

Creating Efficient SQL – The De-normalize Transpose

Paul D Sherman, San Jose, CA

ABSTRACT

Normalization and warehousing are battle cries of two warring factions. Although vertical or row-wise designs ultimately minimize storage access and retrieval time, the law of the relational model precludes simultaneous access from more than one table row. Quite often, we desire knowledge of two variables for the same aggregate observation, when performing high level arithmetic or reporting spreadsheet like tabature. This article presents an efficient query method for turning rows into columns at SQL run time, which farmer Jack uses for his bean quality audit policy, and discusses some finer points which make the transposition process database system independent.

INTRODUCTION

There's no question flat table structures are easy for humans to perceive. We naturally scan sideways, horizontally, for attributes or variables of the same observation. We scan down the page when comparing observations or looking for the aggregate "bottom line." Dealing with data in a two-dimensional row by column layout therefore is very rewarding.

Database scientists prefer as few columns as possible in any given table. Each column is akin to a member variable in a class, structure or abstract data type. Having fewer members helps keep per-instance storage overhead small. With each new piece of data, one need allocate minimal memory for each structure object.

In a previous article we have addressed the problem of normalizing data, or stacking together information from many sources without using the UNION clause. This work presents the converse operation, namely how to flatten out a normalized or parameterized table of data. While both transpositions are conjugal, their theory and implementation are totally different, owing to the nature of relational algebra.

WHAT'S WRONG WITH FLAT?

Each time a new parameter is desired, one must alter the underlying table structure and add appropriate columns for the new parameter information. To maintain good database integrity, structural change permission is often restricted to a select group of users. Furthermore, as each row need be updated with the new columns, such structure modifications can be exceedingly tedious when table capacity is large. Greatest data flexibility is achieved with vertical, row-wise or normalized designs.

It takes only $n_k + n_p + n_v$ columns to store an unlimited quantity of parameter values in a normalized layout, whereas flat tables need many more columns, $n_k + n_{pv} \cdot n_v$ since each new parameter adds n_v columns. n_k , n_p and n_v are the number of key, parameter and output value columns, respectively. n_{pv} is the number of distinct parameter values.

```
n_k = 3 ( ID, TYPE, SUBTYP )
n_p = 1 ( PARM )
n_{pv} = 4 ( 'Attr1', 'Attr2', 'Attr3', 'Attr4' )
n_v = 5 ( AVG, STD, MIN, MAX, N )
```

Normal form: Column count is independent of n_{pv}

$n_k + n_p + n_v = 3 + 1 + 5 = 9$ columns

ID	TYPE	SUBTYP	PARM	AVG	STD	MIN	MAX	N
.	.	.	Attr1
.	.	.	Attr2
.	.	.	Attr3
.	.	.	Attr4

Flat form: Column count increases with each additional parameter value

$$n_k + n_{pv} \cdot n_v = 3 + 4 \cdot 5 = \mathbf{23 \text{ columns}}$$

ID	TYPE	SUBTYP	A_AVG	A_STD	A_MIN	A_MAX	A_N	B_AVG	B_STD	B_MIN	B_MAX	B_N	...
.
			----- 'Attr1' -----				----- 'Attr2' -----				-- ...		

In general, when there is more than one parameter column and each one, i , has a different quantity of distinct values n_{piv} , the total column count for a flat table structure is

$$n_{flat-columns} = \sum_{i=1}^{n_p} n_{piv} \cdot n_v$$

The world could be flat, too! Every table row is a distinct uncorrelated observation and the subject of subsequent loops in a data, `for()`, `do()` or `while()` step. Ideally, there is no persistence between loop iterations. Any variable for which we desire correlation must be self-contained within the same loop iteration. Remember, each loop incurs a separate fetch operation on the table or its index; within the same loop, member variable access is given for free.

```

/* Normalized Table Access */      /* Flat Table Access */
data _null_;                       data _null_;
  set my_table;                   set my_table;
  put _n_ id parm value;          put _n_
  run;                             id
                                   value_pa
                                   value_pb
                                   value_pc
                                   ...
                                   ;
                                   run;

