副問い合わせ

SQLには、SQL文の内部に別のSELECT文を記述する、**副問い合わせ**という機能が備わっています。この機能を使うことで１つのSQL文で２つ以上の処理を実行することが出来ます。

# ２回のSELECTが必要な状況

以前学習したグループ関数を使うことで「最も大きい給与金額」などを算出することが出来ます。しかし、このデータは集約されているため他の列データを参照することが出来ません。

その為、「最も大きい給与金額」をもらっているのは誰か？を調べるには以下のように２回SELECT文を発行する必要があります。

１．SELECT MAX(SALARY) FROM EMPLOYEE ;

|  |
| --- |
| MAX(SALARY) |
| 5000 |

２．SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY = **5000** ;

|  |  |  |
| --- | --- | --- |
| EMP\_NO | ENAME | SALARY |
| 00001 | 小林友治 | 5000 |

なぜ、このように２回に分けたかというと、最も大きい給与金額は**実際に従業員表を調べてみないと値がわからない**からです。今回のように「ひとまずSELECT文で何らかの検索結果を求めて、その値を用いてさらにSELECTやUPDATEなどを実行する」ことは、データベースではよくあります。

# SELECTをネストする

先程の例だと、最大値が5000と結果が出たがその値がずっと最大値である保証はありません。例えば、小林友治さんが退職すれば値が変わりますし、他の従業員が昇給などをした場合、最大値が変わる可能性があります。その度に最大値を求める為だけに２回SQLを発行するのは手間がかかります。

そこで、SQL文の内部に別のSELECT文を記述する**副問い合わせ**を使うことで、先程のSQL文は次のように書き換えることが出来ます。

ハンズオン　副問い合わせを利用する

1. 従業員の名前を表示するSQL文の検索条件に最大給与を求めるSQL文を記述する

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY = **( SELECT MAX(SALARY) FROM EMPLOYEE )** ;

このように、SQL文の中に( )を使って別のSELECT文を記述することで、別のSQLが実行されます。この( )部分のSQLを**副問い合わせ**や**サブクエリ**と呼びます。

# 副問い合わせを習得するコツ

副問い合わせを使うことで、複雑で高度なSQL文を書くことが可能になりますが、はじめから長くて複雑なSQLを書くのは難しいです。しかし、ひとつひとつの副問い合わせを部品ととらえれば単純なSQLに過ぎません。つまり、**個々のSQL文を一つずつ作り、あとから組み立てればOK**です。主に副問い合わせを理解するコツは２つあります。

1. 副問い合わせが処理されるしくみを知る

副問合せはSQL文がネスト構造になっています。関数と同じようにまずは内側にあるSELECT文から実行されて、副問合せがそのまま結果に置き換わります。

先程のハンズオンを例に処理の流れを記載します。

メインのSQL文(主問合せ)

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY =

副問合せ

**( SELECT MAX(SALARY) FROM EMPLOYEE )**

メインのSQL文(主問合せ)

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY =

結果

5000

1. 副問い合わせのパターンを知る

副問い合わせで得られる検索結果について、考えてみましょう。副問い合わせの中身はSELECT文なので検索結果の形は以下の３種類が考えられます。

・単一の値が求まるSELECT文

・複数の値が求まるSELECT文

・２次元に並んだ値（表形式）が求まるSELECT文

ハンズオン　SELECTの検索結果を確認する。

1. 従業員表から最大給与を求める。【単一の値が求まるSELECT文】

SELECT MAX(SALARY) FROM EMPLOYEE ;

1. 職種が店長の給与を求める。【複数の値が求まるSELECT文】

SELECT SALARY FROM EMPLOYEE

WHERE JOB = '店長' ;

1. 中崎町店に勤務する従業員の従業員番号と名前と職種と給与を求める。

【２次元に並んだ値（表形式）が求まるSELECT文】

SELECT EMP\_NO, ENAME, JOB, SALARY

FROM EMPLOYEE

WHERE WORK\_STORE = '001' ;

このようにSELECT文の内容により結果のデータ構造が変わります。副問い合わせを扱うにはこの違いを理解しておく必要があります。次節から各パターンを説明していきます。

なお、算数の九九表を参考にすればデータ構造のイメージが付きやすいかと思います。

|  |
| --- |
| 8 |

２×４【値】 ：

(スカラー)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |

２の段【配列】：

(ベクター)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| … | … | … | … | … | … | … | … | … |
| 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |

九九【表】 ：

(マトリックス)

