**〇注意すること**

・2つの表を用いて条件をかけるときは直積からの内部結合よりも自然結合からのWHERE文のほうが書きやすい。直積からの内部結合だと共通している列をWHERE文で指定しなければならないのでめんどう

**〇基本**

・「CREATE TABLE 表名( 表の中身 );」で表を作る

・列(カラム)の定義は「列名 型」とする。後ろにNOT NULLをつけると空白を禁止にできる

例、科目表

CREATE TABLE 科目 (

科目番号 CHAR(3) NOT NULL,

科目名 VARCHAR(30) NOT NULL,

単位数 INT

);

科目

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |

・主キーと外部キーを設定するときに単一カラムのとき

行(カラム)の定義に続けて記述できる。主キーは「列名 型 PRIMARY KEY」、外部キーは「列名 型 REFERENCES 別の表名(列名)」

・主キーと外部キーを設定するときに複数カラムのとき

行(カラム)の定義とは別に独立に主キーと外部キーを定義する必要がある。主キーは「PRIMARY KEY(列名1, 列名2)」、外部キーは「FOREIGN KEY(列名1, 列名2) REFERENCES 別の表名(列名1, 列名2)」

例、履修登録

CREATE TABLE 履修登録 (

科目番号 CHAR(3) NOT NULL,

学籍番号 CHAR(6) NOT NULL,

セメスター INT,

PRIMARY KEY ( 科目番号, 学籍番号 ),

FOREIGN KEY (科目番号) REFERENCES 科目(科目番号),

FOREIGN KEY (学籍番号) REFERENCES 学生(学籍番号),

CHECK ( 1 <= セメスター AND セメスター<= 12 )

) ;

・CREATE TABLEの中で定義域を設けることができる。「CHECK (条件)」とする

・テーブルの破棄は「DROP TABLE 表名;」

**〇データ操作**

・データを追加するときは「INSERT INTO 表名 VALUES (値１, 値2, …, ), (値1, 値２, …, ), …, (値1, 値2, …, );」とする。VALUESの後ろにタプルを並べて書くことができる

例、科目表に値を3行追加

INSERT INTO 科目 VALUES

('001', 'データベース', 2),

('002', 'プログラミング', 1),

('003', 'ネットワーク', 2);

科目

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |
| 002 | プログラミング | 1 |
| 003 | ネットワーク | 2 |

・データを参照するときは「SELECT 列名1, 列名2, FROM 表名1, 表名2, WHERE 条件;」とすると指定した列を「射影」して、条件に合う行を「選択」できる

例、科目表から2行目の科目名列のみ取り出す

SELECT 科目名 FROM 科目 WHERE 科目番号= '002’ ;

科目

|  |
| --- |
| 科目名 |
| プログラミング |

・SELECT文では「SELECT \* FROM 表名」のように列名を指定するところで「\*」を使うことで、すべての列を取り出せる

例、科目表からすべてのデータを参照する

SELECT \* FROM 科目 ;

科目

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |
| 002 | プログラミング | 1 |
| 003 | ネットワーク | 2 |

・データの更新をするときは「UPDATE 表名 SET 更新事項 WHERE 条件」とする

例、科目表の科目名のプログラミングをプログラミング演習にして単位数を2にする

UPDATE 科目

SET 科目名 = 'プログラミング演習', 単位数 = 2 WHERE 科目番号 = '002';

科目

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |
| 002 | プログラミング演習 | 2 |
| 003 | ネットワーク | 2 |

・データを削除するときには「DELETE FROM 表 WHERE 条件」で条件に合う列を削除する

例、科目表から科目番号003の列を消去する

DELETE FROM 科目 WHERE 科目番号 = '003';

科目

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |
| 002 | プログラミング演習 | 2 |

**〇表の変形**

・重複行の削除には「SELECT DISTINCT 文」

例、成績表の科目番号から重複行を削除する

成績表

|  |  |  |
| --- | --- | --- |
| 科目番号 | 学籍番号 | 得点 |
| 001 | TB0001 | 90 |
| 001 | TB0002 | 80 |
| 002 | TB0001 | 70 |
| 002 | TB0003 | 90 |