```

<u>n</u>	<u>id</u>	<u>parm</u>	<u>value</u>	<u>n</u>	<u>id</u>	<u>value_pa</u>	<u>value_pb</u>	<u>value_pc</u>
1	61	a	1.12	1	61	1.12	10	100
2	61	a	10	2	62	1	10	100
3	61	a	100	3	63	1	10	100
4	62	b	1					
5	62	b	10					
6	62	b	100					
7	63	c	1					
8	63	c	10					
9	63	c	100					

One may force inter-loop persistence with the SAS® data step `RETAIN` statement, although doing so is beyond our present discussion.

```

// spins n_obs * n_parms times      // spins only n_obs times
while(table.hasMoreRows()) {        while() {
  row = table.nextRow();             row = table.nextRow();
  row.columnAt("parm");              row.columnAt("value_pa");
  row.columnAt("value");             row.columnAt("value_pb");
}                                     row.columnAt("value_pc");
                                     ...
}

```

Flat table access requires the database engine to spin fewer n_{obs} loops, although picking up more data with each loop.

THE PROCEDURE

Seven simple steps to the De-normalize Transpose method follow. The last is optional, whether one has privilege or desire to create views on the database.

1. Determine the set of key variables comprising smallest row-set aggregate.
2. Identify parameter variables.
3. Enumerate (laboriously) all distinct values of all parameters.
4. Construct table object references for each in step (3).
5. Use outer joins to prevent data loss from incorrect or missing parameter values.
6. Split into small groups, typically ten each, avoiding DBMS limitations.
7. Nail down input requirements in the pivot table object; replicate to all sub-groups.
8. Package into view or stored procedure -
recommend no heavy further filtering in outer-most WHERE clause.

EXAMPLE – SPREADSHEET REPORTING

With increasing popularity, farmer Jack's business enterprise grows to gargantuan proportions. His independent quality measuring system requires two-dimensional spreadsheet data on taste and laboratory results from each bean sampled. Jack needs your help to flatten his normalized measurement data into proper form for his quality audit policy.

STEP 1 – KEY VARIABLES

The `BEAN.BAG` table monitors inventory of all of farmer Jack's beans, and its `BEANID` column provides our unique identifier.

STEP 2 – PARAMETER VARIABLES

Taste and laboratory test data are contained in `BEAN.TASTE` and `BEAN.LAB` tables, respectively, where each table's `PARM` column is the parameter for which we desire de-normalization.

STEP 3 – PARAMETER VARIABLE VALUES

All possible values of the `BEAN.TASTE.PARM` column are for example 'a', 'b' or 'c', therefore we need only supply three instances of `BEAN.TASTE`. Those of the `BEAN.LAB.PARM` column are for example 'd' and 'e'. A useful test query for deriving such exhaustive enumeration is a fullselect `DISTINCT` or `GROUP BY`, the latter giving useful frequency count information and data cleaning criteria as to how many beans of each parameter value were found. One might not wish to create an entire column only for a mis-spelled parameter name occurring for a single bean sample.

STEP 4 – TABLE OBJECTS

Provide one parameter table instance for each distinct parameter variable value.

```

+-----p--+
| BEAN  |
| BAG   |
+-----+
<k>-| BEANID | |-----+-----+-----+...
| GENUS | |         |         |         |
~      ~ | +-----n1--+ | +-----n2--+ | +-----n3--+
+-----+ | | BEAN  | | | BEAN  | | | BEAN  |
          | | TASTE | | | TASTE | | | TASTE |
          | +-----+ | +-----+ | +-----+
          +--o BEANID | +--o BEANID | +--o BEANID |
          ['a']--o PARM | ['b']--o PARM | ['c']--o PARM |
          |         | |         | |         |
          | VALUE |--+ | VALUE |--+ | VALUE |--+
          +-----+ | +-----+ | +-----+ |
                                     +---<value_c>
                                     +-----<value_b>
                                     +-----<value_a>

```

STEP 5 – OUTER JOINS

For any given row of pivot table `p`, all parameter 'input' or normalization columns have been completely specified so that no other table returns more than a single row. The outer join, shown as symbol "o" in the model above, insures non-existence of any will not eliminate the pivot table observations.

STEP 6 – LOGICAL GROUPS

While the de-normalization procedure is a simple exhaustive parallel joining of parameter tables, one instance or copy for each parameter value, in practice database systems often limit the maximum number of table objects in a query FROM clause.

