# 関係データベース言語
# ＳＱＬ（その１）



# 授業計画

第1回　ガイダンス・データベースの基本概念
第2回　データモデル
第3回　関係代数
第4回　データベース設計
第5回　リレーションの正規化
第6回　中間まとめ
第7回　関係データベース言語（SQL1）
第8回　関係データベース言語（SQL2）
第9回　計算機実習
第10回　データの検索機構　MySQL実習
第11回　トランザクション管理　MySQL実習
第12回　障害回復　MySQL実習
第13回　分散データベース　MySQL実習
第14回　期末まとめ
第15回　応用技術と将来動向　MySQL実習

## ６．関係データベース言語ＳＱＬ(その１)

# 講義内容

- SQLの概要
- データ定義言語（DDL）
  CREATE TABLE, CREATE VIEW
- データ操作言語（DML）
  SELECT, INSERT～VALUE, INSERT～SELECT,
  SELECT-FROM-WHERE-GROUP BY-HAVING-ORDER BY,
  UPDATE
- データ制御言語（DCL）
  GRANT, REVOKE
- メタデータとリポジトリ



## ６．関係データベース言語ＳＱＬ(その１)

# １．SQLの概要

- データベースの利用者にとって，関係データモデルのデータ操作を関係代数を用いて直接使用することは必ずしも適していない。
- 実用的なDBMSでは**データの更新**や**データベースの定義**(スキーマの定義)，**アクセス権の制御**，問合せの**集計処理**，**ソーティング**などが要求されるが，関係代数はそのような要求を満足させることはできない。
- SQLは，このような背景から開発された関係データモデルに基づく**標準のデータベース言語**である。
- 現在，ほとんどのRDBMSではSQLを用いてデータアクセスをサポートしているので，情報システムの開発・運用などでRDBMSに関わることになるならば，このSQLを避けて通ることは出来ない。

## ６．関係データベース言語ＳＱＬ(その１)

# １．SQLの概要

■ SQLの特徴

- **関係データモデルへの準拠：** 関係データベースモデルに基づいて定義された言語であること。
- **公的機関による標準化：** 公的機関により標準化されており，ほとんどのRDBMS製品が実際に使用している規格であること。
- **管理機能の実装：** データへのアクセスだけでなく，テーブルなどの管理機能も標準化していること。
- **セット処理への対応：** データに1件ずつアクセスするだけでなく，条件を満たすデータに対して一括して操作を行うことができる。



## ６．関係データベース言語ＳＱＬ(その１)

# １．SQLの概要

■ 関係データモデルとSQLの用語の対応関係

関係データモデルとSQLの用語の対応関係

| 関係データモデル | SQL |
|---|---|
| 関係(リレーション) | 表(テーブル) |
| 属性(アトリビュート) | 列 |
| タプル | 行 |
| 定義域(ドメイン) | 定義域(ドメイン) |

■ SQLが提供している3つの言語

① データ定義言語(DDL)
② データ操作言語(DML)
③ データ制御言語(DCL)

## １．SQLの概要

■ データ定義言語(DDL)

| 命　令 | 意　味 |
|---|---|
| CREATE TABLE | テーブルの作成 |
| CREATE VIEW | ビューの作成 |

■ データ操作言語(DML)

| 命　令 | 意　味 |
|---|---|
| SELECT | データの参照 |
| UPDATE | データの変更 |
| INSERT | データの追加 |
| DELEATE | データの削除 |

■ データ定義言語(DDL)

| 命　令 | 意　味 |
|---|---|
| GRANT | 権限の付与 |
| REVOKE | 権限の取消 |





## ２．データの定義言語(DDL)

■ DDLにはテーブル作成やビュー作成を行う命令がある。

■ テーブルは，実表，導出表およびビュー表に分類されている。

■ 実表は実際にデータベース中にデータが格納される表，導出表は問合せにより導出される表，ビュー表はビュー定義により定義される名前付きの導出表である。

■ 表定義

```
CREATE  TATBLE  テーブル名 (
  列名1      データ型      列制約,
  列名1      データ型      列制約,
   ・・・
  表制約
)
```

