1. RDB の操作

1.1 データ検索

1.1.1 ある表にある全てのデータを見る

【構文】 `Select * from 表名`

《例》表「受注明細」の全てのデータを表示する。

`Select * from 受注明細`

【Exp 1】表「商品」の全てのデータを表示してください。

1.1.2 ある表の特定の列を見る

【構文】 `Select 列名 1, 列名 2, ... from 表名`

《例》表「社員」の列名が社員コード、社員名、であるデータを表示する。

`Select 社員コード, 社員名 from 社員`

【Exp 2】表「社員」の列名が社員名、住所、電話番号のデータを表示して下さい。

1.1.3 条件指定による検索

【構文】 `Select * from 表名 Where 条件 (And または Or 条件)`

【構文】 `Select 列名 1, 列名 2, ... from 表名 Where 条件
(And または Or 条件)`

1.1.3.1 ある値と等しいものを検索する

《例》表「商品」の列値 単価 = 300 の 商品番号、商品名 を表示する。

`Select 商品番号, 商品名 from 商品 Where 単価 = 300`

検索条件は、列のデータ型によって記述が異なります。

<文字型>

variant char, char などの文字型を比較するときは、値をシングルコーテーション「'」でくくります。大文字、小文字も正しく入力する必要があります。

<数値型>

数値型のタイプを比較するときは、値をそのまま記述します。

<日付型>

Date など日付型はシングルコーテーション「'」でくくります。ただし、日付の表現は幾通りかありますので、複数の書き方があります。

Date 型の場合、'99/07/01' でも'99-07-01'、'99/7/1' でもかまいません。

【注意】 データベースによっては 'yyyy-mm-dd' でないとダメなものもあります

《例》表「顧客」の列名 顧客名 の値の最後に商会がつくデータを表示する。

```
Select * from 顧客 where 顧客名 like '%商会'
```

《例》表「顧客」の列名 住所 の値に千代田区が含まれるデータを表示する。

```
Select * from 顧客 where 住所 like '%千代田区%'
```

【Exp 7】表「顧客」の列名 電話番号 の値が 092 で始まるデータを表示して下さい。

1.1.3.5 値の集合により検索する

【構文】 Where 列名 IN (値 1, 値 2, ……)

列の内容が括弧中のどれかの値と等しければ検索対象になります。

《例》表「商品」の列名 商品名 の値が システム0型 と ディスク1型 のデータを表示する。

```
Select * from 商品 where 商品名 in ('システム0型', 'ディスク1型')
```

《例》表「商品」の列名 商品名 の値が システム0型 と ディスク1型 以外のデータを表示する。

```
Select * from 商品 where 商品名 not in ('システム0型', 'ディスク1型')
```

【Exp 8】表「顧客」の列名 顧客コード の値が C005 のデータを表示して下さい。

⇒注目：in の()内のリストは select 文の結果を指定することもできます。のちに説明するテーブルのジョインと同じ効果をもたらします。

【Exp 8-1】in のリスト指定を「受注明細で数量が 20 以上の商品番号を検索」として商品名を表示できるようにして下さい。

【Exp 8-2】受注がある社員名を表示して下さい。

⇒注目：distinct 文を使うと重複を排除できます。

```
select 社員コード from 受注 → select distinct 社員コード from 受注
```

