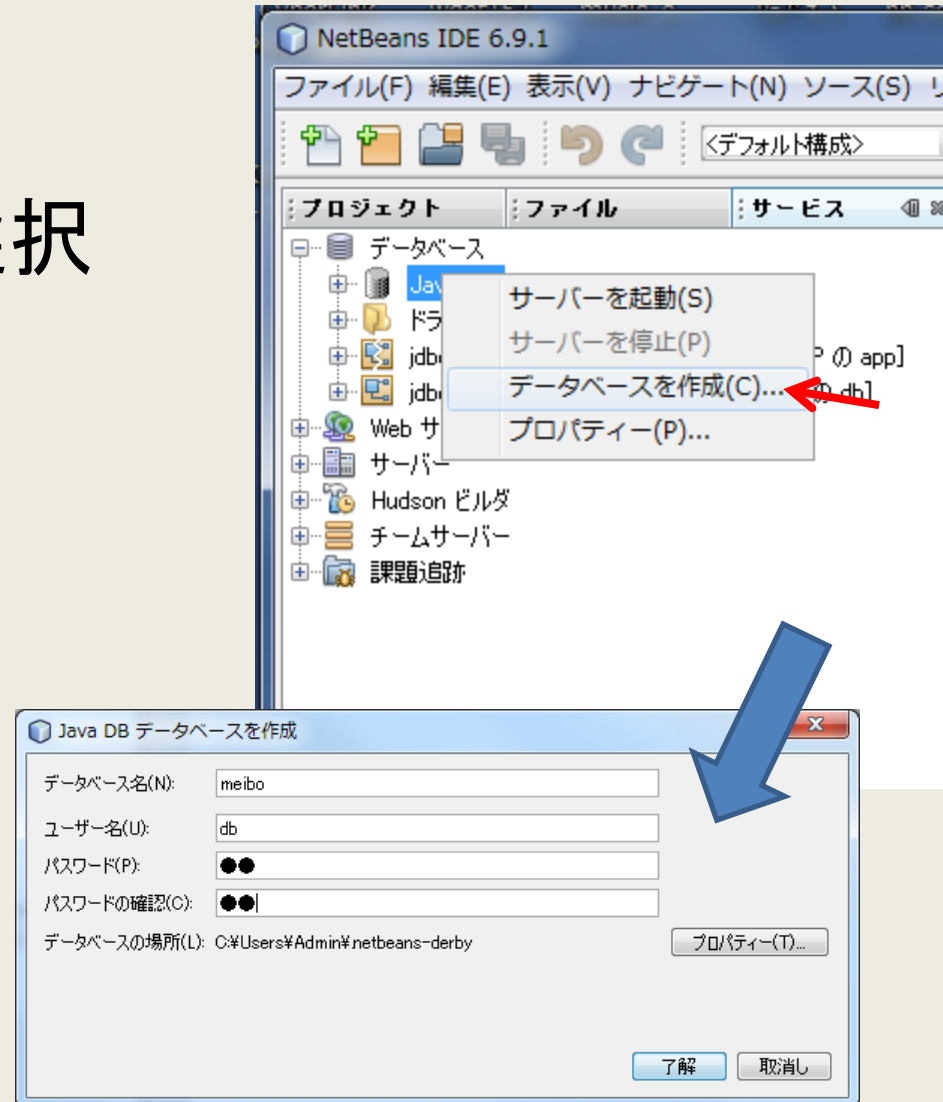
- 今回はJava8に標準で付属しているデータベースJavaDBを使う
- 「サービス」ウィンドウで「データベース⇒Java DB」を右クリック
- 「サーバを起動」を選択





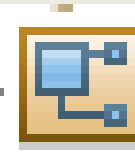
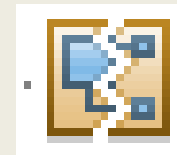
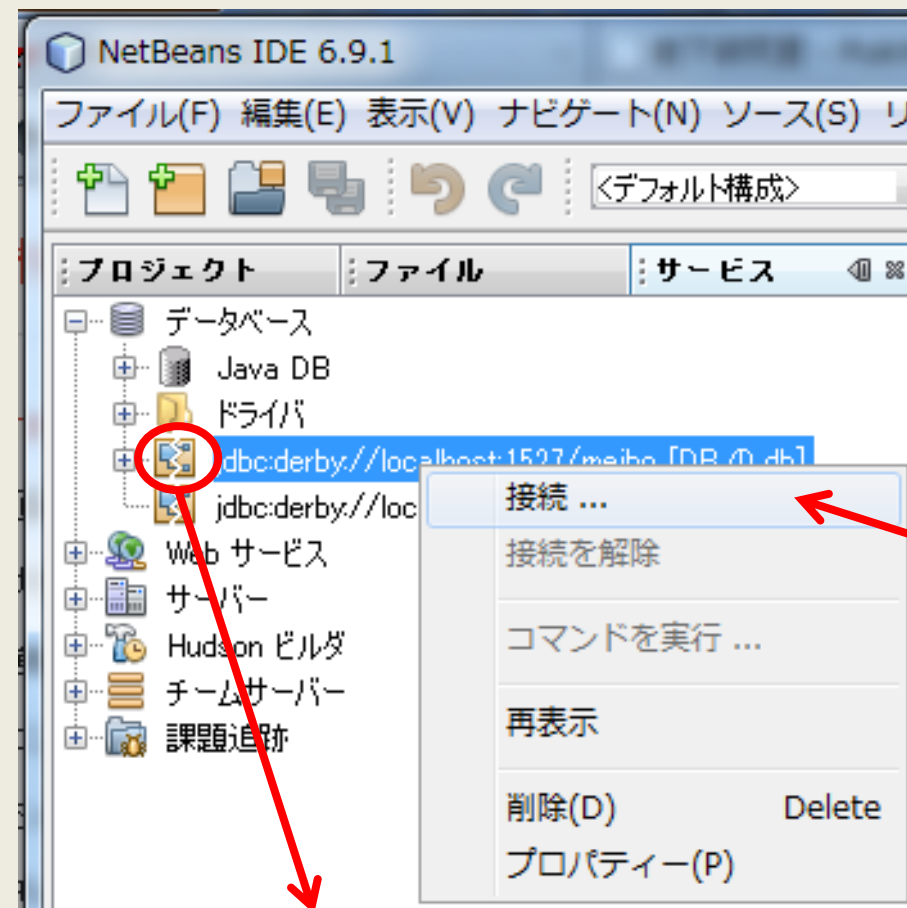
## 2-ii. データベースの作成(1)

