東京ITスクール
TOKYO IT SCHOOL

データの登録と検索

目次

1. はじめに ____ 1
2. データの登録 ____ 2
3. データの検索 ____ 5

1. はじめに

ここからは、データベース操作の基本となる CRUD の基本を学習します。CRUD とは、Create（登録）、Read（検索）、Update（変更）、Delete（削除）の頭文字をとったものです。今回はその中でも Create（登録）、Read（検索）について見ていきましょう。

まずは、Employee テーブルと Dept テーブルを作成してください。以下に Employee テーブルと Dept テーブルの CREATE 文を示します。

（サンプルコード）CREATE TABLE 文

```
CREATE TABLE Dept(
   dept_id NUMBER(2) PRIMARY KEY,
   dept_name VARCHAR2(20) NOT NULL
);

CREATE TABLE Employee (
   emp_id NUMBER(5) PRIMARY KEY,
   emp_pass VARCHAR2(10) DEFAULT '9999' NOT NULL,
   emp_name VARCHAR2(20) NOT NULL,
   gender NUMBER(1) NOT NULL,
   address VARCHAR2(30),
   birthday DATE,
   dept_id NUMBER(2) NOT NULL,
   CONSTRAINT f1 FOREIGN KEY (dept_id) REFERENCES Dept(dept_id)
);
```

2. データの登録

1　INSERT 文の基本構文

テーブルにデータを登録する事を「挿入」と呼びます。データの挿入には INSERT 文を使用します。INSERT 文は、DML 文の一種です。
INSERT 文の基本構文は、次の通りです。

（構文）INSERT 文

```
INSERT INTO <テーブル名> (列 1,列 2,列 3,……) VALUES(値 1,値 2,値 3,……);
```

テーブル名の後の列リストと、VALUES 句の値リストは、列数が一致している必要があります。次のように、数が不一致だとエラーになって挿入出来ません。

（サンプルコード）エラーになる INSERT 文

```
INSERT INTO Employee (emp_id, emp_pass) values (1,'pass', 'system shared');
```

また、INSERT 文は、基本的に 1 回で 1 行を挿入します。従って、複数の行を挿入したい場合は、原則的にその行数だけ INSERT 文も繰り返し実行する必要があります。

2　列リストの省略

テーブル名の後の列リストは、テーブルの全列に対して INSERT を行なう場合、省略することが出来ます。このとき、VALUES 句の値が暗黙のうちに、左から順に各列に割り当てられます。
以下の 2 つの構文は同じデータを挿入します。

（サンプルコード）列リストの省略

```
--列リストあり
INSERT INTO Employee (emp_id, emp_pass, emp_name, gender, address, birthday,
dept_id) values (1,'1234', 'system shared', 1, '千葉県', '2009/09/09', 1);

--列リストなし
INSERT INTO Employee values (1,'1234', 'system shared', 1, '千葉県', '2009/09/09',
1);
```

それでは実際にデータを挿入してみましょう。

（サンプルコード）INSERT 文

```
INSERT INTO Dept values (1,'総務部');
INSERT INTO Dept values (2,'営業部');
INSERT INTO Dept values (3,'経理部');
INSERT INTO Dept values (4,'資材部');

INSERT INTO Employee values (1,'1111', 'system shared', 1, '千葉県',
'2009/09/09', 1);
INSERT INTO Employee values (2,'2222', 'systemsss', 1, '松戸市',
'2009/09/10', 1);
INSERT INTO Employee values (3,'3333', 'administrator', 2, '東京都',
'2009/09/10', 2);
INSERT INTO Employee values (4,'4444', 'edudbuser', 2, '埼玉県',
'2009/09/14', 3);
```

また、登録したデータを保存するために COMMIT を実行しましょう。詳しくは後ほど学びますが、まずは登録したデータをしっかりと保存する機能だと思ってください。データの登録、変更、削除を行った後は基本的にこちらを実行しなければ、データが失われてしまいます。

（サンプルコード）コミット

```
COMMIT;
```

3　特殊な値の挿入

NULL を挿入する

INSERT 文で、ある列に NULL を割り当てたい場合は、VALUES 句の値リストに NULL をそのまま記述します。

ただし、NULL を割り当てられる列は、NOT NULL 制約のついていない列に限られます。NOT NULL 制約のついている列に NULL を指定した場合、INSERT 文はエラーとなり、データの挿入に失敗します。

デフォルト値を挿入する

列の制約としてデフォルト値が設定されていた場合、自動的にそれを INSERT 文の例の値として利用することが出来ます。その利用方法には、「明示的な方法」と「暗黙的な方法」の 2 種類があります。

①明示的にデフォルト値を挿入する

VALUES 句に、DEFAULT キーワードを指定します。

（サンプルコード）明示的にデフォルト値を挿入

```
INSERT INTO Employee values (5, DEFAULT, 'system555', 1, '千葉県',
'2009/09/09', 1);

COMMIT;
```