SELECT DISTINCT 科目番号 FROM 成績表;

|  |
| --- |
| 科目番号 |
| 001 |
| 002 |

・行を昇順、降順に並び替えるときはSELECT文の中で「ORDER BY ～ ASC/DESC」とする。昇順はASC、降順はDESC、何も指定しないと昇順になる

例、成績表を得点で昇順にする

SELECT \* FROM 成績 ORDER BY 得点;

|  |  |  |
| --- | --- | --- |
| 科目番号 | 学籍番号 | 得点 |
| 002 | TB0001 | 70 |
| 001 | TB0002 | 80 |
| 001 | TB0001 | 90 |
| 002 | TB0003 | 90 |

・テーブルの和、差、積をとるときはそれぞれUNION, EXCEPT, INTERSECTを用いてSELECT文２つ（2つの表）で囲む。「(SELECT ～) UNION (SELECT ～);」

**〇表の結合**

・直積は2つの表をすべて組み合わせる。表名をコンマで区切る。「SELECT \* FROM 表1, 表2;」

例、科目表(3行)と成績表(4行)の直積をとると12行になる

・内部結合は2つの表の直積に対して条件をかけて一致する行のみ選択する。条件文はWHERE の後ろにくっつける「SELECT \* FROM 表１, 表２ WHERE 条件;」

例、成績表と科目表の科目番号が一致する行を選択する

SELECT \* FROM 科目, 成績 WHERE 科目.科目番号 = 成績.科目番号;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 科目番号 | 科目名 | 単位数 | 科目番号 | 学籍番号 | 得点 |
| 001 | データベース | 2 | 001 | TB0001 | 90 |
| 001 | データベース | 2 | 001 | TB0002 | 80 |
| 002 | プログラミング | 1 | 002 | TB0001 | 70 |
| 002 | プログラミング | 1 | 002 | TB0003 | 90 |

・自然結合は2つの表の共通する列をすべてくっつける「SELECT \* FROM 表１ NATURAL JOIN 表２;」

例、成績表と科目表を自然結合

SELECT \* FROM 科目 NATURAL JOIN 成績;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 科目番号 | 科目名 | 単位数 | 学籍番号 | 得点 |
| 001 | データベース | 2 | TB0001 | 90 |
| 001 | データベース | 2 | TB0002 | 80 |
| 002 | プログラミング | 1 | TB0001 | 70 |
| 002 | プログラミング | 1 | TB0003 | 90 |

・指定した列のみ自然結合するときは「SELECT \* FROM 表１ JOIN 表2 USING(列名);」

例、上と同じように成績表と科目表を自然結合

SELECT \* FROM 科目 JOIN 成績 USING(科目番号);

・外部結合は自然結合する際に条件に一致しない行も残してNULL値で埋める。左の表に合わせる左外部結合「LEFT OUTER JOIN」と右の表に合わせる右外部結合「RIGHT OUTER JOIN」とどちらの表にも合わせる「FULL OUTER JOIN」がある。USINGで結合する列を指定する必要がある

例、科目表に合わせて成績表を科目番号で外部結合

SELECT \* FROM 科目 LEFT OUTER JOIN 成績 USING(科目番号);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 科目番号 | 科目名 | 単位数 | 学籍番号 | 得点 |
| 001 | データベース | 2 | TB0001 | 90 |
| 001 | データベース | 2 | TB0002 | 80 |
| 002 | プログラミング | 1 | TB0001 | 70 |
| 002 | プログラミング | 1 | TB0003 | 90 |
| 003 | ネットワーク | 2 | NULL | NULL |

**〇副問合せ**

・副問合せとはSELECT文の中に別のSELECT文を入れて、入れ子にして実行すること。入れ子の中身（副問合せ）の表を先に作って、そのあとに主問合せの表を考える

例、成績表の得点が80より小さい科目番号に当てはまる科目表の科目番号

SELECT 科目名 FROM 科目

WHERE 科目番号 in (SELECT 科目番号 FROM 成績 WHERE 得点 < 80);

副問合せ結果（薄字は実行結果ではない）

|  |  |  |
| --- | --- | --- |
| 科目番号 | 学籍番号 | 得点 |
| 002 | TB0001 | 70 |

主問合せ結果（薄字は実行結果ではない）

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 002 | プログラミング | 1 |

・副問合せの結果が上の例のように1行の場合比較演算子を使うことができる

例、科目表の科目名がデータベースである科目番号と成績表の科目番号が一致している行の学籍番号と得点を表示

SELECT 学籍番号, 得点 FROM 成績WHERE 科目番号 =

(SELECT 科目番号 FROM 科目 WHERE 科目名 = ‘データベース’);