- 「Java DB」を右クリック
- 「データベースを作成」を選択  
(下記を入力)
- データベース名 : meibo
- ユーザ名 : db
- パスワード : db



# データベースの作成(2)

- 「サービス」ウィンドウの「データベース」中に表示された、作成したデータベースを右クリック → 「接続」 でデータベースへ接続
- 接続が完了するとアイコンが変わる



# テーブルの作成(1)

- 今回は以下に示す2つのテーブルを作成

【学部テーブル:T\_GAKUBU】

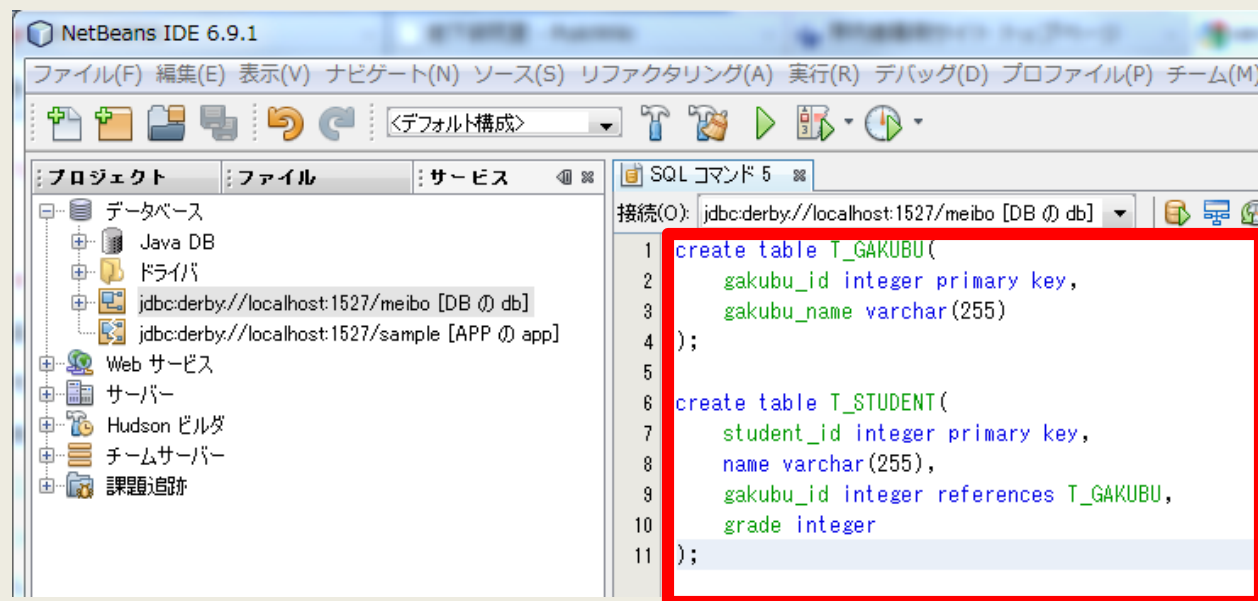
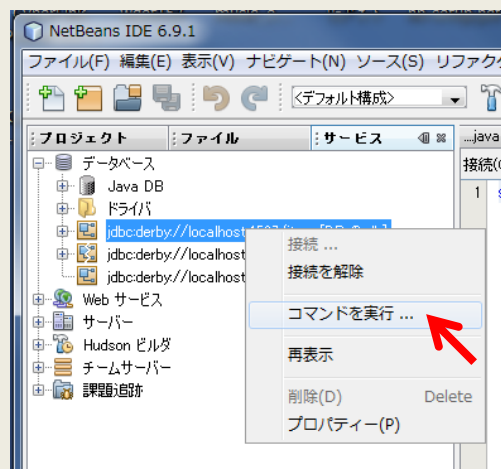
フィールド	説明	型	備考
<b>gakubu_id</b>	学部ID	整数	主キー
gakubu_name	学部名	文字列	

【学生情報テーブル】:T\_STUDENT

フィールド	説明	型	備考
student_id	学生番号	整数	主キー
fullname	氏名	文字列	
<b>gakubu_id</b>	学部ID	整数	
grade	学年	整数	