②暗黙的にデフォルト値を挿入する

暗黙的にデフォルト値を挿入する場合は、デフォルト値が設定されている列を、列リストからも VALUES からも省略します。

（サンプルコード）暗黙的にデフォルト値を挿入

```
INSERT INTO Employee (emp_id, emp_name, gender, address, birthday, dept_id)
values (6, 'system666', 1, '千葉県', '2009/09/09', 1);

COMMIT;
```

また、デフォルト値が設定されていない列を省略した場合は、NULL が割り当てられます。従って、NOT NULL 制約がつけられている列を省略すると、INSERT 文はエラーになります。

3. データの検索

1　列を出力する

テーブルからデータを取り出すときには、SELECT 文を使います。また、SELECT 文で必要なデータを検索し、取り出すことを「問い合わせ」あるいは「クエリ（query）」と呼ぶこともあります。

SELECT 文は、数ある SQL 文の中で最も多用され、かつ最も基本となる SQL 文です。SELECT 文をマスターすることは、SQL をマスターすることに繋がります。

SELECT 文の基本的な構文は次の通りです。

（構文）基本的な SELECT 文

```
SELECT <列名>, ……
  FROM <テーブル名>;
```

SELECT 句には、テーブルから出力したい列の名前を、表示させたい順番に、カンマ区切りで並べて指定します。FROM 句には、データを取り出すテーブルの名前を指定します。

例として、Emloyeep テーブルから emp_id（社員 ID）列、emp_name（社員名）列、address（住所）列を出力してみましょう。

Employee テーブル

emp_id	emp_pass	emp_name	gender	address	birthday	dept_id
1	1111	system shared	1	千葉県	2009/09/09	1
2	2222	systemsss	1	松戸市	2009/09/10	1
3	3333	administrator	2	東京都	2009/09/10	2
4	4444	edudbuser	2	埼玉県	2009/09/14	3
5	9999	system555	1	千葉県	2009/09/09	1
6	9999	system666	1	千葉県	2009/09/09	1

実行する SQL 文と実行結果を以下に示します。

（サンプルコード）SELECT 文

```
SELECT emp_id, emp_name, address FROM Employee;
```

（実行結果）

```
EMP_ID    EMP_NAME             ADDRESS
---------- -------------------- ------------------
1          system shared        千葉県
2          systemsss            松戸市
3          administrator        東京都
4          edudbuser            埼玉県
5          system555            千葉県
6          system666            千葉県

6 行選択されました
```

2　全ての列を出力する

全ての列を出力したいときには、全ての列を意味するアスタリスク（*）を使用します。

（構文）全ての列を出力

```
SELECT *
  FROM <テーブル名>;
```

例えば、Employee テーブルの全ての列を出力する場合には、次のように記述します。

（サンプルコード）全ての列を出力

```
SELECT * FROM Employee;
```

実行結果は次の通りです。

（実行結果）

```
EMP_ID    EMP_PASS             EMP_NAME             GENDER    ADDRESS   BIRTHDAY             DEPT_ID
---------- -------------------- -------------------- ---------- ---------- -------------------- ----------
1          1111                 system shared        1         千葉県    09-09-09             1
2          2222                 systemsss            1         松戸市    09-09-10             1
3          3333                 administrator        2         東京都    09-09-10             2
4          4444                 edudbuser            2         埼玉県    09-09-14             3
5          9999                 system555            1         千葉県    09-09-09             1
6          9999                 system666            1         千葉県    09-09-09             1

6 行選択されました
```

ただし、アスタリスクを使うと、結果の列の並び順を指定することは出来ません。この場合は、CREATE TABLE 文で定義したときの順番で列が並びます。

3　列に別名をつける

SQL 文では、AS キーワードを使って、列に別名をつけることが出来ます。

（構文）AS キーワード

```
SELECT <列名> AS <別名>, ……
  FROM <テーブル名>;
```

例えば、「emp_id」列を「id」という別名で出力する場合には、次のように記述します。

（サンプルコード）AS キーワード

```
SELECT emp_id AS id FROM Employee;
```

実行結果は次の通りです。

（実行結果）

```
ID
----------
1
2
3
4
5
6

6 行選択されました
```

別名には日本語を使うことも出来ます。その場合には、別名をダブルクォーテーション（「"」）で囲みます。シングルクォーテーション（「'」）ではないことに注意しましょう。

4　定数の出力

SELECT 句には、列名だけでなく定数を書くことも出来ます。定数が文字列や日付の場合は、シングルクォーテーション（'）で囲むことを忘れないでください。

例えば、4 列目に定数として会社名「システムシェアード」を出力する場合には、次のように記述します。

（サンプルコード）定数

```
SELECT
emp_id, emp_name, address, 'システムシェアード' AS comp_name
FROM Employee;
```

