n Einträge pro Gruppe

== Problemstellung ==
Es gibt die Tabelle User (Benutzer), die Tabelle Items (Artikel) und sie werden über die Tabelle Sales (Einkäufe) miteinander verknüpft, sobald der Benutzer einen Artikel kauft.
easy-coding.de/Attachment/489/…6596fe87fe77f9f2c8ab8e3f3

== Abfrage aller Einkäufe ==
Quellcode

```
1. SELECT userid,itemid,date FROM `sales` WHERE userid IN (1,3,5);
```

Brainfuck-Quellcode

```
1. +--------+--------+------------+
2. | userid | itemid | date |
3. +--------+--------+------------+
4. | 1 | 915 | 2005-08-17 |
5. | 1 | 2442 | 2004-04-14 |
6. | 1 | 3321 | 2005-09-27 |
7. | 1 | 4326 | 2005-10-29 |
8. | 1 | 11589 | 2005-10-19 |
9. | 1 | 13651 | 2004-06-16 |
10. | 1 | 14756 | 2005-12-27 |
11. | 1 | 14924 | 2005-10-04 |
12. | 1 | 16272 | 2005-01-20 |
13. | 1 | 21722 | 2005-02-07 |
14. | 1 | 30245 | 2004-10-19 |
15. | 1 | 31913 | 2004-10-15 |
16. | 1 | 34907 | 2005-08-17 |
17. | 1 | 38052 | 2004-06-03 |
18. | 1 | 42921 | 2005-10-04 |
19. | 1 | 42930 | 2005-05-14 |
20. | 3 | 1333 | 2004-05-18 |
21. | 3 | 3417 | 2005-09-27 |
22. | 3 | 3718 | 2004-02-09 |
23. | 3 | 4783 | 2003-04-25 |
24. | 3 | 5225 | 2005-07-07 |
25. | 3 | 6384 | 2003-08-26 |
26. | 3 | 6460 | 2003-10-07 |
27. | 3 | 6510 | 2004-09-13 |
28. | 3 | 6689 | 2003-02-20 |
29. | 3 | 10251 | 2004-06-02 |
30. | 3 | 10996 | 2003-11-28 |
31. | 3 | 14035 | 2005-10-12 |
32. | 3 | 15420 | 2004-03-29 |
33. | 3 | 16229 | 2005-11-09 |
34. | 3 | 16272 | 2003-02-13 |
35. | 3 | 16818 | 2004-02-23 |
36. | 3 | 21296 | 2003-09-04 |
37. | 3 | 21777 | 2003-08-28 |
38. | 3 | 21983 | 2003-04-19 |
39. | 3 | 22853 | 2004-02-10 |
40. | 3 | 23174 | 2003-03-07 |
41. | 3 | 24344 | 2004-08-20 |
42. | 3 | 25049 | 2004-05-17 |
```

```
| 3 | 27061 | 2004-10-14 |
| 3 | 28995 | 2005-06-30 |
| 3 | 29948 | 2003-12-21 |
| 3 | 32555 | 2003-12-15 |
| 3 | 32902 | 2004-01-16 |
| 3 | 33849 | 2005-07-11 |
| 3 | 35770 | 2004-12-28 |
| 3 | 37939 | 2005-07-11 |
| 3 | 38129 | 2005-02-07 |
| 3 | 40112 | 2003-03-16 |
| 3 | 41371 | 2004-01-21 |
| 3 | 41947 | 2003-10-04 |
| 5 | 685 | 2005-11-30 |
| 5 | 3321 | 2005-09-09 |
| 5 | 8117 | 2005-11-13 |
| 5 | 11186 | 2005-03-13 |
| 5 | 13432 | 2005-05-08 |
| 5 | 17063 | 2005-03-01 |
| 5 | 19006 | 2005-03-22 |
| 5 | 19289 | 2005-02-17 |
| 5 | 21153 | 2005-10-03 |
| 5 | 23948 | 2005-11-15 |
| 5 | 29350 | 2005-02-24 |
| 5 | 31361 | 2005-08-24 |
| 5 | 33256 | 2004-11-29 |
| 5 | 36818 | 2005-02-01 |
| 5 | 39526 | 2005-01-20 |
+--------+--------+------------+
```