	<u>max. number of FROM clause tables</u>
Database "A"	16
Database "B"	10
Database "C"	25
Database "D"	32

Why is there such limitation? A join of two tables is a doubly-nested loop. With n tables there need be coded $n-1$ nested loops. Accommodating an unlimited number of object joins, or even a large quantity of them, requires tricky coding and memory management on the part of the database system.

```

FROM A
  inner join B on ...
  inner join C on ...
  ...
  inner join Z on ...
      while(Z ...) {
        while(Y ...) {
          while(X ...) {
            ...
            while(B ...) {
              while(A ...) { ... } // pivot
            }
            ...
          }
        }
      }
  
```

Each loop requires its own temporary stack space.

AVOIDING THE DATABASE FROM CLAUSE QUANTITY LIMITATION

It is an easy matter to sub-group de-parameterized table object instances, so that each group fullselect contains a minimal set of table objects, well under any database system limit. Ten (10) is a nice quantity.

Note that it makes no perceivable difference to query output how tables are sub-grouped because regardless of group, all tables refer to the same row observation. In practice, one often groups logically, keeping relevant sets of attributes bound tightly together for query coding convenience.

STEP 7 – INPUT SPECIFICATION

Subset as desired. For best results, make sure these pivot sub-queries are identical.

```

SELECT p.r, p.s,
       a.v1, a.v2,
       b.v1, b.v2,
       ...
FROM ( SELECT k1, k2 FROM pivot WHERE ... ) as p (r, s)
  inner join (
    SELECT x.r, x.s, n1.value, n2.value
    FROM ( SELECT k1, k2 FROM pivot WHERE ... ) as x (r, s)
      left outer join norm as n1 on ... and n1.parm = 'a'
      left outer join norm as n2 on ... and n2.parm = 'b'
  ) as a (rr, ss, v1, v2) on p.k1 = a.rr and p.k2 = a.ss
  inner join (
    SELECT y.r, y.s, n3.value, n4.value
    FROM ( SELECT k1, k2 FROM pivot WHERE ... ) as y (r, s)
      left outer join norm as n3 on ... and n3.parm = 'c'
      left outer join norm as n4 on ... and n4.parm = 'd'
  ) as b (rr, ss, v1, v2) on p.k1 = b.rr and p.k2 = b.ss
  ...
  
```

<u>PIVOT</u>
K1
K2

<u>NORM</u>
K1
K2
PARM
VALUE

Although it may appear this procedure is ultimately limited, i.e., that the outer-most FROM clause must have fewer than n tables (thereby proposing ultimate limit of $n \times n$, or n outer-most by n inner-most, tables), indeed there is no such limit. One simply must arrange further sub-groups as follows:

```

10 tables, max.
SELECT p.k, a.v1, a.v2, ..., a.vn, b.v1, b.v2, ..., b.vn, ...
FROM ( ... ) as p
    inner join ( 10 tables (9 joins), max.
        SELECT
        FROM ( ... ) as ax
            inner join ( 10 tables, max. Inner-most layer always outer joined
                SELECT
                FROM ( ... ) as aax
                    left outer join ...
                    left outer join ...
                ) as aa ( ... ) on ...
            inner join (
                SELECT
                FROM ( ... ) as aby
                    left outer join ...
                    left outer join ...
                ) as ab ( ... ) on ...
            ...
        ) as a ( ... ) on ...
    inner join (
        SELECT
        FROM ( ... ) as bx
            inner join (
                SELECT
                FROM ( ... ) as bax
                    left outer join ...
                    left outer join ...
                ) as ba ( ... ) on ...
            inner join (
                SELECT
                FROM ( ... ) as bby
                    left outer join ...
                    left outer join ...
                ) as bb ( ... ) on ...
            ...
        ) as b ( ... ) on ...
    ...

```

Thus, to any given fullselect there must be no more than n tables. Fullselect statements may, however, have unlimited quantities of subquery fullselects(1).

FARMER JACK'S QUALITY SPREADSHEET

Constituent tables BEAN.TASTE and BEAN.LAB which are transpose de-normalized according to prescription yield the following output. In our example, the same bean cannot both be tasted and laboratory measured. Original data in normal form is given in the Appendix.