# テーブルの作成(2)

- meiboを右クリックし,「コマンドを実行」
- SQLコマンドによりテーブルを作成



次ページのSQL文を記述

# テーブルの作成(3)

- 2つのテーブルを作成

```
create table T_GAKUBU(  
    gakubu_id integer primary key,  
    gakubu_name varchar(255)  
);
```

```
create table T_STUDENT(  
    student_id integer primary key,  
    fullname varchar(255),  
    gakubu_id integer,  
    grade integer  
);
```

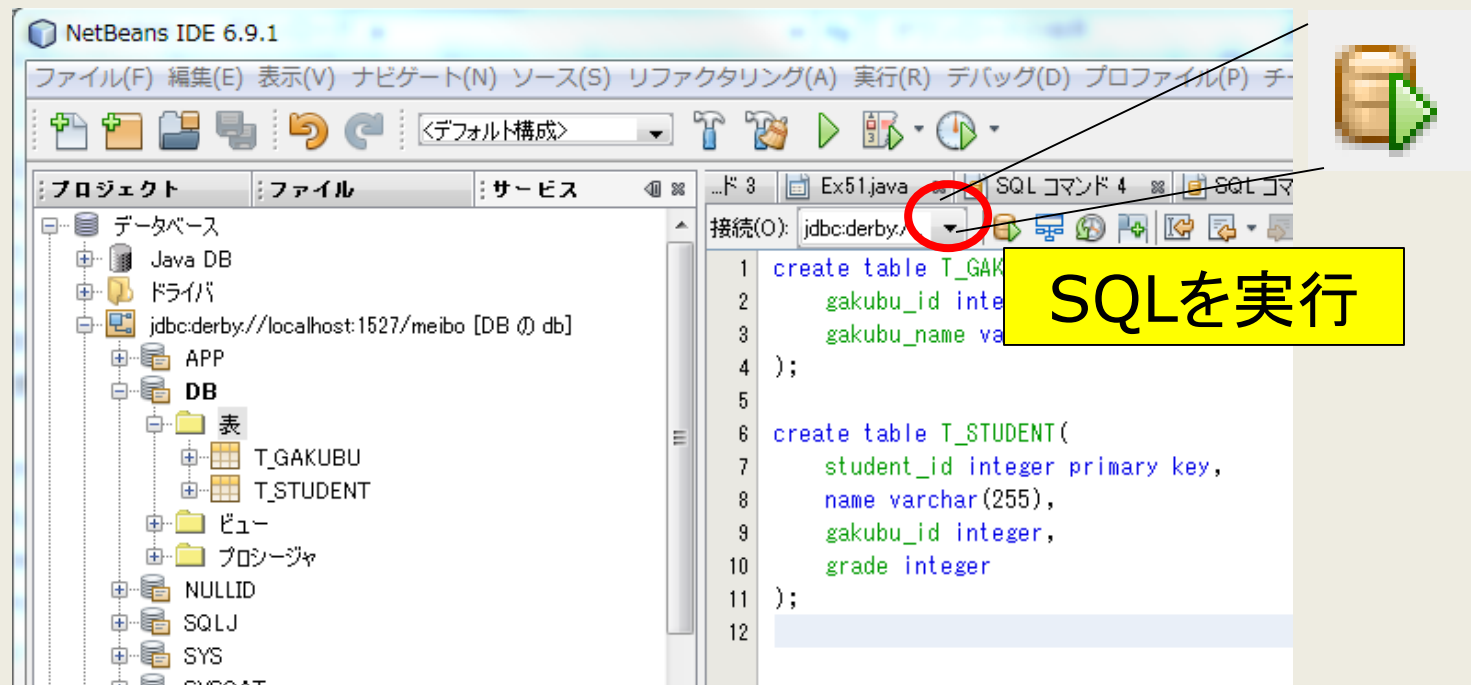
【テーブルを作成】  
create table テーブル名(  
 フィールド名 型 **主キー**,  
 ...  
 フィールド名 型  
);

