Table of Contents

1. Generating a Numbers table.

There are times when it is helpful to have a table which has a certain number of rows. It seems simple but there are times that this comes in handy. You might also want a table that just contains integers from 0 to whatever. Some people keep a table like that in their database. For a lot of rows (millions of rows) keeping a table might be a good choice, but for a smaller table - maybe a few hundred rows, we might consider creating this table on the fly in the query.

This technique for generating rows may not be terribly efficient for generating a lot of numbers but it is interesting.

1.1. Cross join

We mentioned the cross join briefly early in the semester as something you normally want to avoid. Here we want to use the cross join. With an inner or outer join you define the joining condition between the two tables; with a cross join there is no joining condition and every row in the first table is "matched up" with every row in the second table.

Demo 01:

Suppose we create these two tables and insert a few rows. I am not setting a pk or a fk for these two tables.

```
create table a_testbed.tblbook  ( bookid int, title varchar(25));
create table a_testbed.tbltopic ( bookid int, topic varchar(15));

insert into a_testbed.tblbook values
  (1, 'SQL for Fun and Profit')
, (2, 'Gibson''s History');
insert into tbltopic values
  (1, 'SQL')
, (1, 'Bus')
,(2, 'History');
```

If we do an inner join, we get back three rows which make sense - book 1 has two topics and book 2 has one topic.

```
select bookid, title,topic
from a_testbed.tblbook
join a_testbed.tbltopic using(bookid);
+--------+------------------------+---------+
| bookid | title                  | topic   |
+--------+------------------------+---------+
|      1 | SQL for Fun and Profit | SQL     |
|      1 | SQL for Fun and Profit | Bus     |
|      2 | Gibson's History       | History |
+--------+------------------------+---------+
```

But if we do a cross join, then every books seems to be associated with every topic. Based on our general understanding of those tables, that is just wrong.

```
select tblbook.bookid, title,topic
from a_testbed.tblbook
cross join a_testbed.tbltopic;
+--------+------------------------+---------+
| bookid | title                  | topic   |
+--------+------------------------+---------+
|      1 | SQL for Fun and Profit | SQL     |
|      2 | Gibson's History       | SQL     |
|      1 | SQL for Fun and Profit | Bus     |
|      2 | Gibson's History       | Bus     |
|      1 | SQL for Fun and Profit | History |
|      2 | Gibson's History       | History |
+--------+------------------------+---------+
Drop table a_testbed.tblbook; Drop table a_testbed.tbltopic;
```

But for this discussion, cross joins will be what we want.

1.2. Getting numbers the decimal way

We start with a Union to produce a result set that consists of the rows with the digits 0 to 9.

Demo 02: This is just a bunch of unions of selects with the literal numeric value

```
select 0 as digit
union  select 1
union  select 2
union  select 3
union  select 4
union  select 5
union  select 6 union  select 7 union  select 8 union  select 9;
+-------+
| digit |
+-------+
|     0 |
|     1 |
|     2 |
|     3 |
|     4 |
|     5 |
|     6 |
|     7 |
|     8 |
|     9 |
+-------+
```

Demo 03: If we cross join this table expression with itself, we get the following. I have put the unions on fewer lines to make this a bit shorter. The two subqueries are the same. We get 100 rows (10 *10).

```
Select *
from
( select 0 as digit union  select 1 union  select 2 union  select 3 union
          select 4  union  select 5 union  select 6 union  select 7 union
          select 8  union  select 9 ) Ones
CROSS JOIN
( select 0 as digit union  select 1 union  select 2 union  select 3 union
          select 4  union  select 5 union  select 6 union  select 7 union
          select 8  union  select 9 ) Tens;
```

```
+-------+-------+
| digit | digit |
+-------+-------+
|     0 |     0 |
|     1 |     0 |
|     2 |     0 |
|     3 |     0 |
|     4 |     0 |
|     5 |     0 |
|     6 |     0 |
|     7 |     0 |
|     8 |     0 |
|     9 |     0 |
|     0 |     1 |
|     1 |     1 |
|     2 |     1 |
. . .
|     7 |     9 |
|     8 |     9 |
|     9 |     9 |
+-------+-------+
100 rows in set (0.00 sec)
```