<u>BEANID</u>	<u>GENUS</u>	<u>TASTE</u>	<u>FLAVOR</u>	<u>AROMA</u>	<u>SIZE</u>	<u>WEIGHT</u>
843001	LIMA	3.1	2.2	1.1	.	.
734001	JELLY	.	.	.	14.2	5.0
333002	JELLY	3.2	2.3	1.2	.	.

Enumerated BEAN.TASTE parameter values were taken as { 'Taste', 'Flavor', 'Aroma' } and BEAN.LAB parameter values as { 'Size', 'Weight' }. Inflexible hard-coding of information is precisely the penalty one pays for flat, de-normalized table structures.

The full query statement is this:

```
SELECT p.beanid, p.genus, a.taste, a.flavor, a.aroma, b.size, b.weight
FROM bean.bag as p
  inner join (
    SELECT x.beanid, n1.value, n2.value, n3.value
    FROM bean.bag as x
      left outer join bean.lab as n1 on x.beanid = n1.beanid
        and n1.parm = 'Taste'
      left outer join bean.lab as n2 on x.beanid = n2.beanid
        and n2.parm = 'Flavor'
      left outer join bean.lab as n3 on x.beanid = n3.beanid
        and n3.parm = 'Aroma'
  ) as a (beanid, taste, flavor, aroma) on p.beanid = a.beanid
  inner join (
    SELECT y.beanid, n4.value, n5.value
    FROM bean.bag as y
      left outer join bean.lab as n4 on y.beanid = n4.beanid
        and n4.parm = 'Size'
      left outer join bean.lab as n5 on y.beanid = n5.beanid
        and n5.parm = 'Weight'
  ) as b (beanid, size, weight) on p.beanid = b.beanid
```

CONCLUSION

Through comparison of normalized and flat table forms we understand when each is appropriate. A transposition framework is presented, along with special concern for avoiding database system FROM clause quantity limitations. Most often, flattened output will be observed manually or used as higher level aggregation summaries and calculations involving more than one parameter value from the same observation. Together with a previous article, this work completes a conjugal pair of SQL query transposition operations.

REFERENCES

Sherman, Paul. *Creating Efficient SQL - Union Join without the UNION Clause*, in *Proceedings of the Twenty-Ninth Annual SAS User Group International Conference*. Cary, NC: SAS Institute Inc., 2004. Paper 064-29.

1. It is conceivable that some database systems may limit the depth of nested sub-queries.

ACKNOWLEDGMENTS

I owe a great many thanks to Katesuda Chinmeteepitak for supervisory editing and synthesizing an eye-catching title, Xiaoguang Liang for a critical reading of the manuscript and testing its theories, Russ Lavery for discussions relevant to SAS database operation, and to Lynne Babior and Tyler Cole for introducing me to Pharmaceutical terminology.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Paul D Sherman
335 Elan Village Lane, Apt. 424
San Jose, CA 95134
Phone: (408) 383-0471
Email: sherman@idiom.com
<http://www.idiom.com/~sherman/paul>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

APPENDIX – FARMER JACK’S DATA

BEAN.TASTE

<u>PARM</u>	<u>VALUE</u>	<u>BEANID</u>
Taste	3.1	843001
Flavor	2.2	843001
Aroma	1.1	843001
Taste	3.2	333002
Flavor	2.3	333002
Aroma	1.2	333002

BEAN.LAB

<u>PARM</u>	<u>VALUE</u>	<u>BEANID</u>
Size	14.2	734001
Weight	5.0	734001

BEAN.TASTEDEF

<u>NAME</u>	<u>DESCRIP</u>	<u>VALUE</u>
Aroma	Fragrant	1.1
Aroma	Offensive	1.2
Aroma	Rotten	1.3
Aroma	Stink	1.4
Flavor	Sweet	2.1
Flavor	Sour	2.2
Flavor	Bitter	2.3
Flavor	Bland	2.4
Taste	Superb	3.1
Taste	Yuck	3.2

