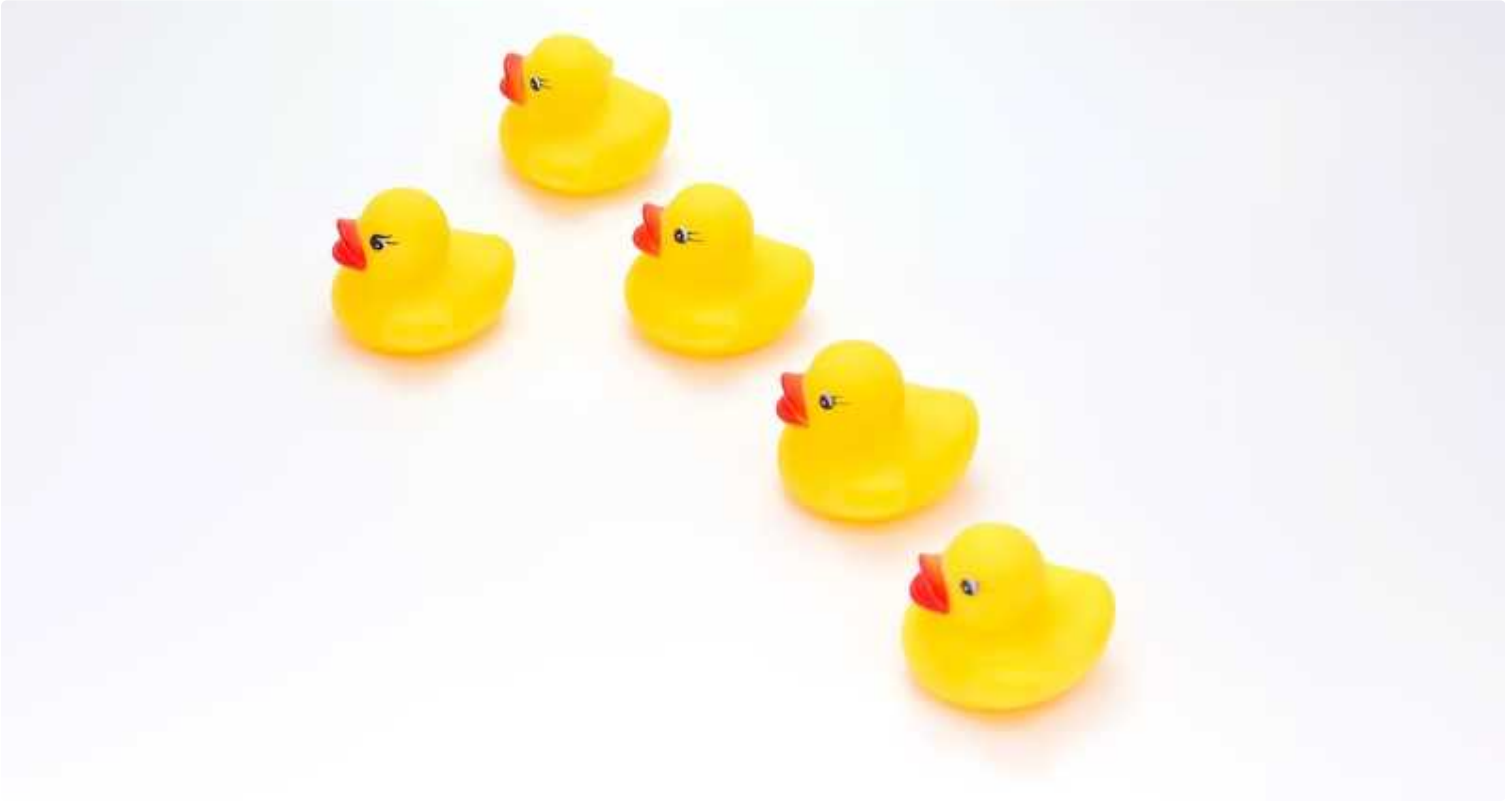
【SQL】グループごとに最大の値を持つレコードを取得する方法3選

2021-04-15



SQL

こんにちはー。

SQLでデータ取得するときに「条件ごとにグループ分けして、それぞれで最大の値を持つレコードを取得したいなー」ということがたまにありますよね。

今回はそういうことを実現する方法を3つご紹介します。

まとめ

`not exists` を使うのが最速。 **`row_number()`** で順番つけるのが意図がわかりやすい。
`group by` 使うのはいまいち。

方法1: group by して max() した結果をjoin

例として、何かしらのスコアを記録しておくテーブルがあって、ユーザーごとに一番新しいレコードを取得したいとしましょう。

「グループごと」、「最大」というキーワードからまず思いつくのは、**group by** して **max** 関数で取得することでしょうか。

以下のような、**user_scores** テーブルがあるとします。

id	user_id	score	created_at
1	1	100	2021-03-01 12:00:00
2	1	150	2021-03-02 12:00:00
3	2	130	2021-03-01 12:00:00
4	2	140	2021-03-01 14:00:00
5	3	210	2021-03-01 11:00:00
6	5	120	2021-03-01 13:00:00
...	...	...	...