Demo 04: Now we take the digit from the first column and add it to ten times the digit in the second column and the result will be the numbers from 0 to 99.

```
Select Ones.digit + 10 * Tens.digit
from
(select 0 as digit union  select 1 union  select 2 union  select 3 union
select 4
union  select 5 union  select 6 union  select 7 union  select 8 union  select 9
) Ones
CROSS JOIN
(select 0 as digit union  select 1 union  select 2 union  select 3 union
select 4
union  select 5 union  select 6 union  select 7 union  select 8 union  select 9
) Tens;
+------------------------------+
| Ones.digit + 10 * Tens.digit |
+------------------------------+
|                            0 |
|                            1 |
|                            2 |
|                            3 |
|                            4 |
. . . .
|                           98 |
|                           99 |
+------------------------------+
100 rows in set (0.01 sec)
```

To get a thousand rows, put in another cross join and repeat the subquery and for the select use

```
Select Ones.digit + 10 * Tens.digit + 100 * Hundreds.digit
. . .
```

If you need the numbers from 100 to 500, you could add a filter.

```
where Ones.digit + 10 * Tens.digit + 100 * Hundreds.digit between 100 and 500
```

1.3. Getting numbers the binary way

Take a look at a similar technique that uses the base values 1, 2, 4, 8, 16, 32, 64, etc.

Demo 05: These are our starter queries

```
select 0 val union  select 1;
+-----+
| val |
+-----+
|   0 |
|   1 |
+-----+

select 0 val union  select 2;
+-----+
| val |
+-----+
|   0 |
|   2 |
+-----+

select 0 val union  select 4;
+-----+
| val |
+-----+
|   0 |
|   4 |
+-----+
select 0 val union  select 8;
select 0 val union  select 16;
select 0 val union  select 32;
```

Demo 06: Cross join those queries and you get 64 rows. Some of the rows are displayed here

```
select *
FROM
(select 0 val union  all  select 1) b1
Cross join
(select 0 val union  all  select 2) b2
Cross join
(select 0 val union  all  select 4) b4
Cross join
(select 0 val union  all  select 8) b8
Cross join
(select 0 val union  all  select 16) b16
Cross join
(select 0 val union  all  select 32) b32
;
+-----+-----+-----+-----+-----+-----+
| val | val | val | val | val | val |
+-----+-----+-----+-----+-----+-----+
|   0 |   0 |   0 |   0 |   0 |   0 |
|   1 |   0 |   0 |   0 |   0 |   0 |
|   0 |   2 |   0 |   0 |   0 |   0 |
|   1 |   2 |   0 |   0 |   0 |   0 |
|   0 |   0 |   4 |   0 |   0 |   0 |
```

```
|   1 |   0 |   4 |   0 |   0 |   0 |
|   0 |   2 |   4 |   0 |   0 |   0 |
|   1 |   2 |   4 |   0 |   0 |   0 |
|   0 |   0 |   0 |   8 |   0 |   0 |
|   1 |   0 |   0 |   8 |   0 |   0 |
|   0 |   2 |   0 |   8 |   0 |   0 |
|   1 |   2 |   0 |   8 |   0 |   0 |
|   0 |   0 |   4 |   8 |   0 |   0 |
|   1 |   0 |   4 |   8 |   0 |   0 |
|   0 |   2 |   4 |   8 |   0 |   0 |
. . .
|   0 |   0 |   4 |   8 |  16 |  32 |
|   1 |   0 |   4 |   8 |  16 |  32 |
|   0 |   2 |   4 |   8 |  16 |  32 |
|   1 |   2 |   4 |   8 |  16 |  32 |
+-----+-----+-----+-----+-----+-----+
64 rows in set (0.00 sec)
```

Demo 07: Now add the numbers across each row and you get the numbers from 1 to 64

```
select b1.val + b2.val + b4.val + b8.val + b16.val + b32.val
FROM
(select 0 val union  all  select 1)   b1
Cross join
(select 0 val union  all  select 2)   b2
Cross join
(select 0 val union  all  select 4)   b4
Cross join
(select 0 val union  all  select 8)   b8
Cross join
(select 0 val union  all  select 16)  b16
Cross join
(select 0 val union  all  select 32)  b32;
```

What would it take to get numbers from 0 to 999?