1.1.3.6 データを並べ替えて表示する

【構文】 Order By 列名 1, 列名 2, ……

複数の列名を指定すると、最初の列名が第一キーとなり昇順で並び替えます。

降順で並び替えたいときは、列名の後に DESC とつけます。

《例》表「商品」を第一キー 単価 で昇順に並び替えてデータを表示する。

Select * from 商品 order by 単価

《例》表「商品」を第一キー 単価 で昇順に、第二キー 商品番号 で降順に並び替えてデータを表示する。

Select * from 商品 order by 単価, 商品番号 DESC

Where 句を記述するときは、Order By 句の前に記述します。

《例》**Select * from 商品 Where 単価 < 900 order by 単価**

1.1.4 検索内容を加工する

1.1.4.1 演算子を使って計算する

《例》表「商品」の列名 単価 の値を $\times 1.05$ にして表示する。

Select 商品番号,商品名,単価*1.05 from 商品

1.1.4.2 関数を使用して計算する

ROUND (数値、桁数) : 四捨五入 TRUNC (数値、桁数) : 切り捨て

《例》表「商品」の列名 単価 の値を $\times 1.05$ にして円以下四捨五入して表示する。

Select 商品番号,商品名,単価*1.05, ROUND(単価*1.05, 0) from 商品

【Exp 9】表「商品」の列名 単価 の値を $\times 1.05$ にして円以下切り捨てて表示して下さい。

関数は、数値関数、文字列関数、日付関数、集計関数等様々なものがあります。

1.1.4.3 列名の表示を変更する

SQLには、検索時に列のタイトルを変更して表示し、利用者に解りやすいものに指定する機能があります。

【構文】 **Select 列名 as 表示したい列名, 列名 as 表示したい列名, ... From ...**

《例》表「商品」の列名 単価 の値を $\times 1.05$ にして円以下四捨五入した列を 税込価格 として表示する。

Select 商品番号,商品名,単価, ROUND(単価*1.05, 0) as 税込価格 from 商品

この場合、[税込価格] 列名の上にスペースを入れた場合 [税込 価格] をダブルコーテーションでくくります。

【Exp 10】表「商品」の列名 単価 の値を $\times 1.05$ にして円以下切り捨てた列を 税込価格 端数切捨 として表示して下さい。

1.1.4.4 条件判断を行い表示する

SQL の中で場合わけを行うことができます。

```
【構文】 ① Select 列名 as 表示したい列名, 列名 as 表示したい列名,  
          CASE 列名  
            WHEN 値式 THEN 値式  
            ELSE 値式  
          END  
          FROM 表名  
② Select 列名 as 表示したい列名, 列名 as 表示したい列名,  
          CASE  
            WHEN 探索条件 THEN 値式  
            ELSE 値式  
          END  
          FROM 表名
```

《例》表「商品」の列名 単価 の値が 300 円の商品は一律 10%の値上げを行い新単価とする。

```
Select 商品番号,商品名,単価, CASE 単価 WHEN 300 THEN 単価*1.1 ELSE  
      単価 END as 新単価 from 商品
```

【Exp 11】表「商品」の列名 商品番号 の値が 1000 番台のプリンタは一律 10%の値下げを行い値下げ単価として表示して下さい。

1.1.5 グループ処理（グループ集計）

集計関数（例えば SUM 関数等）は、ひとつの結果しか返さないが、部門ごとの集計等複数の結果を返すようにするための SQL 文があります。

```
【構文】 Group By 列名
```

《例》単純な集計

表「受注明細」の列名 受注番号 の値が 102 番の数量の合計と件数を表示する。

```
Select SUM(数量) as 受注番号 102 の数量合計,COUNT(*) as 明細件数  
      From 受注明細 Where 受注番号=102
```

《例》グループ毎の集計

表「受注明細」の列名 受注番号 毎に数量の合計と明細件数を表示する。

```
Select 受注番号,SUM(数量) as 数量合計,COUNT(*) as 明細件数  
      From 受注明細 GROUP By 受注番号
```

【Exp 12】 上記例を数量合計で昇順に並べて表示して下さい。
グループ化した項目に対する検索条件指定

【構文】 HAVING 条件

《例》 上記例で数量合計が 5 より大きいものを表示する。

```
Select 受注番号,SUM(数量) as 数量合計,COUNT(*) as 明細件数
      From 受注明細 GROUP By 受注番号
      Having SUM(数量) > 5
```

【Exp 13】 表「受注明細」の列名 受注番号 毎に数量の合計と明細件数を表示する際に
数量合計が 5 より大きいものを受注番号順に表示して下さい。

SELECT ステートメントの主な句の記述順を以下にまとめます。

```
SELECT select_list
[INTO new_table_]
FROM table_source
[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC] ]
```

1.2 表の結合

1.2.1 二つの表の結合（等結合：内部結合：INNER JOIN）

通常、データベースでは、複数の表を組み合わせて初めて見やすい表示形式になります。そこで、2つのテーブルに共通のフィールドで関連付け、表を結合します。その結果、2つの表が1つの表のイメージになるので、今までのように検索条件を指定できるようになります。

等結合は、最も一般的な結合で、Where句に次のように書きます。

【構文】 Where 表名. 列名 = 表名. 列名

《例》表「社員」と表「家族」をお互いの表にある列名 社員コード で結合して社員名、住所、家族コード、家族名を表示する。

```
Select 社員.社員名,社員.住所,家族.家族コード,家族.家族名
from 社員,家族
where 社員.社員コード = 家族.社員コード
```

```
Select 社員.社員名,社員.住所,家族.家族コード,家族.家族名
from 社員 Inner Join 家族 on 社員.社員コード = 家族.社員コード
```

(列名（フィールド名）が各々の表で異なれば、表名は省略できます。)

1.2.2 二つの表の結合（外部結合：LEFT JOIN,RIGHT JOIN,FULL JOIN）

等結合では互いの結合列で合致するものだけを取り出していました。これは言い換えると一方の表には存在するが、他方には存在しないものは結果の表には現れないということになります。そこでどちらか一方の表にデータがあれば取り出す方法を考えてみます。

相手先の表に対応するレコードがあるなしにかかわらず、全てのレコードを抽出する結合を外部結合といいます。例えば「得意先マスター」と「売上」表において売上があるなしにかかわらず全得意先を抽出するといった場合に使います。