主キーには「primary key」  
と付ける

【型】  
integer: 整数  
varchar: 文字列  
( )の中はバイト数を表す

# テーブルの作成(4)

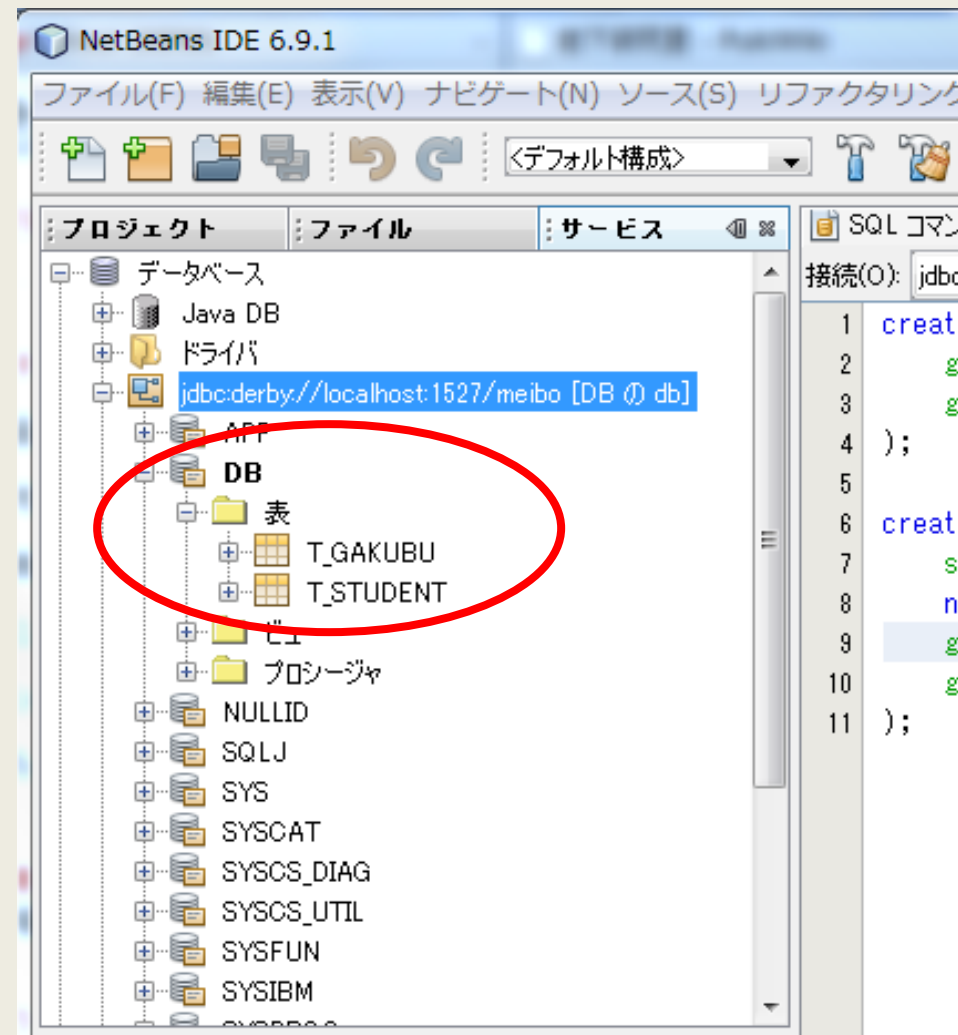
- エディタ上部の「SQLを実行」をクリックするとSQLが実行され、テーブルが作成される



- ※出力部分に「0個のエラーが発生しました」と出ていればOK
- ※エラーが発生している場合はSQL文が間違っている

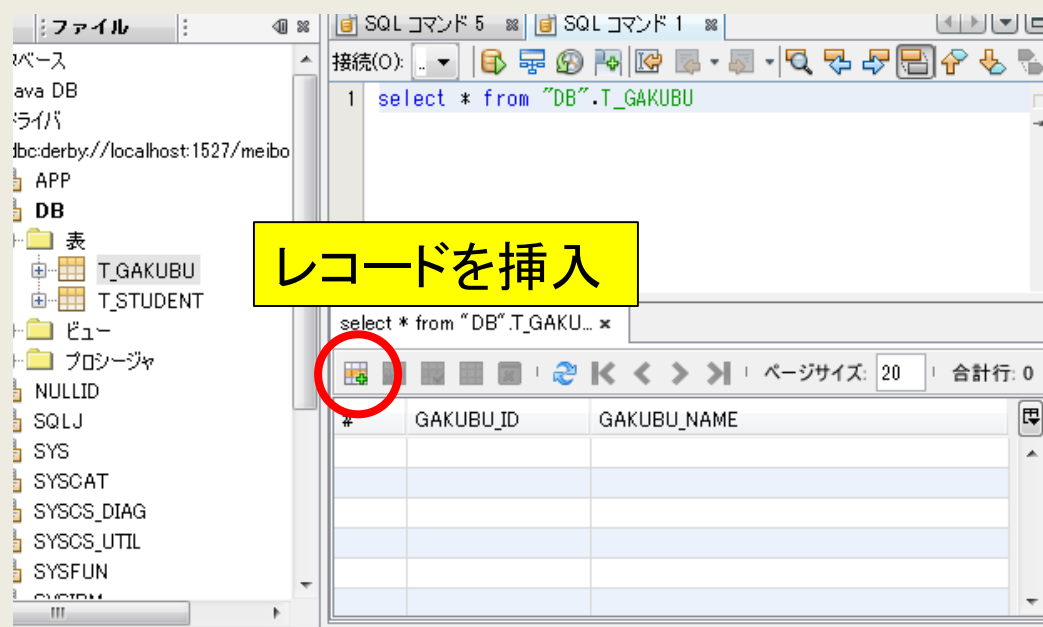
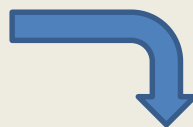
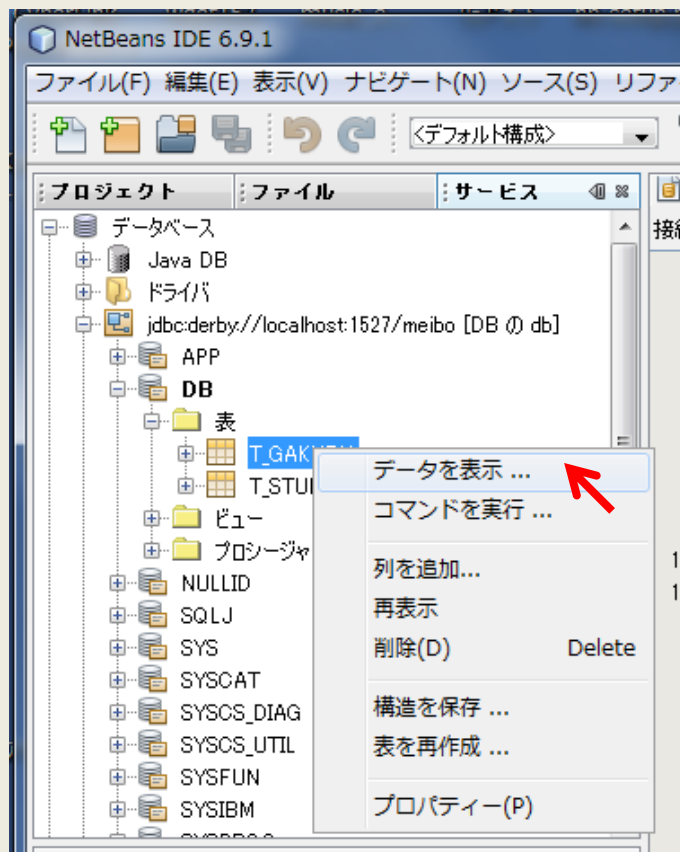
## 2-iii. テーブルにレコードを追加(1)