# ２．データの定義言語(DDL)

## ■ データ型

| 分　類 | データ型 | 説　明 |
|---|---|---|
| 文字列型 | CHAR（n）<br>NCHAR（n）<br>VARCHAR（n） | n文字の固定長文字列<br>n文字の国際化対応文字列<br>最大n文字の可変長文字列 |
| 数値型 | NUMBER（p, q）<br>INT<br>REAL | q桁の小数部を持つp桁の10進数<br>符号付きの整数<br>符号付きの浮動小数 |
| 日付／時間型 | DATE<br>TIME<br>TIMESTAMP | 日付（年月日）<br>時間（時分秒）<br>タイプスタンプ |





# ２．データの定義言語(DDL)

## ■ テーブルの制約

| 制約 | 説　明 |
|---|---|
| UNIQUE | 一貫性制約を定義する。<br>テーブルの中で値が一定であることが保障される。ただし，NULL値は複数の行に存在してもよい。 |
| NOT NULL | NULL値は許可しないという制約である。この制約を定義した列には必ず値を設定する必要がある。 |
| PRIMARY KEY | 主キー制約を定義する。<br>UNIQUEとNOT NULL制約の性質を持っている。 |
| CHECK | 設定可能な値の範囲などを指定できる。 |
| DEFAULT | 値を明示せずに行を追加する場合，自動的に設定される値を指定できる。 |
| FOREIGN KEY | 外部キー制約を定義する。<br>指定したテーブルの列に存在しない値を設定できないという制約をかけることができる。 |

## ２．データの定義言語(DDL)

（例）

商品

| 商品コード | 商品名 | 単価 | 仕入先コード |
|---|---|---|---|
| 1001 | ボールペン | 200 | S001 |
| 1002 | シャーペン | 150 | S002 |
| 1003 | ボールペン | 100 | S002 |
| 1004 | 定規 | 250 | S003 |
| 1005 | 消しゴム | 100 | S003 |

仕入先

| 仕入先コード | 仕入先名 |
|---|---|
| S001 | 田中商店 |
| S002 | 中村商店 |
| S003 | 斎藤商店 |

```
CREATE TATBLE 商品（
  商品コード     CHAR(4)  PRIMARY KEY,
  商品名         NCHAR(10)  NOT NULL,
  単価           INT,
  仕入先コード   CHAR(4),
  FOREIGN KEY （仕入先コード） REFERENCES （仕入先）,
  CHECK （ 単価 <= 500 ）
）
```





## ２．データの定義言語(DDL)

■ ビュー定義

```
CREATE VIEW ビュー名（列名1，列名2，・・・）
  AS SELECT ・・・
）
```

## ３．データの操作言語(DML)

■ DMLは，表内のデータの参照・更新・追加・削除などを行うための命令がある。

■ データの参照（SELECT文）

```
SELECT 列名1，列名2，・・・
FROM テーブル名1，テーブル名1，・・・
WHERE 抽出条件
GROUP BY グループ化を行う列
HAVING グループ化後の抽出条件
ORDER BY 並べ替え指定
```





## ３．データの操作言語(DML)

（使用例1）全ての列の抽出：
全ての列を取り出す場合には，“＊”が用いられる。

```
SELECT *
FROM 学生
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

## ３．データの操作言語(DML)

**（使用例2）特定列の抽出：**

**特定列を取り出す場合には，列名を並べればよい。指定した列順に抽出される。**

```
SELECT 学科, 名前
FROM 学生
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学科 | 名前 |
|---|---|
| 電気工学 | 前田 |
| 機械工学 | 田中 |
| 情報工学 | 大杉 |
| 情報工学 | 矢野 |
| 情報工学 | 佐藤 |





## ３．データの操作言語(DML)

**（使用例3）重複行の除外：**

**重複行を除外するには，DISTINCTが用いられる。**

```
SELECT DISTINCT 学科
FROM 学生
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学科 |
|---|
| 電気工学 |
| 機械工学 |
| 情報工学 |

## ３．データの操作言語(DML)

**（使用例4）抽出条件WHERE句の使用：**

**抽出条件を指定するには，WHERE句が用いられる。ここで，「N’情報工学’」はNCHARA型（国際化対応文字列）の定数の表記法である。**