外部結合には、左右に結合する表のうち左にだけ存在するものを結びつける左外部結合（反対を右外部結合）、左と右の外部結合を両方含む全外部結合があります。

《例》表「社員」と表「家族」において、家族がない社員についても社員コードで結合して、社員名、住所、家族コード、家族名を表示する。

【構文】 From 全データを取り出す表名 LEFT JOIN 相手方の表名
ON 結合条件

※これでは家族のいない山田一郎さんは表示されません

```
Select 社員.社員名,社員.住所,家族.家族コード,家族.家族名
from 社員,家族
where 社員.社員コード = 家族.社員コード
```

RIGHT
FULL

※よって外部結合を使います。

```
Select 社員.社員名,社員.住所,家族.家族コード,家族.家族名
      from 社員 LEFT Join 家族 on
      社員.社員コード = 家族.社員コード
```

1.2.3 三つ以上の表の結合

where 句に and 演算子を用いて 3 つ以上の表を決合することも可能です。

《例》表「受注」と表「受注明細」と表「商品」において、受注番号と商品番号でそれぞれの表を結合して受注番号、受注明細番号順に、受注番号、受注明細番号、商品名、単価、数量、金額を表示する。

```
Select 受注.受注番号,受注明細.受注明細番号,商品.商品名,商品.単価,受注明細.数量,
      商品.単価*受注明細.数量 as 金額
      from 受注,受注明細,商品
      where 受注.受注番号 = 受注明細.受注番号 and
            受注明細.商品番号 = 商品.商品番号
      Order by 受注.受注番号,受注明細.受注明細番号
```

【Exp 14】表「受注」と表「受注明細」と表「商品」と表「顧客」において、受注番号と商品番号と顧客コードでそれぞれの表を結合して受注番号、受注明細番号順に、受注番号、顧客名、受注明細番号、商品名、単価、数量、金額を表示して下さい。

【Exp 15】上記の条件にさらに受注した社員の情報も担当者として顧客名の前に付け加えて表示して下さい。

1.3 データ登録・更新・削除

前節では、すでにあるデータの検索について基本的なことを説明しました。この節では、データの登録・更新・削除について説明します。

1.3.1 レコードの追加

新しいレコード（行）を追加するには、INSERT 文を使用します。

```
【構文】 INSERT INTO 表名 (列名 1, 列名 2, ……)
        VALUES (値 1, 値 2, ……)
```

括弧の中に表の列名を記述します。次の VALUES 句の後の括弧の中に、先に指定した列の順番で値を記述します。指定していない列には、デフォルト値が指定してあればそのデフォルト値が、そうでなければ NULL が代入されます。値の記述で、文字列と日付はシングルクォーテーション ['] でくくります。NULL を禁止している列に値を指定しないと (NULL を代入する) INSERT 文は実行できません。

すべての列に値を代入する場合は、列名を省略できます。この場合、表の列の順序で値を記入しなければいけません。

```
【構文】 INSERT INTO 表名 VALUES (値 1, 値 2, ……)
```

《例》表「商品」に次のようなレコードを追加する。

```
INSERT INTO 商品(商品番号,商品名,単価)
VALUES(3001,'システム 1',500)
```

次ぎの SQL 文を実行して、結果を確認してみましょう。

```
Select * From 商品
```

1.3.2 レコードの更新

すでにあるレコードの更新をするには、UPDATE 文を使用します。

```
【構文】 UPDATE 表名 SET 更新列名 1 =更新値, 更新列名 2=更新値, ……
        Where 条件式
```

Where 句で条件が設定されていないと、SET で指定した列名を全て更新してしまいます。したがって、どのレコードを更新するかという指定が重要です。1レコードだけ更新することも、複数レコードまとめて更新することも可能です。

《例》表「商品」のレコードを次のように変更する。

```
UPDATE 商品 SET 商品番号= 4001,商品名='ディスク X'  
Where 商品番号=2001
```

この更新はうまくいきません。表「受注明細」の外部キー設定における参照制約で「更新：制限付」の定義がなされているためです。そこで「更新：波及」になっている表「社員」の社員コードを変更してみます。

```
UPDATE 社員 SET 社員コード= 'M004'  
Where 社員コード='M002'
```

次ぎの SQL 文を実行して、結果を確認してみましょう。

```
Select * From 社員
```

更新が波及したテーブルの状態も確認してみます。

```
Select * From 家族  
Select * From 受注
```

1.3.3 レコードの削除

レコードを削除するには、DELETE 文を使用します。

【構文】 DELETE FROM 表名 Where 条件式

Where 句で削除するレコードを指定する条件を指定します。条件にマッチする複数のレコードが削除されます。Where 句の指定がないと、全てのレコードが削除されます。注意が必要です。

《例》表「部門」の営業二課のデータを削除する。

```
DELETE FROM 部門 Where 部門コード='E02'
```

次ぎの SQL 文を実行して、結果を確認してみましょう。

```
Select * From 部門
```

波及する表「社員」の内容も合わせて確認しましょう（列名 部門コード の該当列値が Null になっているはずです）。

```
Select * From 社員
```