- 「接続」ノードで右クリック  
→「リフレッシュ」
- DBの下「表」の中に、  
「T\_GAKUBU」「T\_STUDENT」  
という2つのテーブルが  
作成されているのを確認



# テーブルにレコードを追加(2)

- 表「T\_GAKUBU」を右クリックし、「データを表示」をクリック
- 「レコードを挿入」でダイアログを表示





## テーブルにレコードを追加(3)

- ダイアログ内で  
データを入力
- 「行を追加」をクリックするとフィールドを追加

# テーブルにレコードを追加(4)

- 学部テーブル(T\_GAKUBU)に3つのレコードを追加

GAKUBU_ID	GAKUBU_NAME
1	コンピュータサイエンス
2	応用生物
3	メディア

# テーブルにレコードを追加(5)

- 学生情報テーブル(T\_STUDENT)に4つのレコードを追加

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1

# NetBeansでSQL文を実行

- 実行方法: テーブルに対してSQL文を記述して実行ボタンを押す

NetBeans IDE 6.9.1

②SQL文を実行

①SQL文を記述

③結果が反映される

#	STUDENT_ID	NAME	GAKUBU_ID
1		1 子門真人	
2		2 小田和正	
3		3 福山雅治	
4		4 安室奈美恵	
5		5 藤井フミヤ	
6			

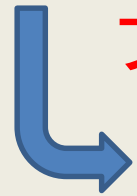
### 3. SQL文

## 3-i. SQL文: SELECTによるレコード抽出(1)

- select文: レコードの取り出し

select 抽出するフィールド from テーブル

例) select \* from T\_STUDENT




フィールドに「\*」を指定すると全てのフィールドを抽出

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1

## SQL文: SELECTによるレコード抽出(2)

例) `select student_id, fullname from T_STUDENT`

フィールドを複数指定するときは「,」で区切る



STUDENT_ID	FULLNAME
1	浜崎あゆみ
2	小田和正
3	福山雅治
4	安室奈美恵

STUDENT\_IDと  
FULLNAMEのみを表示

# SQL文: SELECTによるレコード抽出 ～検索～(1)

- where句: 条件を指定して絞り込み

select 抽出するフィールド from テーブル where 条件

例) select \* from T\_STUDENT where grade=1



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
2	小田和正	2	1
4	安室奈美恵	1	1

GRADEが1のレコード  
のみを表示



# SQL文: SELECTによるレコード抽出 ～検索～(2)

- 文字列の部分一致: **like**演算子
- 「**%**」は任意の文字列に一致する記号

例) `select * from T_STUDENT where fullname like '%雅治'`



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
3	福山雅治	3	3

「雅治」で  
終わる

例) `select * from T_STUDENT where fullname like '浜%'`



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4

「浜」で  
始まる

例) `select * from T_STUDENT where fullname like '%和%'`



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
2	小田和正	2	1

「和」を含む


# SQL文: SELECTによるレコード抽出 ～並べ替え～

- order by句: 並べ替え

select 抽出するフィールド from テーブル order by 並べ替えるフィールド

例) select \* from T\_STUDENT  
order by grade asc

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
2	小田和正	2	1
4	安室奈美恵	1	1
3	福山雅治	3	3
1	浜崎あゆみ	1	4

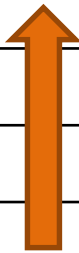


asc:  
省略可能

asc: 昇順に並べ替え  
※同じ場合は環境によって異なる

例) select \* from T\_STUDENT  
order by grade desc

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
3	福山雅治	3	3
2	小田和正	2	1
4	安室奈美恵	1	1