```
SELECT　＊
FROM　学生
WHERE　学科 = N’　情報工学’
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学生番号 | 学科 | 名前 |
|---|---|---|
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |





## ３．データの操作言語(DML)

**（使用例5）グループ化GROUP　BY句の使用：**

**特定の列内で同じ値を持つデータをグループ化し，データを集計するためにはGROUP　BY句が用いられる。たとえば，科目別の平均点や合計点，店舗別の売上などの集計で使用される。**

```
SELECT　学科，COUNT(＊)
FROM　学生
GROUP　BY 学科
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学科 | COUNT(＊) |
|---|---|
| 電気工学 | 1 |
| 機械工学 | 1 |
| 情報工学 | 3 |

## ３．データの操作言語(DML)

**（使用例6）グループ化後の抽出条件HAVING句の使用：**

GROUP BY句でグループ化した結果に対して，抽出条件を指定するHAVING句が用いられる。たとえば，合計点数が500点以上の学生リストを作成したい場合などで使用される。

```
SELECT 学科, COUNT(*)
FROM 学生
GROUP BY 学科
HAVING COUNT(*) >= 3
```

学生

| 学生番号 | 学科 | 名前 |
|---|---|---|
| AA10001 | 電気工学 | 前田 |
| BA10002 | 機械工学 | 田中 |
| CA10003 | 情報工学 | 大杉 |
| CA10004 | 情報工学 | 矢野 |
| CA10005 | 情報工学 | 佐藤 |

⇒

| 学科 | COUNT(*) |
|---|---|
| 情報工学 | 3 |





## ３．データの操作言語(DML)

**（使用例7）並べ替え指定のORDER BY句の使用：**

ORDER BY句は，指定した列を昇順や降順に並べ替える場合に用いられる。昇順はASC，降順はDESCでソートキーの後に指定する。なお，指定を省略した場合はASCが指定されたことになる（デフォルトはASC）。使用例は，学科を昇順，点数を降順で並べ替えたものである。

```
SELECT *
FROM 試験成績
GROUP BY 学科, 点数 DESC
```

試験成績

| 学生番号 | 学科 | 点数 |
|---|---|---|
| AA10001 | B | 55 |
| BA10002 | C | 60 |
| CA10003 | A | 78 |
| CA10004 | A | 35 |
| CA10005 | A | 80 |

⇒

| 学生番号 | 学科 | 点数 |
|---|---|---|
| CA10005 | A | 80 |
| CA10003 | A | 78 |
| CA10004 | A | 35 |
| AA10001 | B | 55 |
| BA10002 | C | 60 |

# ３．データの操作言語(DML)

■ データの追加（INSERT文）

- テーブルに新しくデータを追加するには，INSERT文を用いる。INSERT文には，値を指定して1行ずつデータを追加するINSERT～VALUES文と，別のテーブルのデータを追加するINSERT～SELECT文がある。

■ INSERT～VALUES文の構文

```
INSEERT INTO テーブル名（列名1，列名2，・・・）
VALUES （値1，値2，・・・）
```

■ INSERT～SELECT文の構文

```
INSEERT INTO テーブル名（列名1，列名2，・・・）
SELECT　列名1，列名2，・・・
FROM　　参照テーブル名
WHERE　条件
```





# ３．データの操作言語(DML)

■ INSERT～VALUES文の使用例

```
INSEERT INTO 科目（科目番号，科目名）
VALUES （05，’アルゴリズム’）
```

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 02 | OS |
| 03 | プログラミング |
| 04 | データベース |

⇒

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 02 | OS |
| 03 | プログラミング |
| 04 | データベース |
| 05 | アルゴリズム |

## ３．データの操作言語(DML)

■ **データの変更（UPDATE文）**
■ テーブルにすでに存在するデータを変更するには，UPDATE文を用いる。

■ UPDATE文の構文

```
UPDATE  テーブル名
  SET 列名1 = 値1，列名2 = 値2，・・・
WHERE  条件
```





## ３．データの操作言語(DML)

