# Complex Queries and a VOQL

Jeff Lusted
University of Leicester

# The Twin Study - MySQL

When MySQL was developed the founders participated in a large research project involving data collected on all twins in Sweden older than 65 Years.

Select...
From...
Where...

```
SELECT
CONCAT(p1.id, p1.tvab) + 0 AS tvid,
CONCAT(p1.christian_name, ' ', p1.surname) AS Name,
p1.postal_code AS Code,
p1.city AS City,
pg.abrev AS Area,
IF(td.participation = 'Aborted', 'A', ' ') AS A,
p1.dead AS dead1,
l.event AS event1,
td.suspect AS tsuspect1,
id.suspect AS isuspect1,
td.severe AS tsevere1,
id.severe AS isevere1,
p2.dead AS dead2,
l2.event AS event2,
h2.nurse AS nurse2,
h2.doctor AS doctor2,
td2.suspect AS tsuspect2,
id2.suspect AS isuspect2,
td2.severe AS tsevere2,
id2.severe AS isevere2,
l.finish_date
```

```
FROM
twin_project AS tp
LEFT JOIN twin_data AS td ON tp.id = td.id
AND tp.tvab = td.tvab
LEFT JOIN informant_data AS id ON tp.id = id.id
AND tp.tvab = id.tvab
LEFT JOIN harmony AS h ON tp.id = h.id
AND tp.tvab = h.tvab
LEFT JOIN lentus AS l ON tp.id = l.id
AND tp.tvab = l.tvab
LEFT JOIN twin_data AS td2 ON p2.id = td2.id
AND p2.tvab = td2.tvab
LEFT JOIN informant_data AS id2 ON p2.id = id2.id
AND p2.tvab = id2.tvab
LEFT JOIN harmony AS h2 ON p2.id = h2.id
AND p2.tvab = h2.tvab
LEFT JOIN lentus AS l2 ON p2.id = l2.id
AND p2.tvab = l2.tvab,
person_data AS p1,
person_data AS p2,
postal_groups AS pg
```

```
WHERE
p1.id = tp.id AND p1.tvab = tp.tvab AND
p2.id = p1.id AND p2.ptvab = p1.tvab AND
tp.survey_no = 5 AND
(p2.dead = 0 OR p2.dead = 9 OR
(p2.dead = 1 AND
(p2.death_date = 0 OR
(((TO_DAYS(p2.death_date) - TO_DAYS(p2.birthday)) / 365)
>= 65))))
AND
(
(td.future_contact = 'Yes' AND td.suspect = 2) OR
(td.future_contact = 'Yes' AND td.suspect = 1
AND id.suspect = 1) OR
(ISNULL(td.suspect) AND id.suspect = 1
AND id.future_contact = 'Yes') OR
(td.participation = 'Aborted'
AND id.suspect = 1 AND id.future_contact = 'Yes') OR
(td.participation = 'Aborted' AND ISNULL(id.suspect)
AND p2.dead = 0))
AND
l.event = 'Finished'
AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
OR h.status = 'Died' OR h.status = 'Other')
```

# Courtesy of Richard McMahon.

I think this query is looking for distinct objects, ie: ones with no close neighbours, but I'm not certain...