desc: 降順に並べ替え  
※同じ場合は環境によって異なる

# SQL文: SELECTによるレコード抽出 ～集計～(1)

- 集計関数を使うとレコードの集計ができる

## 集計関数

関数名	説明
count	レコード数
sum	合計
avg	平均
min	最小値
max	最大値

# SQL文: SELECTによるレコード抽出 ～集計～(2)

- 集計関数の使い方

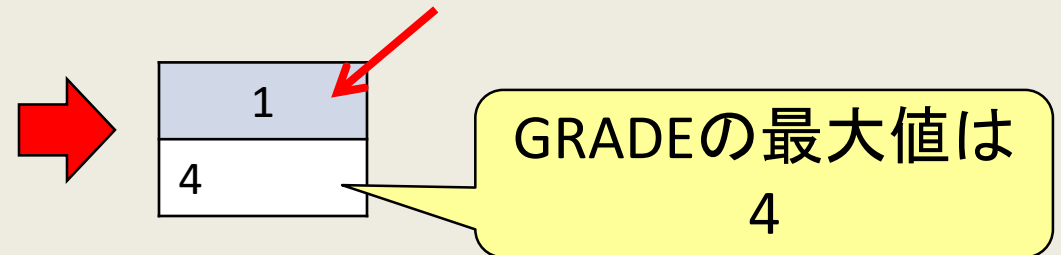
```
select 集計関数(フィールド) from テーブル
```

例) `select max(grade) from T_STUDENT`

T\_STUDENTテーブルのgradeの最大値を出力

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1

集計結果のフィールド名にはデフォルトで  
フィールド番号が入る



1	4
---	---

GRADEの最大値は  
4

# SQL文: SELECTによるレコード抽出 ～グループごと集計～(1)

- group by句: グループ分けして集計する

select 抽出するフィールド from テーブル group by グループ化の式

## ※注意

group by句を指定すると、  
抽出するフィールドには  
group by句で指定した式と  
集計関数のみが指定できる

# SQL文: SELECTによるレコード抽出 ～グループごと集計～(2)

例1) 学部IDごとに学年の最大値を出力

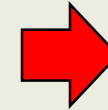
```
select gakubu_id, max(grade) from T_STUDENT  
group by gakubu_id
```

学部IDごと

gradeの最大値

集計結果のフィールド名にはデフォルトで  
フィールド番号が入る

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1



GAKUBU_ID	2
1	4
2	1
3	3

学部IDごとの  
GRADEの最大値

# SQL文: SELECTによるレコード抽出 ～グループごと集計～(3)

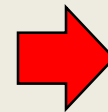
例2) 学部IDごとにレコード数を出力

```
select gakubu_id, count(*) from T_STUDENT  
group by gakubu_id
```

学部IDごと

レコード数(レコード数の場合は  
フィールドを指定する必要が無い  
ので「\*」を指定)

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1



GAKUBU_ID	レコード数
1	2
2	1
3	1

学部IDごとの  
レコード数

# SQL文: SELECTによるレコード抽出 ～グループごと集計～(4)

- 集計結果のフィールド名を変更できる

例) `select gakubu_id, count(*) as G_COUNT from  
T_STUDENT group by gakubu_id`



GAKUBU_ID	G_COUNT
1	2
2	1
3	1

集計結果のフィールドには  
デフォルトではフィールド番号が  
入ってしまうが  
「**as 名前**」で名前を付けられる



## 3-ii. SQL文：UPDATEによるレコードの変更(1)

- update文：レコードの変更

update テーブル set フィールド=値 [where 条件]

例) update T\_STUDENT set grade = grade + 1



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	5
2	小田和正	2	2
3	福山雅治	3	4
4	安室奈美恵	1	2

GRADEに1足す

※条件を指定しないと全てのレコードが変更される

# SQL文: UPDATEによるレコードの変更(2)

## ※条件を指定

例) `update T_STUDENT set grade = grade + 1  
where gakubu_id = 1`



学部IDが1の場合のみgradeに1足す

STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	5
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	2

GAKUBU\_ID=1  
のレコードのみ  
GRADEに1足す

### 3-iii. SQL文:INSERTによるレコードの追加

- insert文:レコードの追加

insert into テーブル(フィールド) values(値)

※文字列のみ「'」で囲む

例) insert into T\_STUDENT(student\_id, fullname, gakubu\_id, grade)  
values (5, '藤井フミヤ', 2, 2)



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
1	浜崎あゆみ	1	4
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1
5	藤井フミヤ	2	2

追加された  
レコード

### 3-iv. SQL文 : DELETEによるレコードの削除

- delete文 : レコードの削除

delete from テーブル [where 条件]

例) delete from T\_STUDENT where student\_id=1



STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE
2	小田和正	2	1
3	福山雅治	3	3
4	安室奈美恵	1	1
5	藤井フミヤ	2	2

STUDENT\_ID=1  
のレコードが  
削除された

### 3. バックアップ

# SQLの履歴を確認する方法

NetBeans IDE 6.9.1

ファイル(F) 編集(E) 表示(V) ナビゲート(N) ソース(S) リファクタリング(A) 実行(R) デバッグ(D) プロファイル

プロジェクト ファイル サービス

データベース

- Java DB
- ドライバ
- jdbc:derby://localhost:1527/meibo [DB の db]
- APP
- DB
  - 表
    - T\_GAKUBU
    - T\_STUDENT
  - ビュー
  - プロシージャ
- NIIT ID

SQL コマンド 1 SQL コマンド 2

接続(O): jdbc:derby://localho...

1 select \* from "DB".T\_STUDENT

SQL 履歴

接続: すべての接続 一致する SQL(M)

実行した SQL	実行...
select * from "DB".T_STUDENT	11/10/...
delete from T_STUDENT where student_id=1	11/10/...
insert into T_STUDENT(student_id, name, gakubu_id, grade) value...	11/10/...
update T_STUDENT set grade=grade-1	11/10/...
update T_STUDENT set grade=grade+1	11/10/...
select * from T_STUDENT, T_GAKUBU	11/10/...
select gakubu_id, count(*) as COUNT from "DB".T_STUDENT w...	11/10/...
select gakubu_id, count(*) as COUNT from "DB".T_STUDENT gr...	11/10/...
select gakubu_id, count(*) from "DB".T_STUDENT group by gaku...	11/10/...