Follow the pattern until you get to 1024 (32, 64, 128, 256, 512, 1024)

```
cross join
(select 0 val union  all  select 1024) b1024
```

And then filter the result. Don't generate more rows than you need.

If you do not want the numeric values, you just want a certain number of rows, you can skip the calculation in the select and just use a numeric literal.

Demo 08: If you want to insert those numbers into a table, create the table and use an insert from the query.

```
create table nums (value int);

insert into nums
select Ones.digit + 10 * Tens.digit
from
 (select 0 as digit union  select 1 union  select 2 union  select 3
  union  select 4 union  select 5 union  select 6 union  select 7
  union  select 8 union  select 9 ) Ones
CROSS JOIN
  (select 0 as digit union  select 1 union  select 2 union  select 3
   union  select 4 union  select 5 union  select 6 union  select 7
   union  select 8 union  select 9 ) Tens;
```

2. Using a number table

Now that we have ways of creating a number table, as a virtual table in our query, how could we use this?

Demo 09: We have 18 weeks in a semester. We need to generate at least 18 rows; 16 rows is not enough so we need 32 rows.

```
select NumValue, concat ('unit: ', numvalue)  as Units
from
(
   select b1.val + b2.val + b4.val + b8.val + b16.val as numvalue
   from
     (select 0 val union  all  select 1) b1
      cross join
     (select 0 val union  all  select 2) b2
      cross join
     (select 0 val union  all  select 4) b4
      cross join
     (select 0 val union  all  select 8) b8
      cross join
     (select 0 val union  all  select 16) b16
 ) as gennum
where numvalue between 1 and 18
;
+----------+----------+
| NumValue | Units    |
+----------+----------+
|        1 | Unit: 1  |
|        2 | Unit: 2  |
|        3 | Unit: 3  |
|        4 | Unit: 4  |
|        5 | Unit: 5  |
|        6 | Unit: 6  |
|        7 | Unit: 7  |
|        8 | Unit: 8  |
|        9 | Unit: 9  |
|       10 | Unit: 10 |
|       11 | Unit: 11 |
|       12 | Unit: 12 |
|       13 | Unit: 13 |
|       14 | Unit: 14 |
|       15 | Unit: 15 |
|       16 | Unit: 16 |
|       17 | Unit: 17 |
|       18 | Unit: 18 |
+----------+----------+
18 rows in set (0.00 sec)
```

2.1. Calendars

Demo 10: Creating a calendar for this semester; we need the start date and the end date and we use the adddate function to calculate the dates

```
set @startdtm  := '2012-08-15';
set @stopdtm   := '2012-12-19';
```

```
select semesterdate
from
(  select numvalue, adddate(@startdtm, numvalue)  as semesterdate
   from
  (  select b1.val + b2.val + b4.val + b8.val + b16.val + b32.val
     + b64.val + b128.val  as numvalue
    from
    (  select 0 val union  all  select 1) b1     cross join
    (  select 0 val union  all  select 2) b2     cross join
    (  select 0 val union  all  select 4) b4     cross join
    (  select 0 val union  all  select 8) b8     cross join
    (  select 0 val union  all  select 16) b16   cross join
    (  select 0 val union  all  select 32) b32   cross join
    (  select 0 val union  all  select 64) b64   cross join
    (  select 0 val union  all  select 128) b128
  ) as gennum
) as calendar
where semesterdate between @startdtm  and @stopdtm
;
--- partial output
+--------------+
| semesterdate |
+--------------+
| 2012-08-15   |
| 2012-08-16   |
| 2012-08-17   |
| 2012-08-18   |
| 2012-08-19   |
| 2012-08-20   |
| 2012-08-21   |
| 2012-08-22   |
| 2012-08-23   |
| 2012-08-24   |
| 2012-08-25   |
| 2012-08-26   |

. . .
| 2012-12-15   |
| 2012-12-16   |
| 2012-12-17   |
| 2012-12-18   |
| 2012-12-19   |
+--------------+
127 rows in set (0.00 sec)
```

The extra level of subquery makes it easier to write the Where clause using the column alias established.

