SQL Задачи и решения

Учебник. Сергей Моисеенко.

Сортировка и NULL-значения стр. 2

В **MySQL** мы можем использовать DISTINCT без обязательного указания в списке SELECT столбцов, по которым выполняется сортировка. Однако здесь нет аналога конструкции WITH TIES, чтобы решить задачу максимально просто.

Поэтому в методе, основанном на сортировке, нам придется использовать подзапрос, чтобы вывести все модели с минимальной ценой:

```
1.  SELECT DISTINCT model FROM PC_
2.  WHERE price = (SELECT price FROM PC_ WHERE price IS NOT NULL ORDER BY price LIMIT 1);
```

Такое же решение будет работать и в **PostgreSQL**, однако он имеет одну особенность, о которой полезно знать. А именно, при сортировке можно указать, где будут выводиться NULL-значения - в начале или в конце результирующего набора. Нам для решения задачи требуется, чтобы NULL выводились в конце отсортированного списка. Тогда не придется выполнять лишнюю операцию по отфильтровыванию NULL-значений:

```
1.  SELECT DISTINCT model FROM PC_
2.  WHERE price = (SELECT price FROM PC_ ORDER BY price nulls last LIMIT 1);
```

Кстати, при сортировке по возрастанию NULL-значения в PostgreSQL идут в конце результирующего рабора. Поэтому конструкция **NULLS LAST**, которую мы использовали выше, можно опустить при решении нашей задачи:

```
1.  SELECT DISTINCT model FROM PC_
2.  WHERE price = (SELECT price FROM PC_ ORDER BY price LIMIT 1);
```

Для того чтобы NULL-значения шли в начале результирующего набора при выполнении сортировки, нужно написать **NULLS FIRST**.

К слову, мы можем смоделировать в MySQL использование конструкций NULLS FIRST/LAST. Для этого воспользуемся тем фактом, что значения логического типа в этой СУБД представляют собой TINYINT(1). Конкретно это означает, что 0 соответствует истинностному значению FALSE (ложь), а ненулевое значение эквивалентно TRUE (истина). При этом логическое выражение, оцениваемое как TRUE будет представлено единицей, т.е.

```
1.  SELECT a IS NULL AS a, b IS NULL AS b FROM (SELECT NULL AS a, 1 AS b) x;
```

даст нам

a	b
1	0

Учитывая то, что 0 при сортировке по возрастанию идет раньше, чем 1, мы можем решение для PostgreSQL адаптировать для MySQL:

```
1.  SELECT DISTINCT model FROM PC_
2.  WHERE price = (SELECT price FROM PC_ ORDER BY price IS NULL, price LIMIT 1);
```

Oracle, как и PostgreSQL, при сортировке по возрастанию помещает NULL-значения в конец результирующего набора. Здесь также имеют место конструкции NULLS FIRST/LAST, но отсутствует аналог LIMIT/TOP N для ограничения количества выводимых строк.

Чтобы смоделировать в Oracle использованный выше подход к решению задачи, можно воспользоваться встроенной функцией **ROWNUM**. Эта функция нумерует строки, но делает это она после выполнения предложений FROM и WHERE, т.е. перед предложениями SELECT и ORDER BY. Такое поведение иллюстрирует результат следующего запроса:

```
1.  SELECT code, model,price, ROWNUM rn FROM PC_ ORDER BY price;
```

CODE	MODEL	PRICE	RN
10	1260	350	10
9	1232	350	9
8	1232	350	8

CODE	MODEL	PRICE	RN
7	1232	400	7
3	1233	600	3
1	1232	600	1
5	1121	850	5
2	1121	850	2
4	1121	850	4
6	1233	950	6
12	1233	970	12
11	1233	980	11
13	2112	NULL	13

Как видно, номер строки не соответствует порядку сортировки. Нетрудно убедиться в том, что нумерация выполнена в соответствии со столбцом code. Это объясняется тем, что оптимизатор использует индекс по этому столбцу при выполнении запроса.

Итак, чтобы найти минимальную цену на основе сортировки, придется использовать подзапрос:

```
1.  SELECT price FROM(
2.  SELECT model,price FROM PC_ ORDER BY price
3.  ) X
4.  WHERE ROWNUM = 1;
```

Теперь, как и в случае MySQL и PostgreSQL, будем использовать этот запрос для получения моделей, которые продаются по цене, найденной с его помощью:

```
1.  SELECT DISTINCT model FROM PC_ WHERE price =
2.  (SELECT price FROM(
3.  SELECT model,price FROM PC_ ORDER BY price
4.  ) X
5.  WHERE ROWNUM = 1
6.  );
```

Как говорил Соломон, от многой мудрости много скорби, и умножающий знание умножает печаль.

Используйте стандартные решения, сказал бы я. :-)

В заключение не могу не сказать о способе, использующем ранжирующие функции.

Идея решения состоит в ранжировании (функция **RANK**) строк по возрастанию цены и выборке (уникальных) строк, для которых ранг равен 1. Чтобы запрос работал под всеми СУБД, которые поддерживают оконные функции, этот алгоритм можно записать следующим образом:

Выполнить Консоль

```
1.  SELECT DISTINCT model FROM (
2.  SELECT model, Rank() OVER (ORDER BY price) rn FROM PC_
3.  WHERE price IS NOT NULL
4.  ) X WHERE rn =1;
```


Oracle PostgreSQL MySQL сортировка NULL WITH TIES NULLS LAST NULLS FIRST ROWNUM TINYINT логический тип данных BOOLEAN

Предыдущая [Комбинация детализированных и агрегированных данных]

[Агрегатная функция от агрегатной функции] Следующая

Последние изменения:

- Упражнение 151 (подсказки и решения)
- Предикат LIKE
- Приложение 2. Список задач стр. 2
- Приложение 2. Список задач
- Футбол

Тэги:
поиск по тэгам

ALL AND AUTO_INCREMENT AVG battles CASE CAST CHAR CHARINDEX CHECK classes COALESCE CONSTRAINT Convert COUNT CROSS APPLY CTE DATEADD DATEDIFF DATENAME DATEPART DATETIME DDL DEFAULT DELETE DISTINCT DML EXCEPT EXISTS EXTRACT FOREIGN KEY FROM FULL JOIN GROUP BY Guadalcanal HAVING IDENTITY IN INFORMATION_SCHEMA INNER JOIN insert INTERSECT IS NOT NULL IS NULL ISNULL laptop LEFT LEFT OUTER JOIN LEN maker

Больше тэгов

Учебник обновлялся
месяц назад
https://exchangesumo.com/obmen/US NIXEUR/ . Определение катета углового сварного шва