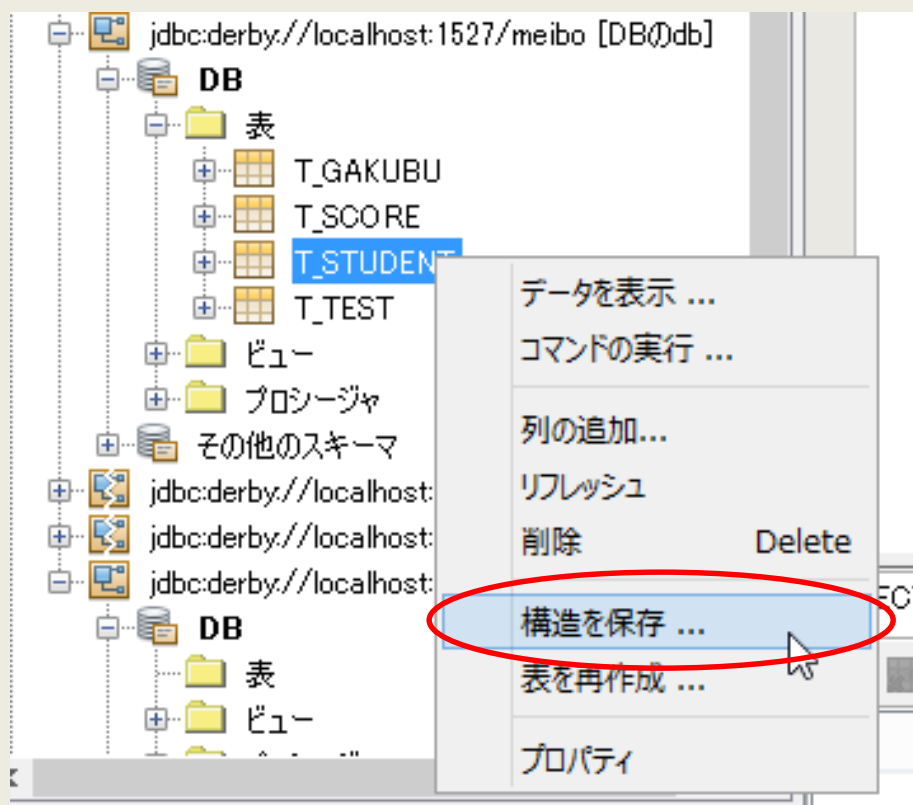
保存する SQL 文の数(N) 100 適用(A)

開じる ヘルプ(H)

SQL履歴

# テーブルのバックアップ(1)

- データベースの構造を保存
  - テーブルを右クリック⇒[構造を保存]