Select...
From...
Where...

```
SELECT
distinct ls.sourceid,ls.ra,ls.dec,
  rtrim(substring
   (mfy.filename,charindex("w2",mfy.filename,1),32))
   as yfilename,
  rtrim(substring
   (mfj.filename,charindex("w2",mfj.filename,-1),32))
   as j_1filename,
  rtrim(substring
   (mfh.filename,charindex("w2",mfh.filename,-1),32))
   as hfilename,
  rtrim(substring
   (mfk.filename,charindex("w2",mfk.filename,-1),32))
   as kfilename,
  lml.yenum as yextnum,
  lml.j_1enum as j_1extnum,
  lml.henum as hextnum,
  lml.kenum as kextnum
```

```
FROM
 ( SELECT sourceID,T2.*
    FROM  ukidssdr1plus..lasSource
    LEFT JOIN (
      SELECT masterObjID,count(*) AS numNeighbs,
                  MIN(distanceMins) AS minSep
      FROM   ukidssdr1plus..lasSourceNeighbours
      GROUP BY masterObjID
    ) AS T2 ON sourceID=T2.masterObjID
  ) AS T1 LEFT JOIN lasSourceNeighbours as X
  ON T1.sourceID=X.masterObjID,
  ukidssdr1plus..LasSource as ls,
  ukidssdr1plus..Lasdetection as ldy,
  ukidssdr1plus..Lasdetection as ldj,
  ukidssdr1plus..Lasdetection as ldh,
  ukidssdr1plus..Lasdetection as ldk,
  ukidssdr1plus..Lasmergelog as lml,
  ukidssdr1plus..Multiframe as mfy,
  ukidssdr1plus..Multiframe as mfj,
  ukidssdr1plus..Multiframe as mfh,
  ukidssdr1plus..Multiframe as mfk
```

```
WHERE
 (T1.numneighbs is null or
  T1.minSep >= 4.0/60.0) and
 T1.sourceid=ls.sourceid and
 ldy.objid=ls.yobjid and
 ldj.objid=ls.j_1objid and
 ldh.objid=ls.hobjid and
 ldk.objid=ls.kobjid and
 ls.framesetid=lml.framesetid and
 mfy.multiframeID=lml.ymfid and
 mfj.multiframeID=lml.j_1mfid and
 mfh.multiframeID=lml.hmfid and
 mfk.multiframeID=lml.kmfid and
 ((ls.j_1apermag3-ls.kapermag3) >= 2.5 or ls.j_1class < -500) and
 (ls.yclass < -500 or
 (ls.yclass > -500 and ls.j_1class > -500 and
  ls.yapermag3 > ls.j_1apermag3)) and
 ls.kapermag3 >= 12.5 and
 ls.kapermag3 <= 17.0 and
 ls.kapermag3err <= 0.1 and
 ldk.x > 64 and
 ldk.y > 64 and
 ldk.x < 4060 and
 ldk.y < 4060 and
 ls.kclass = -1 and
 ls.hclass > -500 and
 (ls.priorsec = 0 or ls.priorsec = ls.framesetid)
 lml.yenum > 0 and lml.j_1enum > 0 and
 lml.henum > 0 and lml.kenum > 0
```

# Did You Spot The Errors?

<<== Go Back

# Two Fundamental Problems

- Difficult to understand

  *"How am I supposed to understand this?"*

- Difficult to be sure you got the data intended (even if you think you understood it!)

  *"How do I know it is right?"*

# A Future Prospect?

Scientists will:

- Struggle with interesting and difficult problems in their own domain.
- Struggle with boring and increasingly complex problems in SQL

# Clarity.

# **Use a Set-Based Language.**

- With a better level of abstraction
- Capable of piecemeal development
- Capable of supporting extensions
- With reasonable levels of re-usability

*But the end product is still usable SQL!*

# Block Structured

```
Query {

include <file name>

    Extrns {} ;
    Macro <name> {} ;
    Map <name> {} ;
    Set <name> {} ;

};
```

# Block Level Language

- An EXTRNS block for specific environment
- One or more MACRO blocks for re-usability
- One or more MAP blocks for UCD's
- (One or more MODEL blocks for data models)
- One or more SET blocks to contain the query

```
Extrns {
     table: LasSource, owner ukidssdr1plus ;
     table: LasSourceNeighbours, owner ukidssdr1plus ;
     result: ObjectsWithMinSeparationData, limit 10000 ;
};
Set MinimumSeparationData {
     from: lasSourceNeighbours ;
     attribute: masterObjID,  numNeighbs: count(*) ,
                    minSep: min( distanceMins ) ;
     group: by masterObjID ;
};
Set SourceFilter {
     from: LasSource ;
     condition:  . . .   ;
     condition: kapermag3 >= 12.5 ;
};
Set ObjectsWithMinimumSeparationData {
     from mp : MinimumSeparationData ;
     from ls : SourceFilter ;
     attribute : ls.sourceID, . . . , mp.* ;
     join: mp left join ls on ls.sourceID = mp.masterObjID ;
};
```