DATABASE DEFINITION STATEMENTS

```
create table bean.bag (  
    beanid integer not null,  
    branch char(8),  
    timest timestamp,  
    name char(8),  
    primary key (beanid)  
);  
  
create table bean.taste (  
    parm char(8) not null,  
    value varchar(32),  
    beanid integer not null,  
    primary key (beanid, parm)  
);  
  
create table bean.lab (  
    parm char(8) not null,  
    value double  
    beanid integer not null,  
    primary key (beanid, parm)  
);
```



```

FROM patients as p
inner join visits as v ON p.id = v.id
inner join (
    SELECT p.id, v.ts,
           cbc_a.a, cbc_a.b, cbc_a.c, cbc_a.d, cbc_a.e,
           cbc_a.f, cbc_a.g, cbc_a.h, cbc_a.i, cbc_a.j,
           cbc_b.a, cbc_b.b, cbc_b.c, cbc_b.d, cbc_b.e
    FROM patients as p
    inner join visits as v ON p.id = v.id
    inner join (
        SELECT p.id, v.ts, a.value, b.value, c.value, d.value, e.value,
               f.value, g.value, h.value, i.value, j.value
        FROM patients as p
        inner join visits as v ON p.id = v.id
        left outer join labdat as a ON v.id=a.id and v.ts=a.ts
            and a.panel='cbc' and a.parm='WBC'
        left outer join labdat as b ON v.id=b.id and v.ts=b.ts
            and b.panel='cbc' and b.parm='RBC'
        left outer join labdat as c ON v.id=c.id and v.ts=c.ts
            and c.panel='cbc' and c.parm='Hemoglobin'
        left outer join labdat as d ON v.id=d.id and v.ts=d.ts
            and d.panel='cbc' and d.parm='Hematocrit'
        left outer join labdat as e ON v.id=e.id and v.ts=e.ts
            and e.panel='cbc' and e.parm='MCV'
        left outer join labdat as f ON v.id=f.id and v.ts=f.ts
            and f.panel='cbc' and f.parm='MCH'
        left outer join labdat as g ON v.id=g.id and v.ts=g.ts
            and g.panel='cbc' and g.parm='MCHC'
        left outer join labdat as h ON v.id=h.id and v.ts=h.ts
            and h.panel='cbc' and h.parm='Platelets'
        left outer join labdat as i ON v.id=i.id and v.ts=i.ts
            and i.panel='cbc' and i.parm='MPV'
        left outer join labdat as j ON v.id=j.id and v.ts=j.ts
            and j.panel='cbc' and j.parm='RDW'
    ) as cbc_a (id, ts, a, b, c, d, e, f, g, h, i, j) ON v.id = cbc_a.id
            and v.ts = cbc_a.ts
    inner join (
        SELECT p.id, v.ts,
               a.value, b.value, c.value, d.value, e.value
        FROM patients as p
        inner join visits as v ON p.id = v.id
        left outer join labdat as a ON v.id = a.id and v.ts = a.ts
            and a.panel = 'cbc' and a.parm = 'AGC'
        left outer join labdat as b ON v.id = b.id and v.ts = b.ts
            and b.panel = 'cbc' and b.parm = 'PMN'
        left outer join labdat as c ON v.id = c.id and v.ts = c.ts
            and c.panel = 'cbc' and c.parm = 'ESR'
        left outer join labdat as d ON v.id = d.id and v.ts = d.ts
            and d.panel = 'cbc' and d.parm = 'INR'
        left outer join labdat as e ON v.id = e.id and v.ts = e.ts
            and e.panel = 'cbc' and e.parm = 'PT'
    ) as cbc_b (id, ts, a, b, c, d, e) ON v.id = cbc_b.id and v.ts = cbc_b.ts
) as cbc (
    id, ts, aa, ab, ac, ad, ae, af, ag, ah, ai, aj, ba, bb, bc, bd, be
) ON v.id = cbc.id and v.ts = cbc.ts