ユーザーごとに取りたいので、**user_id** で **group by** します。一番新しいもの、なので、**created_at** を **max()** すればいいですかね。

どれが誰のデータかわからなくなるので **user_id** もselectしましょう。

SQLにするとこんな感じになりますね。

```
select
  user_id,
  max(created_at) max_created_at
from
  user_scores
group by
  user_id
;
```

実行するとこんな感じのデータが取れるはずです。

user_id	max_created_at
1	2021-03-02 12:00:00

user_id	max_created_at
2	2021-03-01 14:00:00
3	2021-03-01 11:00:00
5	2021-03-01 13:00:00
...	...

普通に考えて、スコアとか、他の値も取りたいですよね。

group by しちゃうと取れないので、元のテーブルに **join** して取得しましょう。

```
select
  user_scores.*
from
  user_scores
  inner join
  (
    select
      user_id,
      max(created_at) created_at
    from
      user_scores
    group by
      user_id
  ) t1
    on
      user_scores.user_id = t1.user_id
      and user_scores.created_at = t1.created_at
;
```

こうなりますね。

usersテーブルに **join** して **score** を取りたいよ〜という場合は、こうなります。

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      user_scores.*
    from
      user_scores
      inner join
      (
        select
          user_id,
```

```
            max(created_at) created_at
          from
            user_scores
          group by
            user_id
        ) t1
          on
            user_scores.user_id = t1.user_id
            and user_scores.created_at = t1.created_at
    ) latest_scores
      on
        users.id = latest_scores.user_id
  ;
```

はい。自分で作っておいてなんですが、正直この方法は微妙ですね。

まず、 `join` の条件にtimestampを使うのがなんかスッキリしないですね。
最大の値そのものを `join` の条件に使って絞り込む、というのが強引な感じがします。

あとは、 `group by` はたいてい遅くなりがちですので、データの多い場合はやめた方がいいです。

別に読みやすいっていうこともないので、良いことないですね。

という訳で、この方法は真似しないように。

方法2: row_number() で順番をつけて1番目のやつを取る

続いて、窓関数の `row_number()` を使う方法です。

窓関数っていうのは、レコード全体をグループ（このグループを窓と呼ぶ）に区切って、その区切りごとにあれこれするものです。

`row_number()` を使うと、窓ごとに並び替えて何番目か、の値を取得することができます。

古いバージョンのデータベースだと使えない場合もあります。今どきのやつだったら普通使えるはずです。

先程と同じ `user_scores` テーブルがあるとします。

id	user_id	score	created_at
1	1	100	2021-03-01 12:00:00
2	1	150	2021-03-02 12:00:00

id	user_id	score	created_at
3	2	130	2021-03-01 12:00:00
4	2	140	2021-03-01 14:00:00
5	3	210	2021-03-01 11:00:00
6	5	120	2021-03-01 13:00:00
...	...	...	...

`row_number()` はこんな感じで使います。

```
select
  *,
  row_number() over (
    partition by
      user_id
    order by
      created_at desc
  ) rownum
from
  user_scores
;
```

こんな感じのデータが取れるはずです。

id	user_id	score	created_at	rownum
1	1	100	2021-03-01 12:00:00	2
2	1	150	2021-03-02 12:00:00	1
3	2	130	2021-03-01 12:00:00	2
4	2	140	2021-03-01 14:00:00	1
5	3	210	2021-03-01 11:00:00	1
6	5	120	2021-03-01 13:00:00	1
...	...	...	...	...

説明すると、 `over()` の中に `row_number()` の条件を記述しますが、
`partition by user_id` で、user_idごとにグルーピング、
`order by created_at desc` で、 `created_at` の降順に番号を振ってく、ということになります。

1番新しいレコードに `1` 、2番目に新しいレコードに `2` 、、、といった調子で値が入ります。

私達が欲しいのは、ユーザーごとに最新のレコードですので、 rownum が 1 であるレコードを取得すれば良いということになります。

where で rownum が 1 のレコードだけに絞り込めば良いという話なんですが、 row_number は select 句の中でしか使えないんですよね。
なので、サブクエリを from に入れて絞り込みます。

```
select
  *
from
  (
    select
      *,
      row_number() over (
        partition by
          user_id
        order by
          created_at desc
      ) rownum
    from
      user_scores
  ) with_rownum
where
  rownum = 1
;
```

こんな感じで、ユーザーごとの最新(rownum が 1)のレコードのデータが取れるはずです。

id	user_id	score	created_at	rownum
2	1	150	2021-03-02 12:00:00	1
4	2	140	2021-03-01 14:00:00	1
5	3	210	2021-03-01 11:00:00	1
6	5	120	2021-03-01 13:00:00	1
...	...	...	...	...

users に join するなら、こうなりますね。

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      *
```

```
    from
      (
        select
          *,
          row_number() over (
            partition by
              user_id
            order by
              created_at desc
          ) rownum
        from
          user_scores
      ) with_rownum
    where
      rownum = 1
  ) latest_scores
    on
      users.id = latest_scores.user_id
;
```

