第５章　複雑な問い合わせ

５－１　ビュー

ビューとテーブル・・・２つの違いはあまりなく、テーブルの中には実際のデータが保存されビューの中にはセレクト文が入っていて仮想のテーブルのようなものがあるイメージ

ビューのメリット・・・メリットはデータを保存しないため、容量を節約できるところ。

セレクト文をいちいち書かなくてもビューとして保存すれば使いまわしがきくので時間の節約につながること。※よく使うセレクト文はビューにするとよい。

使い方

　　CREATE VIEW　ビュー名　（＜ビューの列名１＞、＜ビューの列名２＞）

　　AS

　　SELECT文

このように上にこの分を作るだけでビュー文を作ることができる。

例　　　　　　　　　　　　　　　　　　↓ここは日本語でも大丈夫（ビューの場合のみ）

CREATE VIEW shohinSUM(shohin\_bunrui,cnt\_shohin)　　　これで商品分類のカウントを見る

AS コマンドのビュー文が作られました。

SELECT shohin\_bunrui,COUNT(\*)

FROM shohin

GROUP BY shohin\_bunrui;

次はこれの内容を見てみよう。

見るときは

SELECT　＜ビューで決めた列名１＞、＜ビューで決めた列名２＞

|  |  |
| --- | --- |
| 衣服 | 2 |
| 事務用品 | 2 |
| キッチン用品 | 4 |

FROM＜ビュー名＞

例

SELECT shohin\_bunrui,cnt\_shohin

FROM shohinsum;

これで確認ができますが、これはビュー文なので実際にテーブルに記載などはなく、容量が

節約されます。

　　　　ビューに対する検索

またビュー同士でも重ねることができる。（多段ビュー）

例

|  |  |
| --- | --- |
| 事務用品 | 2 |

CREATE VIEW shohinSumJim(shohin\_bunrui,cnt\_shohin)

AS

SELECT shohin\_bunrui,cnt\_shohin

FROM ShohinSUM

WHERE shohin\_bunrui = '事務用品';

SELECT shohin\_bunrui,cnt\_shohin

FROM shohinSumJim;

これで先ほど作ったものをshohinSumJimをつくりその中で定義することで検索され、見るこ

とができる、しかしパフォーマンスの低下を招くのでなるべくしないようにする

ビューでの制限事項

1. ビュー定義でORDER　BYをしない・・・ビューも順序がないので基本的にはできない
2. ビューに対する更新・・・ビューとテーブルの更新は連動して行われるので集約したビューは

　更新不可能。

もし更新するなら、

CREATE VIEW shohiJim(shohin\_id,shohin\_mei,shohin\_bunrui,hanbai\_tanka,shiire\_tanka,torokubi)

AS

SELECT \* 【集約も結合もされてないSELECT文なので更新する

FROM shohin 　ことができる。しかしビューだけではなく、テーブル

WHERE shohin\_bunrui = '事務用品';　　　　　本体にも追加されるので注意する。】

INSERT INTO shohiJim VALUES ('0009','印鑑','事務用品','95','10','2009-11-30');

　　　　ビューの削除

DROP VIEW　ビュー名・・・これで作成したビューを削除することができる。

　５－２　サブクエリ

サブクエリとビュー

サブクエリ・・・使い捨てのビューのようなもの、ビュー定義でのSELECT文のこと。

　　　　実行後に削除される。

ビューで作ったものの確認をするときは、

SELECT　＜ビューで決めた列名１＞、＜ビューで決めた列名２＞

FROM＜ビュー名＞

で見ることができたが、これをサブクエリを使って表現すると

SELECT <列名１>、＜列名２＞

FROM (SELECT　＜列名１＞、・・・ この()の中のものがサブクエリとなる。

FROM ＜テーブル名＞

GROUP BY　＜列名1＞)　AS　ビュー名

と表現する。

　　　　サブクエリの断層数を増やす

サブクエリの中のFROMにさらにFROMというように複数のサブクエリを設定することができる。

例

SELECT shohin\_bunrui,cnt\_shohin

FROM(SELECT　\*

　　　　 FROM(SELECT shohin\_bunrui,COUNT(\*) AS cnt\_shohin

　　FROM shohin

　　GROUP BY shohin\_bunrui) AS shohinSum

WHERE cnt\_shohin = 4)AS shohinSum2;