Alles anzeigen

== Abfrage der Benutzer mit 'einem' Einkauf ==
Gruppieren wir die erste Abfrage nach der userid erhalten wir alle Benutzer mit dem Artikel und dem Datum des ersten Einkaufs bzw der natürlichen Reihenfolge, die sich durch das Einfügen in die Tabelle erschließt.
Da diese natürliche Reihenfolge durch den Befehl OPTIMIZE TABLE wieder durcheinander gebracht wird, kann man aber eigentlich nur von einem beliebigem Einkauf reden, der angezeigt wird.

Quellcode

```
SELECT userid,itemid,date FROM `sales` WHERE userid IN (1,3,5) GROUP BY userid;
```

Brainfuck-Quellcode

```
+--------+--------+------------+
| userid | itemid | date |
+--------+--------+------------+
| 1 | 915 | 2005-08-17 |
| 3 | 1333 | 2004-05-18 |
| 5 | 685 | 2005-11-30 |
+--------+--------+------------+
```

== Abfrage der Benutzer mit ihrem letzten Einkauf ==
Wollen wir statt dem ersten Einkauf den letzten erhalten, dann müssen wir die natürliche Reihenfolge, nach der MySQL gruppiert umdrehen, indem wir die Abfrage in einem Subselect absteigend nach der Zeit sortieren.

Quellcode

```
SELECT userid,itemid,date FROM (
SELECT * FROM sales ORDER BY date DESC
) sales WHERE userid IN (1,3,5);
```

Brainfuck-Quellcode

```
+--------+--------+------------+
| userid | itemid | date |
+--------+--------+------------+
| 1 | 14756 | 2005-12-27 |
| 3 | 16229 | 2005-11-09 |
| 5 | 685 | 2005-11-30 |
+--------+--------+------------+
```

== Lösung ==
Um nun überhaupt an zwei Artikel zu gelangen, könnte man die letzten beiden Abfragen per UNION verbinden. Doch wir wollen die beiden letzten.
Dazu bedienen wir uns zweier MySQL Variablen. @x und @userid_tmp.
@x ist eine Zählervariable. Sie nummeriert alle gekauften Items eines Benutzers durch. Sobald @userid_tmp nicht mehr der aktuellen userid ist, dann wird der Zähler zurück gesetzt und die Nummerierung beginnt beim nächsten Benutzer.
Wenn wir die letzten beiden Käufe erhalten wollen, brauchen wir In der WHERE Bedingung nur noch auf die Käufe mit den Zahlen 1 und 2 einzuschränken.

Quellcode

```
SET @x = 0;
SET @userid_tmp = 0;
SELECT userid,itemid,date FROM (
SELECT userid,itemid,date,
@x := IF(@userid_tmp = userid, @x+1, 0) AS c,
@userid_tmp := userid AS d
FROM `sales` WHERE userid IN (1,3,5,6,10,12)
ORDER BY userid ASC,date DESC
) sub
WHERE c < 2;
```

Alles anzeigen

Brainfuck-Quellcode

```
+--------+--------+------------+------+----+
| userid | itemid | date | c | d |
+--------+--------+------------+------+----+
| 1 | 915 | 2005-08-17 | 0 | 1 |
| 1 | 2442 | 2004-04-14 | 1 | 1 |
| 3 | 1333 | 2004-05-18 | 0 | 3 |
| 3 | 3417 | 2005-09-27 | 1 | 3 |
| 5 | 685 | 2005-11-30 | 0 | 5 |
| 5 | 3321 | 2005-09-09 | 1 | 5 |
+--------+--------+------------+------+----+
```