UPDATE文 データの更新

# UPDATE文の基本構文

UPDATE 表名

SET 列名 = 値 [ , 列名 = 値 ]

[WHERE 条件式]

UPDATE文は表に存在するデータを書き換えるための命令文です。

１行目のUPDATE句の後に更新したい表を指定します。２行目のSET句には更新したい列名とその列に書き込むデータを指定します。

ハンズオン　店舗名を更新する

１．コマンドプロンプトを起動して、「mysql –u dbuser –p studb」でMySQLに接続を行う。

２．店舗名を更新するUPDATE文を実行する。

UPDATE STORE

SET SNAME = '神戸店';

３．更新した内容を確認するため、SELECT文を実行する。

SELECT \* FROM STORE;

４．ロールバック処理を実行する。※後程説明します。

ROLLBACK;

UPDATE文は条件を指定しないと、表にあるデータを全て更新してしまいます。

対象の行のみ更新する場合は、WHERE句で対象行の絞り込みを行います。

５．今度は三宮店のみを神戸店に更新をして、更新後のデータを確認してください。

※確認出来れば、ROLLBACKを実行してください。

DELETE文 データの削除

# DELETE文の基本構文

DELETE FROM 表名

[WHERE 条件式]

DELETE文は表に存在するデータを削除するための命令文です。DELETE文もWHERE句に条件を指定しないと全件削除になります。全てのデータが削除されてしまうのでWHEREを指定しないDELETE文の扱いには注意が必要です。

ハンズオン　店舗データを削除する

１．生駒店のデータを削除するDELETE文を実行する。

DELETE FROM STORE

WHERE STORE\_NO = 003;

２．削除した内容を確認するため、SELECT文を実行する。

SELECT \* FROM STORE;

３．ロールバック処理を実行する。

ROLLBACK;

４．大阪府の店舗データを削除して、削除後のデータを確認してください。

※確認出来れば、ROLLBACKを実行してください。

INSERT文 データの追加

# INSERT文の基本構文

INSERT INTO 表名 ( 列名 [ , 列名 …] )

VALUES (値 [ , 値 …] )

INSERT文は表に新しいデータを追加するための命令文です。１行目のINSERT INTOの後に表名を指定して、後ろのカッコ内に対象列を指定します。２行目のVALUES句の後に対象列に追加するデータを指定します。列名と値の数は一致する必要があります。

全ての列にデータを入れる場合は、列名を指定するカッコは省略可能です。

ハンズオン　店舗情報を追加する

１．店舗表に桂川店を追加するINSERT文を実行する。

INSERT INTO STORE ( STORE\_NO, SNAME, PREFECTURES, STATUS )

VALUES ( '005', '桂川店', '京都府', '未オープン' );

２．京都店を追加するINSERT文を実行します。今回は全ての列を指定するので

列名を指定せずにデータの追加をしてください。

|  |  |
| --- | --- |
| STORE\_NO | 006 |
| SNAME | 京都店 |
| TEL | 0753613215 |
| PREFECTURES | 京都府 |
| ADDRESS | 京都市下京区東塩小路町７２１−１ |
| OPEN\_TIME | 10:00 |
| CLOSE\_TIME | 22:00 |
| HOLIDAY | 土日 |
| STATUS | 未オープン |

２．追加した内容を確認するため、SELECT文を実行する。

３．ロールバック処理を実行する。

トランザクション制御

# トランザクション

トランザクションとは複数のSQL文をひと固まりとして扱う論理的な単位になります。

皆さんは店舗情報を使ってデータベースの更新、削除、挿入を学習しました。今回の場合は１つの命令文でも特に問題は無いのですが、情報を正確に扱うにはトランザクションを意識しなければなりません。銀行の振込処理を例に上げると、会社からアルバイトに給与を振り込む時に会社の口座のお金を減らす処理とアルバイトの口座のお金を増やす処理は必ずセットで実行されなければなりません。

・銀行システムの口座情報

アルバイト代

7万円

|  |  |  |
| --- | --- | --- |
| 口座番号 | 名義 | 残高 |
| 1000123 | 三宮店 | \13,400,000 |
| 1001001 | 高山一実 | \325,000 |

UPDATE 口座情報 SET 残高 = 残高 – 70,000 WHERE 口座番号 = '1000123';

UPDATE 口座情報 SET 残高 = 残高 + 70,000 WHERE 口座番号 = '1001003';

この２つのSQL文はセットで実行しないとデータの辻褄が合わなくなります。

# トランザクションの中断

データベースはコンピュータ上で動いているため１つ目と２つ目のSQL文の間でエラーなどにより処理が中断される可能性があります。その場合、1つ目のSQL文のみ有効になるとデータの不整合が発生します。なのでデータベースにはトランザクションに含まれるすべてのSQL文は、**「全て実行が完了している」か「１つも実行されていない」どちらかの状態に制御する機能**が必須になります。

# 原子性確保のしくみ

トランザクションに含まれるSQL文を分けずに扱う性質のことを**原子性**(atomicity)といいます。この仕組みを実現するためにDBMSではコミットとロールバックという機能が備わっています。

|  |  |
| --- | --- |
| コマンド | 機能 |
| COMMIT | 変更したデータを確定してトランザクションを終了する。 |
| ROLLBACK | 変更したデータを破棄してトランザクションを終了する。 |

主キーとその必要性

# 特定の行を識別する方法

授業で使用している表に従業員表(EMPLOYEE)があります。

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | ENAME | BIRTHDAY | … |
| 00001  00002  00003 | 小林友治  小林圭輔  友保隼平 | 1972-07-04  1986-03-06  1985-08-11 | …  …  … |

この表には従業員番号(EMP\_NO)がありますが、何の為にあるのでしょうか？

２行目の小林圭輔さんの情報を取得するためには、どのような条件が必要でしょうか。名前？誕生日？その他の項目？いずれも同名や同じ誕生日の従業員が存在した場合に複数のデータが選択されてしまいます。そこで必要なのが従業員番号です。従業員番号は一人一人に必ず割り当てられていて、同じ値はありません。このように「**値を指定することで1件のデータが完全に特定できる**」役割を担う列を**主キー(PRIMARY KEY)**と呼びます。

データベースは情報を保管するためにあるのではなく、情報を管理するために利用されます。その為にはデータを正しく識別できる必要があります。基本的に**表は主キーとなる列が必要**になります**。**

# 複数の列で行を識別する

先程は従業員番号列を作ることで１件のデータを特定しました。しかしデータベースには単独では１件にはならないが、複数列を組み合わせれば重複しない場合があります。

従業員表を例に上げると、同名の従業員は存在する可能性はあるが氏名、生年月日、住所が組み合わされば１件のデータに特定することが出来ます。このように、**複数列を１つの主キーとして扱うものを複合主キー**といいます。