```

```

inner join (
  SELECT p.id, v.ts,
         a.value, b.value, c.value, d.value, e.value, f.value, g.value
  FROM patients as p
        inner join visits as v ON p.id = v.id
        left outer join labdat as a ON v.id = a.id and v.ts = a.ts
                        and a.panel = 'lipid' and a.parm = 'Cholesterol'
        left outer join labdat as b ON v.id = b.id and v.ts = b.ts
                        and b.panel = 'lipid' and b.parm = 'Triglyceride'
        left outer join labdat as c ON v.id = c.id and v.ts = c.ts
                        and c.panel = 'lipid' and c.parm = 'HDL'
        left outer join labdat as d ON v.id = d.id and v.ts = d.ts
                        and d.panel = 'lipid' and d.parm = 'LDL'
        left outer join labdat as e ON v.id = e.id and v.ts = e.ts
                        and e.panel = 'lipid' and e.parm = 'VLDL'
        left outer join labdat as f ON v.id = f.id and v.ts = f.ts
                        and f.panel = 'lipid' and f.parm = 'Chol/HDL Ratio'
        left outer join labdat as g ON v.id = g.id and v.ts = g.ts
                        and g.panel = 'lipid' and g.parm = 'LDL/HDL Ratio'
) as lipid (id, ts, a, b, c, d, e, f, g) ON v.id = lipid.id and v.ts = lipid.ts
inner join (
  SELECT p.id, v.ts,
         a.value, b.value, c.value, d.value, e.value, f.value
  FROM patients as p
        inner join visits as v ON p.id = v.id
        left outer join labdat as a ON v.id = a.id and v.ts = a.ts
                        and a.panel = 'thyroid' and a.parm = 'Free T4'
        left outer join labdat as b ON v.id = b.id and v.ts = b.ts
                        and b.panel = 'thyroid' and b.parm = 'TSH'
        left outer join labdat as c ON v.id = c.id and v.ts = c.ts
                        and c.panel='thyroid' and c.parm='Glycohemoglobin'
        left outer join labdat as d ON v.id = d.id and v.ts = d.ts
                        and d.panel = 'thyroid' and d.parm = 'T3 Total'
        left outer join labdat as e ON v.id = e.id and v.ts = e.ts
                        and e.panel = 'thyroid' and e.parm = 'FTI'
        left outer join labdat as f ON v.id = f.id and v.ts = f.ts
                        and f.panel = 'thyroid' and f.parm = 'T3 Uptake'
) as thyroid (id, ts, a, b, c, d, e, f) ON v.id=thyroid.id and v.ts=thyroid.ts

```

```

inner join (
  SELECT p.id, v.ts,
         mb_a.a, mb_a.b, mb_a.c, mb_a.d, mb_a.e,
         mb_a.f, mb_a.g, mb_a.h, mb_a.i, mb_a.j,
         mb_b.a, mb_b.b, mb_b.c, mb_b.d, mb_b.e, mb_b.f, mb_b.g, mb_b.h
  FROM patients as p
  inner join visits as v ON p.id = v.id
  inner join (
    SELECT p.id, v.ts, a.value, b.value, c.value, d.value, e.value,
           f.value, g.value, h.value, i.value, j.value
    FROM patients as p
    inner join visits as v ON p.id = v.id
    left outer join labdat as a ON v.id = a.id and v.ts = a.ts
      and a.panel='metabolic' and a.parm='Glucose'
    left outer join labdat as b ON v.id = b.id and v.ts = b.ts
      and b.panel = 'metabolic' and b.parm = 'BUN'
    left outer join labdat as c ON v.id = c.id and v.ts = c.ts
      and c.panel='metabolic' and c.parm='Creatinine'
    left outer join labdat as d ON v.id = d.id and v.ts = d.ts
      and d.panel='metabolic' and d.parm='BUN/Creat Ratio'
    left outer join labdat as e ON v.id = e.id and v.ts = e.ts
      and e.panel='metabolic' and e.parm='Calcium'
    left outer join labdat as f ON v.id = f.id and v.ts = f.ts
      and f.panel='metabolic' and f.parm='Sodium'
    left outer join labdat as g ON v.id = g.id and v.ts = g.ts
      and g.panel='metabolic' and g.parm='Potassium'
    left outer join labdat as h ON v.id = h.id and v.ts = h.ts
      and h.panel='metabolic' and h.parm='Chloride'
    left outer join labdat as i ON v.id = i.id and v.ts = i.ts
      and i.panel='metabolic' and i.parm='T.Bilirubin'
    left outer join labdat as j ON v.id = j.id and v.ts = j.ts
      and j.panel='metabolic' and j.parm='AST(SGOT)'
  ) as mb_a (id,ts, a,b,c,d,e,f,g,h,i,j) ON v.id=mb_a.id and v.ts=mb_a.ts
  inner join (
    SELECT p.id, v.ts, a.value, b.value, c.value, d.value, e.value,
           f.value, g.value, h.value
    FROM patients as p
    inner join visits as v ON p.id = v.id
    left outer join labdat as a ON v.id = a.id and v.ts = a.ts
      and a.panel = 'metabolic' and a.parm = 'ALK Phosphatase'
    left outer join labdat as b ON v.id = b.id and v.ts = b.ts
      and b.panel = 'metabolic' and b.parm = 'T.Protein'
    left outer join labdat as c ON v.id = c.id and v.ts = c.ts
      and c.panel = 'metabolic' and c.parm = 'Albumin'
    left outer join labdat as d ON v.id = d.id and v.ts = d.ts
      and d.panel = 'metabolic' and d.parm = 'Globulin'
    left outer join labdat as e ON v.id = e.id and v.ts = e.ts
      and e.panel = 'metabolic' and e.parm = 'A/G Ratio'
    left outer join labdat as f ON v.id = f.id and v.ts = f.ts
      and f.panel = 'metabolic' and f.parm = 'CO2'
    left outer join labdat as g ON v.id = g.id and v.ts = g.ts
      and g.panel = 'metabolic' and g.parm = 'ALT(SGPT)'
    left outer join labdat as h ON v.id = h.id and v.ts = h.ts
      and h.panel = 'metabolic' and h.parm = 'D.Bilirubin'
  ) as mb_b (id,ts, a,b,c,d,e,f,g,h) ON v.id = mb_b.id and v.ts = mb_b.ts
) as metabolic (
  id, ts, aa, ab, ac, ad, ae, af, ag, ah, ai, aj, ba, bb, bc, bd, be, bf, bg, bh
) ON v.id = metabolic.id and v.ts = metabolic.ts
;