# 単一行副問い合わせ

単一行副問い合わせは、検索結果が１行１列の値となる副問い合わせで、**1つの値に化ける**とも考えられます。単一行副問い合わせは、単一の値を記述できる場所であれば基本的にどこでも記述することが出来ます。代表的な例は、SELECT文の選択列やUPDATEのSET句などがあげられます。

ハンズオン　単一行副問い合わせを実行する

1. 小林圭輔さんの時給と従業員の平均時給を表示する。

SELECT EMP\_NO, ENAME, SALARY,

(SELECT AVG(SALARY) FROM EMPLOYEE) AS '平均時給'

FROM EMPLOYEE

WHERE EMP\_NO = '00002' ;

1. 「チキン＆ベーコン」の商品価格を確認する

SELECT PRICE FROM PRODUCT

WHERE PRODUCT\_NO = '0005';

1. 試作品の「MAXチーズピザ」の価格を商品の「チキン＆ベーコン」と

同じ金額に更新する。

UPDATE PROTOTYPE

SET PRICE = ( **SELECT PRICE FROM PRODUCT**

**WHERE PRODUCT\_NO = '0005'** )

WHERE PROTOTYPE\_NO = '9001' ;

1. 試作品の確認を行い、トランザクションの取消しを行う

SELECT \* FROM PROTOTYPE ;

ROLLBACK ;

# 複数行副問合せ

複数行副問い合わせは、検索結果がN行1列の複数の値となる副問い合わせで、**複数の値に化ける**とも考えることが出来ます。複数行副問い合わせはSQL文の中でリストのような値が列挙するような場所に記述することが出来ます。学習済みの範囲ではIN演算子、それ以外ではANY、ALL演算子を用いた条件式で主に使用されます。

・IN演算子で利用する

IN演算子は通常、値のリストをカンマ区切りでしてします。

WHERE JOB IN ( '店長', '配達', '調理' )

このINの右側にある部分を副問い合わせに置き換えることが出来ます。

WHERE JOB IN ( **複数行副問い合わせ** )

・ANY、ALL演算子で利用する

IN演算子は複数の値のいずれかと等しいかを判定することが出来る演算子です。

もし、複数の値と大小を比較したい場合には、ANY演算子やALL演算子を利用することが出来ます。**ANYやALLは単体では使用できなくて必ず直前に比較演算子を付けて利用**します。それぞれの説明は以下の表に記載しています。

|  |  |
| --- | --- |
| 演算子 | 説明 |
| ANY | 値リストのそれぞれと比較して、**いずれか**がTRUEならTRUE |
| ALL | 値リストのそれぞれと比較して、**すべて**がTRUEならTRUE |

IN演算子と同じく、値リストをカンマ区切りで指定を行いますがMySQLではANY、ALL演算子は副問い合わせでしか利用できませんのでイメージでとらえてください。

WHERE SALARY < ANY (1000, 1500, 2000)

SALARYが1000, 1500, 2000のいずれかの値より小さければTRUEとなる。

WHERE SALARY < ALL (1000, 1500, 2000)

SALARYが1000, 1500, 2000のすべての値より小さければTRUEとなる。

こちらもANY、ALLの右側にある部分を副問い合わせに置き換えることが出来ます。

WHERE SALARY < ANY ( **複数行副問い合わせ** )

WHERE SALARY < ALL ( **複数行副問い合わせ** )

ハンズオン　複数行副問い合わせを利用する

1. 職種ごとの最大給与を求める。

SELECT MAX( SALARY ) FROM EMPLOYEE

GROUP BY JOB ;

1. 職種ごとの最大給与と同じ給与の従業員情報を取得する。

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY IN ( **SELECT MAX(SALARY) FROM EMPLOYEE**

**GROUP BY JOB** ) ;

1. 職種ごとの最大給与の値から１つでも多い給与が設定されている従業員を取得する。

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY >= ANY ( **SELECT MAX(SALARY) FROM EMPLOYEE**

**GROUP BY JOB** ) ;

1. 職種ごとの最大給与のすべての値より多い給与が設定されている従業員を取得する。

SELECT EMP\_NO, ENAME, SALARY

FROM EMPLOYEE

WHERE SALARY >= ALL ( **SELECT MAX(SALARY) FROM EMPLOYEE**

**GROUP BY JOB** ) ;

# 表の結果となる副問合せ

表副問い合わせは、副問い合わせの結果がN行Ｍ列の複数行、複数列の値となる副問い合わせで、**表形式に化ける**とも考えられます。このパターンの副問い合わせは通常のSQL文において表を記述することのできる箇所、例えばFROM句やINSERT文などに記述することが出来ます。