（実行結果）

```
EMP_ID     EMP_NAME                ADDRESS                 COMP_NAME
---------- ----------------------- ----------------------- ---------------------------------
1          system shared           千葉県                  システムシェアード
2          systemsss               松戸市                  システムシェアード
3          administrator           東京都                  システムシェアード
4          edudbuser               埼玉県                  システムシェアード
5          system555               千葉県                  システムシェアード
6          system666               千葉県                  システムシェアード

6 行選択されました
```

5　結果から重複行を省く

例えば、Employee テーブルにどんな住所（address）が登録されているのかを調べたいときに、重複するデータを省いて表示出来ると便利です。

Employee テーブル

emp_id	emp_pass	emp_name	gender	address	birthday	dept_id
1	1111	system shared	1	千葉県	2009/09/09	1
2	2222	systemsss	1	松戸市	2009/09/10	1
3	3333	administrator	2	東京都	2009/09/10	2
4	4444	edudbuser	2	埼玉県	2009/09/14	3
5	9999	system555	1	千葉県	2009/09/09	1
6	9999	system666	1	千葉県	2009/09/09	1

重複を省く

address
千葉県
松戸市
東京都
埼玉県

このように重複行を省いて結果を得たい場合は、SELECT 句で DISTINCT というキーワードを使用します。DISTINCT キーワードは先頭の列名の前に記述します。

（構文）DISTINCT キーワード

```
SELECT DISTINCT <列名>, ……
   FROM <テーブル名>;
```

例えば、「address」列から重複を省いて出力する場合には、次のように記述します。

（サンプルコード）DISTINCT キーワード

```
SELECT DISTINCT address FROM Employee;
```

実行結果は次の通りです。

（実行結果）

```
ADDRESS
----------
東京都
松戸市
千葉県
埼玉県
```

DISTINCT を使用するとき、NULL も 1 種類のデータとして扱われますので注意してください。複数の行に NULL がある場合には、やはり 1 つの NULL にまとめられます。

また、DISTINCT は、複数の列の前におくことも出来ます。この場合、複数の列を組み合わせてもなお重複する行が 1 つにまとめられます。

6　WHERE 句による行の選択

ここまでは、テーブルに格納されている全ての行を選択していました。しかし、実際には毎回全ての行を取り出したい訳ではありません。むしろ「部署が総務部の社員」など、何らかの条件に合う行だけを選択したい場合がほとんどです。

SELECT 文では、選択したい行の条件を WHERE 句で指定します。WHERE 句には、「ある列の値がこの文字列と等しい」「ある列の値がこの数値以上」などの条件を指定することが出来ます。

SELECE 文では、WHERE 句を次のように記述します。

（構文）SELECT 文の WHERE 句

```
SELECT <列名>,……
  FROM <テーブル名>
 WHERE<条件式>;
```

下図は、Employee テーブルから「部署 ID（dept_id）が　1　」の行を選択している様子です。

Employee テーブル

emp_id	emp_pass	emp_name	gender	address	birthday	dept_id
1	1111	system shared	1	千葉県	2009/09/09	**1**
2	2222	systemsss	1	松戸市	2009/09/10	**1**
3	3333	administrator	2	東京都	2009/09/10	2
4	4444	edudbuser	2	埼玉県	2009/09/14	3
5	9999	system555	1	千葉県	2009/09/09	**1**
6	9999	system666	1	千葉県	2009/09/09	**1**

「dept_id」列が　「1」　の行を選択

選択した行からは、好きな列を出力することが出来ます。今回は「emp_name」列と「dept_id」列を出力してみます。このときの SELECT 文は以下のようになります。

（サンプルコード）WHERE 句

```
SELECT emp_name, dept_id FROM Employee WHERE dept_id = 1;
```

WHERE 句にある「dept_id　=　1」が検索条件を表わす式（条件式）です。「=」は両辺が等しいかどうかを比較するための記号で、この条件では「dept_id」列の値と「1」を比較し、等しいかどうかを調べます。比較は、Emp テーブルの全ての行に対して行われます。

また、今回の例では正しく選択されているかどうかを確認するため、SELECT 句で検索条件になっている「dept_id」列を出力していますが、これは必須ではありません。出力したい列が「emp_name」のみであれば、「emp_name」列だけを出力することも出来ます。
実行結果は次の通りです。

（実行結果）

```
EMP_NAME            DEPT_ID
-------------------- -----------------
system shared       1
systemsss           1
system555           1
system666           1
```

7　コメントの書き方

コメントとは、SQL 文に説明や注意事項などを付記するものです。コメントは、SQL の実行には一切影響を及ぼしません。そのため、アルファベットでも漢字でも好きに使うことが出来ます。
コメントを付与する方法には、次の 2 つがあります。

●1 行コメント
「--」の後に記述します。1 行の中でしか書けません。

●複数行コメント
「/*」と「*/」で囲った中に記述します。複数行に渡って書けます。