```
UPDATE  科目
  SET 科目名=  ” Cプログラミング’
WHERE  科目番号   = ’ 03’
```

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 02 | OS |
| 03 | プログラミング |
| 04 | データベース |

⇒

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 02 | OS |
| 03 | Cプログラミング |
| 04 | データベース |

## ３．データの操作言語(DML)

- **データの削除（DELETE文）**
- テーブルからデータを削除するには，DELEATE文を用いる。
- DELEATE文の構文

```
DELEATE  FROM  テーブル名
WHERE  条件
```

```
DELEATE  FROM  科目
WHERE  科目番号  =’02’
```

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 02 | OS |
| 03 | プログラミング |
| 04 | データベース |

⇒

科目

| 科目番号 | 科目名 |
|---|---|
| 01 | 論理回路 |
| 03 | プログラミング |
| 04 | データベース |





## ４．データの制御言語(DCL)

- DCLには，ユーザ毎のテーブルやビューへのアクセス権などを定義する命令がある。
- **権限の付与（GRANT文）**
  - テーブルやビューには，データの参照は許可するが，変更は許可しないなどの権限をユーザ毎に設定できる。

| 権　限　名 | 説　明 |
|---|---|
| SELECT | 参照 |
| UPDATE | 変更 |
| INSERT | 追加 |
| DELETE | 削除 |
| ALL （ALL PRIVULEGES） | すべての権限 |

## ４．データの制御言語(DCL)

■ GRANT文の構文

```
GRANT 権限名　　ON　テーブル名　TO　ユーザ
```

（例1）user01というユーザに「商品」表のデータ参照と変更の権限を与えるGRANT文

```
GRANT  SELECT  UPDATE  ON  商品  TO  user01
```

（例2）全てのユーザに「商品」表のデータ参照の権限を付与するGRANT文

```
GRANT  SELECT  ON  商品  TO  PUBLIC
```





## ４．データの制御言語(DCL)

■ 権限の取消(REVOKE文)

- ユーザから権限を取り消すには，REVOKE文を用いる。

■ REVOKE文の構文

```
REVOKE  権限名　 ON  テーブル名　FROM  ユーザ
```

（例）user01というユーザに「商品」表のデータ変更の権限を取り消すREVOKE文

```
REVOKE  UPDATE  ON  商品  FROM  user01
```

## ５．メタデータとリポジトリ

- **メタデータ**は，データに関する情報ことで，メタデータもデータである。
- データベース中のデータに関する情報である“表定義”，“ビュー定義”や“権限定義”などの情報はメタデータと呼ばれている。
- RDBMSではデータベース中のデータは表として格納されるが，メタデータもそれぞれの情報毎に表として格納して管理されている。
- これらの各表を**データ辞書表**または**データディクショナリ表**とよび，これらの表の集合を**データ辞書**または**データディクショナリ**という。
- RDBMSでは，データとメタデータを同じ構造で格納しているので，ユーザはメタデータも通常のデータと同じようにアクセスすることができる。





## ５．メタデータとリポジトリ

- **リポジトリは**，データやファイルを蓄積する場所を意味する語としていろいろな場面で用いられている。
- ソフトウエア開発では多数のソース・ファイルのバージョンを管理しながら蓄積する**CVS**（Concurrent Versions System）などの**バージョン管理システム**が利用される。
- CVSでは，中央のリポジトリにソース・ファイルなどを一括して蓄積しておき，プログラムを修正する場合にはそれぞれの開発者がリポジトリからファイルをローカルにコピーし，その後リポジトリに修正結果を戻すことが行われている。
- RDBMSのデータ辞書もメタデータを格納したリポジトリとも考えられる。

## まとめ

- **SQLの概要**
- **データ定義言語（DDL）**
  CREATE TABLE, CREATE VIEW
- **データ操作言語（DML）**
  SELECT, INSERT～VALUE, INSERT～SELECT,
  SELECT-FROM-WHERE-GROUP BY-HAVING-ORDER BY,
  UPDATE
- **データ制御言語（DCL）**
  GRANT, REVOKE
- **メタデータとリポジトリ**