このように無限につけることができるが、見にくいので基本的には使用しない。

実際これ何かいてるかよくわかんない、なんやねんこれ・・・

　　　スカラ・サブクエリ・・・スカラとは、「単一の」という意味、サブクエリは基本的には複数行を

　結果として返しますが、スカラ・サブクエリでは必ず１行１列だけの戻り値

　戻り値を＝、＜＞などの入力として使うことができる

例

SELECT shohin\_id,shohin\_mei,hanbai\_tanka これだと販売単価が平均値より高いものを

FROM shohin 表示しようとしているが、WHEREにAVGのよ

WHERE hanbai\_tanka > AVG(hanbai\_tanka)　　　うな集約関数をつかえないのでエラーになる。

そこでスカラ・サブクエリを使うことで表示することができる。

SELECT shohin\_id,shohin\_mei,hanbai\_tanka

　　 FROM shohin

　　 WHERE hanbai\_tanka > (SELECT AVG(hanbai\_tanka)←ここで先に計算させてるので

FROM shohin);　AVGの結果の数字として扱われる。

※スカラ・サブクエリはSELECT句でもWHERE句でもどこでも使うことができます。

例

セレクト句で使う場合

SELECT shohin\_id,shohin\_mei,hanbai\_tanka,(SELECT AVG(hanbai\_tanka)

　　　　 　FROM shohin)AS avg\_tanka

|  |  |  |  |
| --- | --- | --- | --- |
| 0001 | Tシャツ | 2000 | 2097.5000 |
| 0002 | 穴あけパンチ | 500 | 2097.5000 |
| 0003 | カッターシャツ | 3000 | 2097.5000 |

FROM shohin;

列に販売単価の平均値を表示するavg\_tanka

が追加された。

HAVING句で使う場合

|  |  |
| --- | --- |
| 衣服 | 2500.0000 |
| キッチン用品 | 2795.0000 |

SELECT shohin\_bunrui,AVG(hanbai\_tanka)

FROM shohin

GROUP BY shohin\_bunrui

HAVING AVG(hanbai\_tanka) > (SELECT AVG(hanbai\_tanka)

　　　　　FROM shohin);

　　　　スカラ・サブクエリを使う注意点

・絶対にサブクエリが複数行を返さないようにする。

例

SELECT shohin\_id,shohin\_mei,hanbai\_tanka,(SELECT AVG(hanbai\_tanka)

FROM shohin

スカラ・サブクエリなのに複数行ある→GROUP BY shohin\_bunrui) AS avg\_tanka

FROM shohin;

これはスカラ・サブクエリが複数行ありますよ　というエラーになる。

５－４　相関サブクエリ

相関サブクエリ・・・小分けしたグループ内での比較をするときに使う。

　　相関サブクエリの結合条件はサブクエリの中に書かないとエラーになる。

サブクエリと相関サブクエリの違い

・・・サブクエリは販売単価が全体の平均より高い商品の表示などができる。

　　　相関サブクエリは商品分類ごとの平均販売単価より高い商品の表示など

　　　より細かい指定ができるサブクエリのこと。

まず普通の商品分類ごとの平均値の表示 　　　　　↓各商品分類ごとの平均値の表示

|  |  |
| --- | --- |
| 衣服 | 2500.0000 |
| 事務用品 | 300.0000 |
| キッチン用品 | 2795.0000 |

SELECT shohin\_bunrui,AVG(hanbai\_tanka)

FROM shohin

GROUP BY shohin\_bunrui;

この分類ごとの単価が平均値を超えたものの表示をしたいが、

SELECT shohin\_id,shohin\_mei,hanbai\_tanka

FROM shohin

WHERE hanbai\_tanka >(SELECT AVG(hanbai\_tanka)