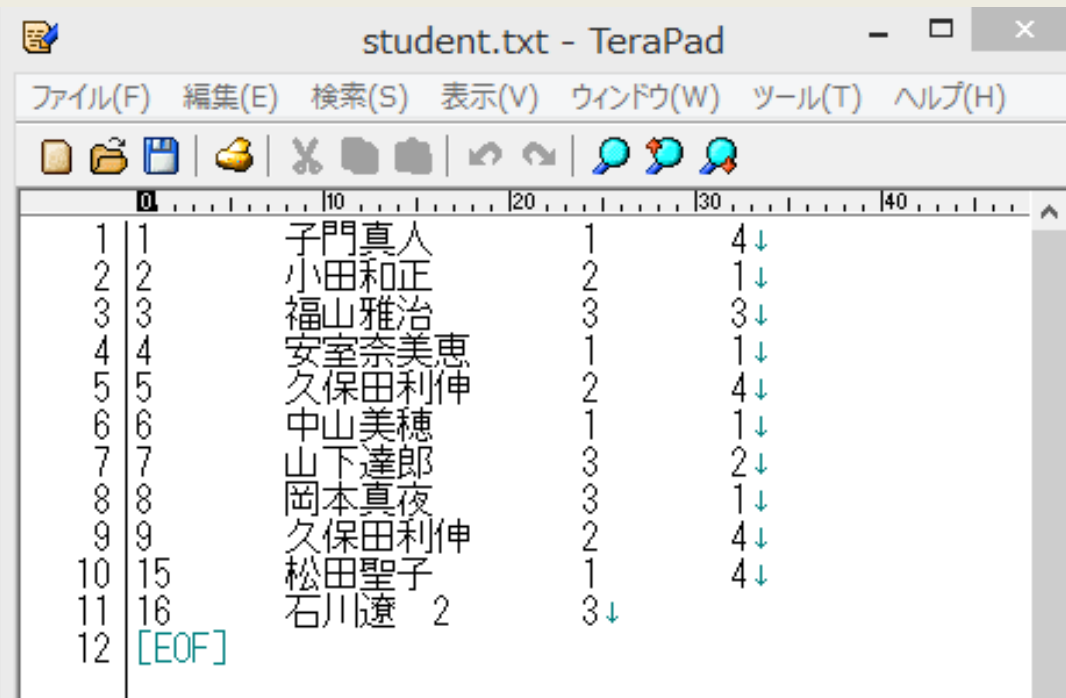
「T\_STUDENT.grab」  
という構造ファイルを保存

# テーブルのバックアップ(2)

- データベースの内容を保存
  - データを表示し、全てを選択してコピー
  - 別のファイル(テキストファイルやExcelファイル)に貼り付け



#	STUDENT_ID	FULLNAME
4	4	安室奈美恵
5	5	久保田利伸
6	6	中山美穂
7	7	山下達郎
8	8	岡本真夜
9	9	久保田利伸
10	15	松田聖子
11	16	石川遼

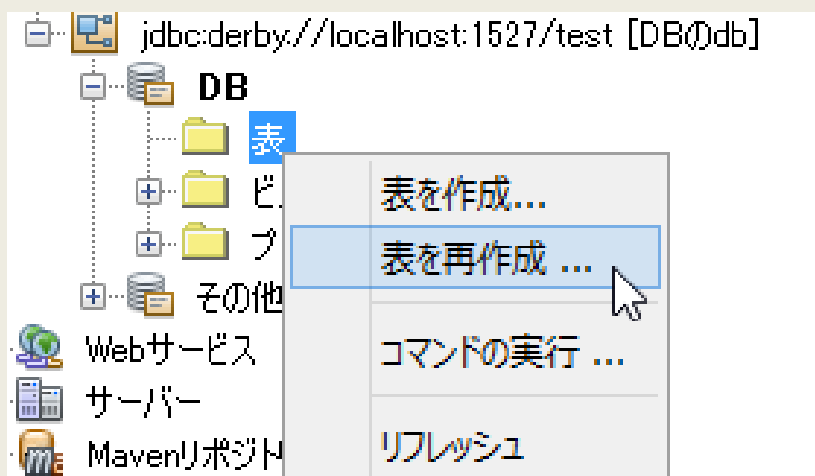


1	1	子門真人	1	4↓
2	2	小田和正	2	1↓
3	3	福山雅治	3	3↓
4	4	安室奈美恵	1	1↓
5	5	久保田利伸	2	4↓
6	6	中山美穂	1	1↓
7	7	山下達郎	3	2↓
8	8	岡本真夜	3	1↓
9	9	久保田利伸	2	4↓
10	15	松田聖子	1	4↓
11	16	石川遼	2	3↓
12	[EOF]			

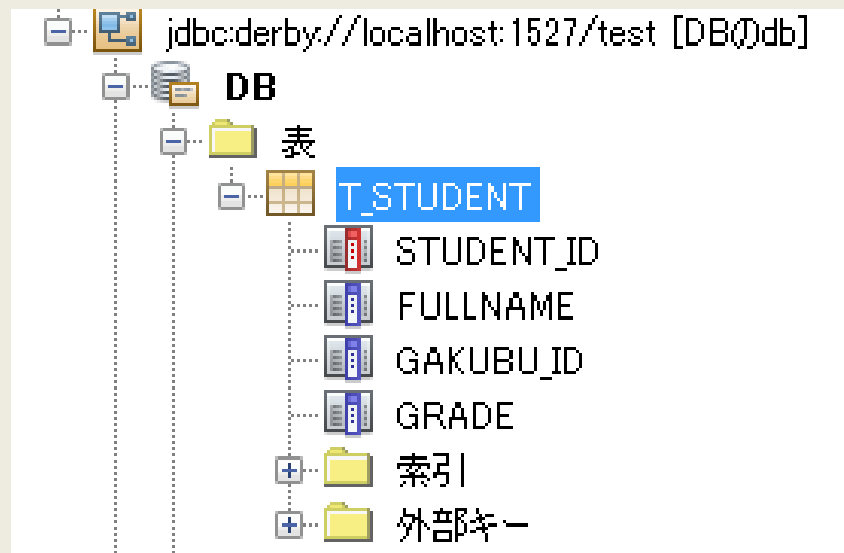


# テーブルのリストア(1)

- 新たなデータベースにテーブルを再作成
  - 表を作成したいデータベースの「DB」⇒「表」を右クリック⇒「表を再作成」
  - 保存した「.grab」ファイルを開く⇒「OK」



T\_STUDENT.  
grabを選択



## テーブルのリストア(2)

- 作成したテーブルにデータを貼り付け
  - データファイルを開き, 全てコピー
  - 新たなテーブルの編集画面を開き, 貼り付け

SELECT \* FROM "DB".T\_STUD... x

ページ・サイズ: 20 | 合計行: 0 | ページ: 1

#	STUDENT_ID	FULLNAME	GAKUBU_ID	GRADE

レコードを挿入

CTRL+Tabで表からのデータ入力モードから抜けます。与えられた列にCTRL+0でNU

#	STUDENT_ID	FULLNAME
1		1 子門真人
2		2 小田和正
3		3 福山雅治
4		4 安室奈美恵
5		5 久保田利伸

ここで貼り付け

貼り付け完了

# データベースの中身全てのバックアップ

- C:¥Users¥ユーザ名¥.netbeans-derby  
の各フォルダに, データベースが保存されている
- フォルダごとコピーして別の場所に置いておく
- リストアするとき
  - データベースのみNetBeansで新たに作成
  - 一旦NetBeansを閉じる
  - 新たなデータベースのフォルダの中身を全て削除
  - 保存しておいたフォルダの中身を全てコピーし, 新たなデータベースのフォルダに貼り付け