副問合せ結果

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |

主問合せ結果

|  |  |  |
| --- | --- | --- |
| 科目番号 | 学籍番号 | 得点 |
| 001 | TB0001 | 90 |
| 001 | TB0002 | 80 |

・副問合せの結果が単一列複数行のとき、その行のいずれかに該当している行を選択するときは「列名1 = any (SELECT 列名2 FROM 表2)」とできる

例、成績表に登録してある科目番号の科目名を科目表から取り出す

SELECT 科目名 FROM 科目

WHERE 科目番号 = any (SELECT 科目番号 FROM 成績);

副問合せ結果

|  |  |  |
| --- | --- | --- |
| 科目番号 | 学籍番号 | 得点 |
| 001 | TB0001 | 90 |
| 001 | TB0002 | 80 |
| 002 | TB0001 | 70 |
| 002 | TB0003 | 90 |

主問合せ結果

|  |  |  |
| --- | --- | --- |
| 科目番号 | 科目名 | 単位数 |
| 001 | データベース | 2 |
| 002 | プログラミング | 1 |

**〇集約関数**

・集約関数とは列の複数の値から1つの値を求める関数。件数、合計、平均など集計や統計処理に利用することができる。

・件数のカウント「count(\*) AS 新列名」「count(列名) AS 新列名」列名に\*を指定するとNULLもカウントして、列名を入れるとNULLを除外してカウントする

例、成績表から科目番号001の受講者数を調べる

SELECT count(\*) AS 受講者数 FROM 成績 WHERE 科目番号 = ‘001’;

|  |
| --- |
| 受講者数 |
| 2 |

・列名の前に「DISTINCT」を指定すると重複のない行数をカウントできる

例、成績表から受講者総数（重複なし）を調べる

SELECT count(DISTINCT 学籍番号) AS 受講者総数 FROM 成績;

|  |
| --- |
| 受講者総数 |
| 3 |

・他の代表的な集約関数は以下の通り

|  |  |
| --- | --- |
| 最大値 | max() |
| 最小値 | min() |
| 平均 | avg() |
| 合計 | sum() |
| 件数 | count() |

・集約関数は慣例で小文字で書くことが多いが大文字でも書くことができる

・集約関数を使用した場合に重複した行を1つにまとめるグループ化をすることができる「GROUP BY 列名」

例、成績表から科目ごとの受講者数と平均点をだす

SELECT 科目番号, count(学籍番号) AS 受講者数, avg(得点) AS 平均点 FROM 成績;

GROUP BY 科目番号

|  |  |  |
| --- | --- | --- |
| 科目番号 | 受講者数 | 平均点 |
| 001 | 2 | 85 |
| 002 | 2 | 80 |

・GROUP BYでグループ化した後の表に対して条件を「HAVING 条件」と指定することができる

例、学籍番号ごとの受講科目数を求めて、結果が2以上のもののみ学籍番号と合わせて表示する

SELECT 学籍番号, count(\*) AS 受講科目数 FROM 成績

GROUP BY 学籍番号 HAVING count(\*) >= 2; # count(\*)は受講科目数でもOK

|  |  |
| --- | --- |
| 学籍番号 | 受講科目数 |
| TB001 | 2 |

・注意点：GROUP BYを使用したSELECTで指定できるのは集約関数を適用した結果とGROUP BYで指定した項目のみ

→できる環境もあるが、指定していない列を追加するとグループ化したときにその列に表示される値が1つの値にまとめられておかしく見えるはず。

例、得点をGROUP BYで指定していないのでエラーになる

SELECT 学籍番号, 得点 FROM 成績 GROUP BY 学籍番号;

**〇その他**

・型

|  |  |  |
| --- | --- | --- |
| **区分** | **データ型** | **詳細** |
| 数値 | INTEGER | 整数値（- 2,147,483,648 ～+ 2,147,483,647） |
| NUMERIC(DECIMAL) | 小数点を含む整数 |
| 文字列 | CHAR | 固定長の文字列 |
| VARCHAR | 可変長の文字列 |
| 日付 | DATE | 日付 |
| TIMESTAMP | 日付と時間 |
| 論理値 | BOOLEAN(BIT) | TRUE / FALSE判定 |

・コメントアウトは1行のときは「--」で囲み、複数行のときは「/\*」と「\*/」で囲む

-- コメント --