```

To screen out a single person, replace all occurrences of

```
patients as p
```

with

```
(VALUES 123) as p (id)
```

although, at present, only DB2 supports the VALUES keyword in a fullselect statement.

To screen out a group of people, replace all occurrences of

```
patients as p
```

with, for example,

```
(SELECT id FROM patients WHERE gender = 'm' and age between 20 and 44) as p (id)
```

DATABASE DEFINITION STATEMENTS

```
create table patients (  
  id integer not null,  
  gender char(1),  
  age smallint,  
  primary key (id)  
);  
create table visits (  
  id integer not null,  
  ts timestamp not null,  
  primary key (id, ts)  
);  
create table labdat (  
  id integer not null,  
  ts timestamp not null,  
  panel char(16) not null,  
  parm char (16) not null,  
  value double,  
  unit char(32),  
  primary key (id, ts, panel, parm)  
);  
insert into patients (id, gender, age) values  
  (123, 'm', 26),  
  (456, 'f', 29),  
  (789, 'm', 32),  
  (1005, 'm', 30)  
;  
insert into visits (id, ts) values  
  (123, timestamp('2005-02-22-10.20.53.000000')),  
  (456, timestamp('2004-09-16-09.10.05.000000')),  
  (456, timestamp('2005-02-22-13.35.24.000000')),  
  (789, timestamp('2004-11-06-15.05.42.000000'))  
;  
insert into labdat (id, ts, panel, parm, value, unit) values  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'WBC', 0, 'cells/uL/cu mm'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'RBC', 0, 'million cells/uL/cu mm'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'Hemoglobin', 0, 'gm/dL'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'Hematocrit', 0, '%'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'MCV', 0, 'cu um'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'MCH', 0, 'pg/cell'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'MCHC', 0, '% hemoglobin/cell'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'Platelets', 0, 'counts/mL'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'MPV', 0, ''),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'RDW', 0, ''),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'AGC', 0, ''),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'PMN', 0, ''),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'ESR', 0, 'mm/hr'),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'INR', 0, ''),  
  (456, timestamp('2004-09-16-09.10.05.000000'), 'cbc', 'PT', 0, 'sec'),  
  (456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'WBC', 0, 'cells/uL/cu mm'),
```

```

(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'RBC', 0, 'million/uL/cu mm'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'Hemoglobin', 0, 'gm/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'Hematocrit', 0, '%'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'MCV', 0, 'cu um'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'MCH', 0, 'pg/cell'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'MCHC', 0, '% hemoglobin/cell'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'Platelets', 0, 'counts/mL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'RDW', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'AGC', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'PMN', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'ESR', 0, 'mm/hr'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'INR', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'cbc', 'PT', 0, 'sec'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'Cholesterol', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'Triglyceride', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'HDL', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'LDL', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'VLDL', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'Chol/HDL Ratio', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'lipid', 'LDL/HDL Ratio', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'Free T4', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'TSH', 0, 'u/uL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'Glycohemoglobin', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'T3 Total', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'FTI', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'thyroid', 'T3 Uptake', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Glucose', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'BUN', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Creatinine', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'BUN/Creat Ratio', 0, '-'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Calcium', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Sodium', 0, 'mEq/L'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Potassium', 0, 'mEq/L'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Chloride', 0, 'mEq/L'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'T.Bilirubin', 0, 'mg/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'AST(SGOT)', 0, 'units/L'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'ALK Phosphatase', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'T.Protein', 0, 'gm/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Albumin', 0, 'gm/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'Globulin', 0, 'gm/dL'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'A/G Ratio', 0, '-'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'CO2', 0, 'mm Hg'),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'ALT(SGPT)', 0, ''),
(456, timestamp('2005-02-22-13.35.24.000000'), 'metabolic', 'D.Bilirubin', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'WBC', 0, 'cells/uL/cu mm'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'RBC', 0, 'million/uL/cu mm'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'Hemoglobin', 0, 'gm/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'Hematocrit', 0, '%'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'MCV', 0, 'cu um'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'MCH', 0, 'pg/cell'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'MCHC', 0, '% hemoglobin/cell'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'Platelets', 0, 'counts/mL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'MPV', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'RDW', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'AGC', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'PMN', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'ESR', 0, 'mm/hr'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'INR', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'cbc', 'PT', 0, 'sec'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'Cholesterol', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'Triglyceride', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'HDL', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'LDL', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'VLDL', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'Chol/HDL Ratio', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'lipid', 'LDL/HDL Ratio', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'Free T4', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'TSH', 0, 'u/uL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'Glycohemoglobin', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'T3 Total', 0, ''),