この方法は割といいと思います。1番目のやつを取ってるんだなーと言うのが明確です。
速度も **group by** でやるよりは速いことが多いです。

あと、あんまり必要になることがないですが、「ユーザーごとに最新の3件までのレコードを取得」ということも、これだとできます。

```
select
  *
from
  (
    select
      *,
      row_number() over (
        partition by
          user_id
        order by
          created_at desc
      ) rownum
    from
      user_scores
  ) with_rownum
where
  rownum <= 3 -- 最新の3件まで取得
;
```

という訳で、使ってみても良いんじゃないでしょうか。

方法3: not existsで同じテーブルを比較して絞る

最後に **not exists** を使った方法です。これが一番おすすめです。

先程と同じ **user_scores** テーブルがあるとします。

id	user_id	score	created_at
1	1	100	2021-03-01 12:00:00
2	1	150	2021-03-02 12:00:00
3	2	130	2021-03-01 12:00:00
4	2	140	2021-03-01 14:00:00
5	3	210	2021-03-01 11:00:00
6	5	120	2021-03-01 13:00:00
...	...	...	...

user_scores テーブルを取得する条件に、 **not exists** を使って、
「そのレコードと同じグループで、そのレコードより値が大きいレコードが存在しない」レコードに絞り込みます。

「どゆこと？」ってなるかと思うんですが、つまりは「グループごとに最大のレコード」だけが取得されることになります。

こんな感じです。

```
select
  *
from
  user_scores
where
  not exists (
    select
      1
    from
      user_scores sub
    where
      user_scores.user_id = sub.user_id
      and user_scores.created_at < sub.created_at
  ) -- user_idが同じで、created_at がより大きいレコード が存在しない
;
```

exists() の中を説明すると、

まずここは、

```
    from
      user_scores sub
```

大元のクエリで `select` するのと同じ、 `user_scores` を `from` に指定しています。

大元のクエリの `from` と同じ名前になってしまい、条件が書けなくなってしまうので、適当に別名(ここでは `sub`)をつけています。

で、 `where` の方ですが、

```
    where
      user_scores.user_id = sub.user_id
      and user_scores.created_at < sub.created_at
```

1つ目の条件で、大元の `from` にしているレコードと、 `exists` 内で `from` にしているレコードで、同じ `user_id` のものを探します。
さらに2つ目の条件で、大元の `from` にしているレコードより、 `created_at` が大きいレコードを探します。

`not exists` ですので、他に `created_at` が大きいレコードがあったら除外です。
そうすると、 `created_at` の大きさ比べで最後まで勝ち抜いたレコード達だけが選抜される訳です。これでやりたいことが実現できましたね。

`users` と `join` するならこうです。

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      *
    from
      user_scores
    where
      not exists (
        select
          1
        from
          user_scores sub
        where
          user_scores.user_id = sub.user_id
          and user_scores.created_at < sub.created_at
      )
  ) latest_scores
    on
```

```
      users.id = latest_scores.user_id
;
```

私の経験上ですが、この方法でやるのが一番処理が速いです。
複雑なクエリで `row_number()` を使っている箇所を `not exists` を使った方法に変えたら、7秒かかっていたのが0.1秒になったことがあります。圧倒的に速い。

欠点としては慣れてないとぱっと見意味が分からないかも、ってことですかね。

ともかく、速さは正義みたいなところあるのでどんどん使ったら良いと思います。

おさらい

ユーザーごとに最新のスコアを取得したいよーというとき

方法1: group by して max() した結果をjoin

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      user_scores.*
    from
      user_scores
      inner join
      (
        select
          user_id,
          max(created_at) created_at
        from
          user_scores
        group by
          user_id
      ) t1
        on
          user_scores.user_id = t1.user_id
          and user_scores.created_at = t1.created_at
  ) latest_scores
    on
      users.id = latest_scores.user_id
;
```

方法2: row_number() で順番をつけて1番目のやつを取る

```
```

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      *
    from
      (
        select
          *,
          row_number() over (
            partition by
              user_id
            order by
              created_at desc
          ) rownum
        from
          user_scores
      ) with_rownum
    where
      rownum = 1
  ) latest_scores
    on
      users.id = latest_scores.user_id
;
```

方法3: not existsで同じテーブルを比較して絞る

```
select
  users.*,
  latest_scores.score
from
  users
  inner join
  (
    select
      *
    from
      user_scores
    where
      not exists (
        select
          1
        from
          user_scores sub
        where
          user_scores.user_id = sub.user_id
          and user_scores.created_at < sub.created_at
      )
  ) latest_scores
    on
      users.id = latest_scores.user_id
;
```

わかりやすさ

`row_number()` 使う > `not exists` 使う > `group by` 使う

速さ

`not exists` 使う >>> `row_number()` 使う > `group by` 使う

以上です。皆様も良きSQLライフを。