・FROM句で利用する

複数行、複数列の値となる副問い合わせは、表形式なのでFROM句に記述することが出来ます。この時、副問い合わせの結果を識別できるように別名を付けます。

SELECT 別名.列名 FROM ( **副問い合わせ** ) AS 別名

・INSERT文で利用する

INSERT文は、基本的に１回の命令で１行しか追加できません。そのため100件データを登録する場合100件分のINSERT文を実行する必要があります。しかし副問い合わせを使えば１回のINSERT文で複数行のデータを登録することができます。

この副問い合わせはINSERT文のVALUESの部分が副問い合わせの結果になります。

INSERT INTO 表名 ( 列名１, 列名２ )

SELECT 列名１, 列名２ FROM 表名

※厳密に言うと、この例は副問い合わせでは無く、INSERTの特殊構文になります。

ハンズオン　表副問い合わせを利用する

1. 商品表と試作品表の中から最大価格の商品金額を表示させる。

SELECT MAX( P.PRICE )

FROM (SELECT \* FROM PRODUCT

UNION

SELECT \* FROM PROTOTYPE ) AS P ;

1. 試作品表の情報を商品表に追加する。

INSERT INTO PRODUCT( PRODUCT\_NO, PNAME, CATEGORY, PRICE )

SELECT PROTOTYPE\_NO, PROTONAME, CATEGORY, PRICE

FROM PROTOTYPE ;

1. 商品表に追加されているか確認して、トランザクションの取消しを行う。

SELECT \* FROM PRODUCT ;

ROLLBACK ;

# 相関副問合せ

通常、副問い合わせは外側のSQL（主問合せ）から独立しています。このため、従業員表から自身が勤務している店舗の平均時給より多く給与をもらっているデータを取得したい場合、**従業員ごとに平均時給の値が変わるため**、通常の副問い合わせでは対応できません。

このような場合、副問い合わせの内部に主問い合せの表や列を利用する**相関副問い合わせ**を使うことで結果を求めることが出来ます。

ハンズオン　相関副問い合わせを利用する

1. 勤務店舗ごとの平均時給を取得する。

SELECT WORK\_STORE, AVG(SALARY)

FROM EMPLOYEE

GROUP BY WORK\_STORE

ORDER BY WORK\_STORE ;

1. 勤務店舗ごとの平均時給より多く給与を得ている従業員を取得する。

SELECT E.ENAME, E.WORK\_STORE, E.SALARY

FROM EMPLOYEE AS E

WHERE E.SALARY >= ( SELECT AVG(SUB.SALARY)

FROM EMPLOYEE AS SUB

WHERE SUB.WORK\_STORE = **E.WORK\_STORE** )

ORDER BY E.WORK\_STORE ;

・相関副問い合わせ動作イメージ

|  |  |  |  |
| --- | --- | --- | --- |
| EMP\_NO | ENAME | SALARY | WORK\_STORE  →この値で副問い合わせを実行 |
| 00001  00002  00003 | 小林友治  小林圭輔  友保隼平 | 5000  2875  950 | 003  →この値で副問い合わせを実行  001  →この値で副問い合わせを実行  001 |

通常の副問い合わせは外側のSQLが実施される前に１回実施されるだけですが、

相関副問い合わせは、外側のSQLの行ごとに副問い合わせを実行されるためDBMSの負荷は大きくなります。

・EXISTS演算子

EXISTS演算子は、副問い合わせの結果が存在しているかチェックする演算子になります。

相関副問い合わせと合わせてよく使われます。なお、EXISTS演算子を使った副問い合わせは結果が存在しているかどうかなので、列指定はあまり意味を成しません。なので＊などで無駄に列増やさずに'X'などの固定値を記述して列を使用しないことを明示化します。

ハンズオン　EXISTS演算子を利用する

1. 従業員表から上司の従業員番号の一覧を取得する。

SELECT DISTINCT MGR\_NO FROM EMPLOYEE ;

1. 上司の従業員番号に指定されている従業員を取得する。

SELECT \* FROM EMPLOYEE AS E

WHERE EXISTS ( SELECT 'X' FROM EMPLOYEE AS SUB

WHERE SUB.MGR\_NO = **E.EMP\_NO** ) ;

・EXISTS演算子動作イメージ

|  |  |  |
| --- | --- | --- |
| ENAME | SALARY | EMP\_NO  →この値の副問い合わせ結果が１件でもあれば表示 |
| 小林友治  小林圭輔  友保隼平 | 5000  2875  950 | 00001  →この値の副問い合わせ結果が１件でもあれば表示  00002  →この値の副問い合わせ結果が１件でもあれば表示  00003 |

※副問い合わせの結果が存在しないかチェックする場合はNOT EXISTSを使用します。