```

```

(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'FTI', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'thyroid', 'T3 Uptake', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Glucose', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'BUN', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Creatinine', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'BUN/Creat Ratio', 0, '-'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Calcium', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Sodium', 0, 'mEq/L'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Potassium', 0, 'mEq/L'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Chloride', 0, 'mEq/L'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'T.Bilirubin', 0, 'mg/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'AST (SGOT)', 0, 'units/L'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'ALK Phosphatase', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'T.Protein', 0, 'gm/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Albumin', 0, 'gm/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'Globulin', 0, 'gm/dL'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'A/G Ratio', 0, '-'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'CO2', 0, 'mm Hg'),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'ALT (SGPT)', 0, ''),
(123, timestamp('2005-02-22-10.20.53.000000'), 'metabolic', 'D.Bilirubin', 0, 'mg/dL')
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'WBC', 0, 'cells/uL/cu mm'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'RBC', 0, 'million/uL/cu mm'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'Hemoglobin', 0, 'gm/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'Hematocrit', 0, '%'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'MCV', 0, 'cu um'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'MCH', 0, 'pg/cell'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'MCHC', 0, '% hemoglobin/cell'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'Platelets', 0, 'counts/mL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'MPV', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'RDW', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'AGC', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'PMN', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'ESR', 0, 'mm/hr'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'INR', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'cbc', 'PT', 0, 'sec'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'Cholesterol', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'Triglyceride', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'HDL', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'LDL', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'VLDL', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'Chol/HDL Ratio', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'lipid', 'LDL/HDL Ratio', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'Free T4', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'TSH', 0, 'u/uL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'Glycohemoglobin', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'T3 Total', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'FTI', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'thyroid', 'T3 Uptake', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Glucose', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'BUN', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Creatinine', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'BUN/Creat Ratio', 0, '-'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Calcium', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Sodium', 0, 'mEq/L'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Potassium', 0, 'mEq/L'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Chloride', 0, 'mEq/L'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'T.Bilirubin', 0, 'mg/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'AST (SGOT)', 0, 'units/L'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'ALK Phosphatase', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'T.Protein', 0, 'gm/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Albumin', 0, 'gm/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'Globulin', 0, 'gm/dL'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'A/G Ratio', 0, '-'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'CO2', 0, 'mm Hg'),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'ALT (SGPT)', 0, ''),
(789, timestamp('2004-11-06-15.05.42.000000'), 'metabolic', 'D.Bilirubin', 0, 'mg/dL')

```