Demo 11: What were the weekend days this semester? (You remember weekends!) Just improve the filter.

```
set @startdtm  := '2013-08-17';
set @stopdtm   := '2013-12-25';

select semesterdate
from
(  select numvalue, adddate(@startdtm, numvalue)  as semesterdate
   from
  (  select b1.val + b2.val + b4.val + b8.val + b16.val + b32.val
     + b64.val + b128.val  as numvalue
```

```
    from
    (  select 0 val union  all  select 1) b1     cross join
    (  select 0 val union  all  select 2) b2     cross join
    (  select 0 val union  all  select 4) b4     cross join
    (  select 0 val union  all  select 8) b8     cross join
    (  select 0 val union  all  select 16) b16   cross join
    (  select 0 val union  all  select 32) b32   cross join
    (  select 0 val union  all  select 64) b64   cross join
    (  select 0 val union  all  select 128) b128
  ) as gennum
) as calendar
where semesterdate between @startdtm  and @stopdtm
and dayname(semesterdate) in ('Sunday', 'Saturday')
;
+--------------+
| semesterdate |
+--------------+
| 2013-08-17   |
| 2013-08-18   |
| 2013-08-24   |
| 2013-08-25   |
| 2013-08-31   |
| 2013-09-01   |
| 2013-09-07   |
| 2013-09-08   |
. . .
| 2013-11-16   |
| 2013-11-17   |
| 2013-11-23   |
| 2013-11-24   |
| 2013-11-30   |
| 2013-12-01   |
| 2013-12-07   |
| 2013-12-08   |
| 2013-12-14   |
| 2013-12-15   |
| 2013-12-21   |
| 2013-12-22   |
+--------------+
38 rows in set (0.02 sec)
```

Demo 12: We know how to find the dates in December 2012 where we had any sales in the Altgeld mart tables (a_oe database)

```
select distinct ord_date
from a_oe.order_headers
where year(ord_date) = 2012 and month(ord_date) = 12;
+---------------------+
| ord_date            |
+---------------------+
| 2012-12-05 00:00:00 |
| 2012-12-07 00:00:00 |
| 2012-12-09 00:00:00 |
| 2012-12-15 00:00:00 |
| 2012-12-30 00:00:00 |
+---------------------+
5 rows in set (0.00 sec)
```

Demo 13: What days in December 2012 do we not have any sales? Finding the missing data is a bit more work but is a critical task for a business. Get a Dec 2012 calendar and then do a NOT In subquery.

```
-- calendar for dec 2012
set @startdtm  := '2012-12-01';
set @stopdtm   := '2012-12-31';

select theDate as NoSalesDates
from
(
   select adddate(@startdtm, numvalue)  as theDate
   from
   (  select b1.val + b2.val + b4.val + b8.val + b16.val + b32.val
      as numvalue
      from
      (  select 0 val union  all  select 1)  b1    cross join
      (  select 0 val union  all  select 2)  b2    cross join
      (  select 0 val union  all  select 4)  b4    cross join
      (  select 0 val union  all  select 8)  b8    cross join
      (  select 0 val union  all  select 16) b16   cross join
      (  select 0 val union  all  select 32) b32
   ) as gennum
) calendar
where theDate between @startdtm  and @stopdtm
and theDate not in (
   select ord_date
   from a_oe.order_headers);
+--------------+
| NoSalesDates |
+--------------+
| 2012-12-01   |
| 2012-12-02   |
| 2012-12-03   |
| 2012-12-04   |
| 2012-12-06   |
| 2012-12-08   |
| 2012-12-10   |
| 2012-12-11   |
| 2012-12-12   |
| 2012-12-13   |
| 2012-12-14   |
| 2012-12-16   |
| 2012-12-17   |
| 2012-12-18   |
| 2012-12-19   |
| 2012-12-20   |
| 2012-12-21   |
| 2012-12-22   |
| 2012-12-23   |
| 2012-12-24   |
| 2012-12-25   |
| 2012-12-26   |
| 2012-12-27   |
| 2012-12-28   |
| 2012-12-29   |
| 2012-12-31   |
+--------------+
26 rows in set (0.01 sec)
```