FROM shohin

GROUP BY shohin\_bunrui);

|  |  |  |
| --- | --- | --- |
| 0002 | 穴あけパンチ | 500 |
| 0003 | カッターシャツ | 3000 |
| 0004 | 包丁 | 3000 |
| 0005 | 圧力鍋 | 6800 |

これだとスカラ・サブクエリが複数行あるのでエラーになります。

そこで出るのが今回の相関サブクエリです。

SELECT shohin\_id,shohin\_mei,hanbai\_tanka

FROM shohin AS S1←ここをS1と名付け

WHERE hanbai\_tanka >(SELECT AVG(hanbai\_tanka)

FROM shohin AS S2←ここをS2に名付け

　　 ※ここが大事！！→WHERE S1.shohin\_bunrui =S2.shohin\_bunrui←同じものという条件

GROUP BY shohin\_bunrui);

これで分類ごとの平均値以上の値段の商品が表示される。

表で見るとこのようになる

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | 商品名 | 商品分類 | 販売単価 | 仕入単価 | 登録日 |
| ~~0001~~ | ~~Tシャツ~~ | ~~衣服~~ | ~~2000~~ | ~~500~~ | ~~2009-09-20~~ |
| 0003 | カッターシャツ | 衣服 | 3000 | 2800 |  |
| 0002 | 穴あけパンチ | 事務用品 | 500 | 320 | 2009-09-11 |
| ~~0008~~ | ~~ボールペン~~ | ~~事務用品~~ | ~~100~~ |  | ~~2009-11-11~~ |
| 0004 | 包丁 | キッチン用品 | 3000 | 2800 | 2009-09-20 |
| 0005 | 圧力鍋 | キッチン用品 | 6800 | 5000 | 2009-01-15 |
| ~~0006~~ | ~~フォーク~~ | ~~キッチン用品~~ | ~~500~~ |  | ~~2009-09-20~~ |
| ~~0007~~ | ~~おろしがね~~ | ~~キッチン用品~~ | ~~880~~ | ~~790~~ | ~~2008-04-28~~ |

|  |  |
| --- | --- |
| 商品分類 | 平均値 |
| 衣服 | 2500.0000 |
| 事務用品 | 300.0000 |
| キッチン用品 | 2795.0000 |

相関サブクエリの例

order\_basic1 employee1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| oid | odate | cid | eid | memo |
| 20080512-053139 | 2008-05-12 | C00001 | E1 |  |
| 20080512-097948 | 2008-05-12 | C00002 | E2 | 至急 |
| 20080513-023873 | 2008-05-13 | C00001 | E1 |  |
| 20080514-057971 | 2008-05-14 | C00099 |  |  |

|  |  |
| --- | --- |
| eid | ename |
| E1 | 山田 |
| E2 | 小林 |
| E3 | 相沢 |
| E4 | 井上 |

|  |  |
| --- | --- |
| eid | ename |
| E1 | 山田 |
| E2 | 小林 |

(1) order\_basic1, employee1テーブルを使って、注文実績（eid）が１件以上あるものを次のイメージで表現してください。

SELECT eid,ename←今回表示する列

FROM employee1 AS e　←片方のテーブル表示と呼び方をeに指定

WHERE 0 < (SELECT COUNT(\*)←相関サブクエリ内のものが１以上のものを表示

　　　FROM order\_basic1 AS o←もう片方のテーブル表示と呼び方をoと指定

WHERE e.eid = o.eid);←ここで両方のテーブルのeidのものが同じものだという条件定義

（2）order\_basic1, employee1テーブルを使って、２００８年５月１３日以降の注文実績が１つ以上のものを表示してください。

|  |  |
| --- | --- |
| eid | ename |
| E1 | 山田 |

SELECT eid,ename

FROM employee1 AS e

WHERE 0 < (SELECT COUNT(\*)

FROM order\_basic1 AS o

WHERE e.eid = o.eid

AND o.odate>='2008-05-13');←さっきのものにさらにodateが'2008-05-13以降のものという条件も追加