カーソル

# カーソルとは

カーソルとは、**検索結果のデータを一時的に保持しておく仮想的な作業領域**のことです。

ストアドプロシージャやファンクション内で複数データに対して**１行ごとに処理を行いたい場合に利用します**。

カーソルを利用するためには、「**カーソルの宣言**」、利用を始める「**カーソルのオープン**」、１行データの取得を行う「**フェッチ**」、カーソルを破棄する「**カーソルのクローズ**」を行う必要があります。書式は以下の通りです。

・カーソルの基本構文

|  |  |
| --- | --- |
| 処理 | 書式 |
| カーソルの宣言 | DECLARE カーソル名 CURSOR FOR SELECT文 ; |
| カーソルの開始処理 | OPEN カーソル名 ; |
| １行データの取得 | FETCH カーソル名 INTO 変数名 ; |
| カーソルの終了処理 | CLOSE カーソル名 ; |



カーソルの宣言で処理対象データの指定を行い、

OPEN命令を実行すればカーソルが利用可能となる。

FETCH命令でカーソルからデータを１行分読み込む

カーソルが不要になればCLOSE命令でカーソルを終了させる。

・カーソルの動作イメージ

※カーソルを利用することで、１行ごとに処理を変えることが出来る

この行はXXを行う

この行は○○を行う

ハンズオン　カーソルを使用したプロシージャの作成

ハンズオンでは、上司の従業員番号を勤務店舗の店長に変更するMGR\_CHANGEプロシージャの作成を行います。

１．rootユーザでstudbにログインする

mysql -u root -p studb

２．カーソルを利用するMGR\_CHANGEプロシージャを作成する。

-- 勤務する店舗の店長を上司に変更するプロシージャ

DELIMITER //

CREATE PROCEDURE MGR\_CHANGE()

BEGIN

  /\* 宣言は、変数、カーソル、ハンドラーの順で行う必要がある \*/

  -- 変数の宣言

  DECLARE WK\_STORE CHAR(3);

  DECLARE WK\_EMP\_NO CHAR(5);

  DECLARE WK\_JOB INT;

  DECLARE NEW\_MGR\_NO CHAR(5);

  DECLARE done INT DEFAULT FALSE;

  -- カーソルの宣言

  DECLARE cur CURSOR FOR

    SELECT WORK\_STORE, EMP\_NO, JOB\_NO FROM EMPLOYEE;

  -- ハンドラーの宣言

  -- カーソルの最後に達した時にフラグを変更する

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  -- カーソルオープン

  OPEN cur;

  -- カーソルが存在する間ループする

  WHILE NOT done DO

    -- カーソルから１行分のデータをフェッチする

    FETCH cur INTO WK\_STORE, WK\_EMP\_NO, WK\_JOB;

    -- 対象データが店長以外の時

    IF WK\_JOB <> 1 THEN

      -- 勤務店舗の店長の従業員番号を取得する

      SELECT EMP\_NO INTO NEW\_MGR\_NO

      FROM EMPLOYEE

      WHERE WORK\_STORE = WK\_STORE

      AND JOB\_NO = 1;

      -- 上司の従業員番号を更新する

      UPDATE EMPLOYEE

      SET MGR\_NO = NEW\_MGR\_NO

      WHERE EMP\_NO = WK\_EMP\_NO;

    END IF;

  END WHILE;

  -- カーソルクローズ

  CLOSE cur;

END //

DELIMITER;

DELIMITER //

CREATE PROCEDURE MGR\_CHANGE()

BEGIN

DECLARE WK\_STORE CHAR(3);

DECLARE WK\_EMP\_NO CHAR(5);

DECLARE WK\_JOB INT;

DECLARE NEW\_MGR\_NO CHAR(5);

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT WORK\_STORE,EMP\_NO,JOB\_NO FROM EMPLOYEE;

FETCH cur INTO WK\_STORE,WK\_EMP\_NO,WK\_JOB;

IF WK\_JOB <> 1 THEN

SELECT EMP\_NO INTO NEW\_MGR\_NO

FROM EMPLOYEE

WHERE WORK\_STORE = WK\_STORE

AND JOB\_NO = 1;

UPDATE EMPLOYEE

SET MGR\_NO = NEW\_MGR\_NO

WHERE EMP\_NO = WK\_EMP\_NO;

END IF;

END WHILE;

CLOSE cur;

END //

DELIMITER ;

３．丸山隆平と秋元真夏の勤務店舗を中崎町店に変更する

UPDATE EMPLOYEE SET WORK\_STORE = '001'

WHERE EMP\_NO IN ('00010','00011');

