order by 排序

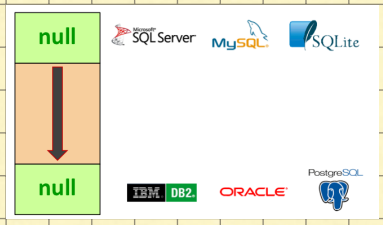
```
order by col1 desc, col2 asc, ...
```

desc是降序，asc是升序。不写默认是升序(asc)

该命令是先将col1降序排，再在相同名内的col2升序排。

排序是依赖于数据类型和DBMS的

ordering depends on the data type



→认为null最小

→认为null最大。

这种排序方式被称为 collations

```
create table ... (
    some_text_column varchar(100)
      collate <collation name> not null,
    ...)

order by nls_sort(some_text_column,
          '<collation name>')
```

PostgreSQL SQL Server MySQL ORACLE

<collation name>是不同数据库中有提供的排序方式

如果要对日期排序，需要注意。

ORACLE PostgreSQL IBM DB2

I've told you that usually dates are converted to a user-friendly format when returned, for instance with TO_CHAR() available in several products.

```
select to_char(a_date_column, 'MM/DD/YYYY')
          as event_date, ...
from ...
where ...
order by event_date
```

No!

But if you sort by this column (text) the sort will be alphabetical! You should sort by the original, date column:

```
order by a_date_column
```

You can sort by a column that isn't returned.

to_char()函数能使日期按一定格式输出为字符串。order by event_date会按字典排序

若想按原来的日期数排序，则要 order by a_date_column.

Director
↓
Producer
↓
Actors

```
order by
  case credited_as
    when 'D' then 1
    when 'P' then 2
    when 'A' then 3
  end
```

The solution is to use CASE ... END to replace each code with a value that sorts as intended. This is frequently used for "custom sorts".

这种方式在少量自定义排序中较为常用。直接指定序号

限制只取前几个

First Page

```
select title,
    country,
    year_released
from movies
order by title
limit 10
```

PostgreSQL MySQL SQLite

```
select title,
    country,
    year_released
from movies
order by title
fetch first 10 rows only
```

IBM DB2 ORACLE PostgreSQL

```
select top 10
    title,
    country,
    year_released
from movies
order by title
```

SQL Server

若要自第几个开始选前几个

```
select title,
    country,
    year_released
from movies
order by title
limit 10      offset 20
```

PostgreSQL MySQL SQLite

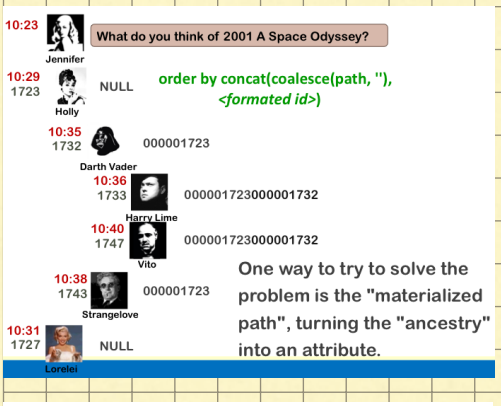
```
select title,
    country,
    year_released
from movies
order by title
offset 20
fetch first 10 rows only
```

IBM DB2 ORACLE PostgreSQL

[x, x+y].

limit y offset x 自x开始取y行。(不包括第x行)。

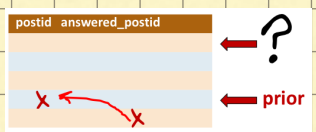
像这样因回复的对象嵌套，形成树状继承结构



可以将前面的id也作为自己的属性

```
ORACLE
select message, ....
from forum_posts ...
connect by answered_postid = prior postid
start with answered_postid is null
     and topicid = ...
order siblings by postid
```

ORACLE提供了一种动态排序的方法
能从某一行指向其逻辑上前面的行



Window Function (窗口函数)

scalar functions (标量函数)：对当行进行操作

aggregate functions (聚合函数)：对一系列行操作.

window functions (窗口函数)：对当行操作但只返回一行为结果.

```
func(parameters) over (magic clause)
```

```
min(year_released)
  over (partition by country)
```

每一个聚合函数都可以被写成窗口函数，只不过不用group by，而用 over (partition by <分组的列名> 一直按这个次序 order by <排序的列名>)

```
select m.country, m.title,
    m.year_released
from
 (select country,
       title,
       year_released,
       min(year_released)
       over (partition by country)
             earliest_year
   from movies) m
where m.year_released = m.earliest_year
 and title <> 'A title here'
```

等价于

```
select a.country, a.title, a.year_released
from movies a
   inner join
   (select country,
         min(year_released) earliest_year
   from movies
   group by country) b
     on b.country = a.country
     and b.earliest_year = a.year_released
where a.title <> 'A title here'
```

若想用聚合函数又不想写group by，可以用窗口函数替代.

```
min(year_released) over()
```

over()中空参数的窗口函数在编译时会翻译成聚合函数.

```
select country_name,
    cnt as number_of_movies,
    round(100 * cnt / sum(cnt) over (), 0)
        as percentage
from (  select c.country_name,
            coalesce(m.cnt, 0) cnt
        from countries c
            left outer join (select country,
                                count(*) cnt
                            from movies
                            group by country) m
                on m.country = c.country_code
order by country_name
```

当有 order by 时，无法在其后计数，所以，应在排序时同时计数，返回后再 order by.

ranking window function (专有窗口函数)：无法与聚合函数相互转化.

row_number()

rank()

dense_rank()

over (order by ...)

over (partition by ... order by ...)

partition by ... → 按…分组

order by ... → 按…排序

专有窗口函数后的 over() 中必须要有 order by...，但 partition by... 可省

over (partition by *col1, col2, ...*
order by *col3, col4, ...*)

可以作用于多列

在排名中，正常排名是 1, 2, 3, 4. 但若前3名并列.

rank() 会返回 1, 1, 1, 4

dense_rank() 会返回 1, 1, 1, 2

row_number() 会返回 1, 2, 3, 4

```
select x.country,
    x.title,
    x.year_released
from
(select country,
    title,
    year_released,
    row_number()
    over (partition by country
        order by year_released desc) rn
 from movies
where x.rn <= 2
```

若对 row_number() 加上条件，效果将会与 limit 作用于 order by 类似.