# Query with Macro

```
Macro extract_name( @source, @target, @len ) {
   TRIM( TRAILING from
    SUBSTRING( @source, POSITION(@target in @source), @len)  )
};

Set DistinctObjects {
      attribute yfilename: extract_name( mfy.filename, 'w2', 32 ) ;
      . . .
      from do: DistinctObjects_FirstCut ;
      from mlf: MergeLogFilter ;
      from  dfy, dfj, dfh, dfk: DetectionFilter ;
      from  mfy, mfj, mfh, mfk: Multiframe ;
      . . .
      condition: . . . ;
};
```

# Query with Macro and Map.

```
Extrns {
    . . .
    table: twomass_psc, owner wfau-dsacat ;
    result: result01 ;
};

include <hv-cone-search.v1-3.macro>
include <psc-ucds.map>

Set result01 {
    map: ucd ;
    map SPECT_FLUX_VALUE: h_msigcom ;
    from t: twomass_psc ;
    condition: cone_search( 83, +5, 0.001 ) ;
    condition:
    . . .
    condition:
};
```

# Macro Include expands to...

```
Macro cone_search( @RA, @DEC, @CIRCRADIUS_DEG ) {
  ( ( POS_EQ_DEC_MAIN <= @DEC + @CIRCRADIUS_DEG )
AND  ( POS_EQ_DEC_MAIN >= @DEC - @CIRCRADIUS_DEG )  )
  AND ( ( POS_EQ_RA_MAIN >= @RA -
     CASE WHEN @CIRCRADIUS_DEG >= PI() / 2 - ABS( @DEC )
       THEN PI()
       ELSE ASIN( COS( PI()/2 - @CIRCRADIUS_DEG ) /
              SIN (PI()/2 - ABS( @DEC )) )
      ) AND ( POS_EQ_RA_MAIN <= @RA +
     CASE WHEN @CIRCRADIUS_DEG >= PI() / 2 - ABS( @DEC )
       THEN PI()
       ELSE ASIN( COS( PI()/2 - @CIRCRADIUS_DEG ) /
              SIN (PI()/2 - ABS( @DEC ) ) ) ) )
  AND ( (2.0 * ASIN(SQRT(
      POWER(SIN((( POS_EQ_DEC_MAIN -
          @DEC) / 2.0)), 2)  +  ( (COS( POS_EQ_DEC_MAIN )  *
           ( COS(@DEC)  *  POWER(SIN( ( ( POS_EQ_RA_MAIN -
                @RA) / 2.0 ) ), 2 ) ) ) ) ) ) )
    < @CIRCRADIUS_DEG  )
};
```

# Map Include expands to...

```
Map ucd {

  POS_EQ_RA_MAIN: ra ;
  POS_EQ_DEC_MAIN: dec ;
  POS_EQ_X: cx;
  POS_EQ_Y: cy;
  POS_EQ_Z: cz;
. . .
  SPECT_FLUX_VALUE: j_m, j_cmsig, h_m, h_cmsig,
                                    h_msigcom, . . . , vr_m_opt;
};
```

# And Other Functionalities...

# Named Conditions...

```
Extrns {
    . . .
    table: twomass_psc, owner wfau-dsacat ;
    result: result01 ;
};

include <hv-cone-search.v1-3.macro>

Set result01 {
    . . .
    from t: twomass_psc ;

    condition: cone_search( calcRA, calcDec, calcRad ) ;
    condition calcRa:  . . . ;
    condition calcDec:  . . . ;
    condition calcRad:  . . . ;
    condition: . . . ;
};
```

**(1) Results in VOSpace**
**(2) Virtual Includes**
**(3) Conditional Includes**

```
Extrns {
    . . .
    table: twomass_psc, owner wfau-dsacat ;
    result feintObjects:
            ivo://uk.le.ac.star#jlusted/results/feintObjects.vot  ;
};

include <wfau-dsacat-support.macro>
if htm( twomass_psc )
    include <htm-cone-search.v09.macro>
else
    include <hv-cone-search.v1-3.macro>

Set feintObjects {
    from t: twomass_psc ;
    condition: cone_search( calcRA, calcDec, calcRad ) ;
    condition: . . . ;
};
```

# Virtual Tables.

```
Extrns {
    . . .
    table: twomass_psc, owner wfau-dsacat ;
    result: result01 ;
};

Set result01 {
    base: twomass_psc ;
    row:
    row:
    row:
    row:
};
```