４．従業員情報を表示する

※上司の従業員番号が勤務店舗変更前の上司になっていることを確認する。

SELECT WORK\_STORE, EMP\_NO, ENAME, JOB\_NO, MGR\_NO

FROM EMPLOYEE

ORDER BY WORK\_STORE, EMP\_NO;

５．MGR\_CHAGEプロシージャを実行する

CALL MGR\_CHANGE();

６．再度、従業員情報を表示する

　　※上司の従業員番号が更新されていることを確認する

７．トランザクションの取消しを行う

ROLLBACK;

トリガー

# トリガーとは

多くのデータベースでは、テーブルに対して特定の動作が起きた時に、その動作を検知して処理を実行できます。この機能を**トリガー**と言います。具体的には「INSERT、DELETE、UPDATE」のような実際にデータベースに対して書き換えを行うアクションが対象です。

トリガーを利用するにはあらかじめ、トリガーとなる条件と実行したい処理を登録しておきます。トリガーを実行するタイミングはトリガーに設定した条件時のため、CALL命令のような特定のステートメントはありません。

１．トリガーを設定したテーブルに

対してINSERTなどの

アクションを実行すると…

 ２．トリガー内で記述されている

処理を実行する。

トリガーの動作イメージは引き金を引くのと同じ

・トリガーの基本構文

CREATE TRIGGER トリガー名 BEFORE 対象イベント

ON テーブル名 FOR EACH ROW

BEGIN

処理したいSQL文 ;

END ;

※対象イベントには「INSERT」「UPDATE」「DELETE」のいずれかを指定する

トリガータイミング

|  |  |
| --- | --- |
| 書式 | 説明 |
| BEFORE | テーブルに対しての処理の実行前にトリガーを実施 |
| AFTER | テーブルに対しての処理の実行後にトリガーを実施 |

・エイリアスについて

トリガーは特定のテーブルの操作をきっかけに動作するため、操作対象テーブルに対して

変更前と変更後の情報をエイリアスとして取得することが出来る。

|  |  |  |
| --- | --- | --- |
| エイリアス | 説明 | 備考 |
| OLD.列名 | 指定した列の変更前・削除前データ | INSERT時は使用不可 |
| NEW.列名 | 指定した列の変更後・挿入後データ | DELETE時は使用不可 |

・トリガーの作成権限について

トリガーの作成には関連付けられたテーブルに対する TRIGGER 権限が必要です。

また、バイナリログを有効している場合、SUPER権限も必要です。

ハンズオン　トリガーの作成と実行

ハンズオンでは、商品表に対して不正な価格や商品番号を更新出来ないようにチェックをするトリガーを作成します。

１．商品表に更新処理を実行した時に実行されるTRI\_PRODUCTトリガーを作成します。

DELIMITER //

CREATE TRIGGER TRI\_PRODUCT BEFORE UPDATE

ON PRODUCT FOR EACH ROW

BEGIN

  -- 価格０以下は０に変更

  IF NEW.PRICE < 0 THEN

    SET NEW.PRICE = 0;

  END IF;

  -- 商品番号は変更させない

  IF NEW.PRODUCT\_NO <> OLD.PRODUCT\_NO THEN

    SET NEW.PRODUCT\_NO = OLD.PRODUCT\_NO;

  END IF;

END //

DELIMITER ;

２．商品表のカテゴリーがその他のデータを表示する

SELECT \* FROM PRODUCT

WHERE CATEGORY = 'その他';

３．タバスコの価格を900円に更新する

UPDATE PRODUCT SET PRICE = 900

WHERE PRODUCT\_NO = '3001';

４．保温バックの価格を-100円に更新する

UPDATE PRODUCT SET PRICE = -100

WHERE PRODUCT\_NO = '3002';

５．エコバックの商品番号を3009に更新する

UPDATE PRODUCT SET PRODUCT\_NO = '3009'

WHERE PRODUCT\_NO = '3003';

６．商品表のカテゴリーがその他のデータを表示する

SELECT \* FROM PRODUCT

WHERE CATEGORY = 'その他';

　※トリガーにより４、５の更新処理に変化があることを確認する。

７．トランザクションの取消しを行う

ROLLBACK;

・トリガーの確認方法

トリガーはファンクションやプロシージャのようにROUTINESでは確認することが出来ません。トリガーを確認する時には、SHOW TRIGGERSを使用します。

SHOW TRIGGERS\G FROM データベース名;

・トリガーの削除

トリガーの削除はDROP TRIGGER文を使用して削除を行います。

　DROP TRIGGER トリガー名;