

SQL5

Version 19.3

DmitryTurin@narod.ru

sql50.euro.ru

Public Domain

Current standards

1. SQL Standard - SQL/XML Functionality. ISO/IEC JTC1/SC32 #1293, 2005-04-22.
<http://jtc1sc32.org/doc/N1251-1300/32N1293-WG3-Presentation-for-SC32-20050418.pdf>
2. K.Kulkarni. Overview of SQL:2003, part 14 "SQL/XML", p. 65. Silicon Valley Laboratory, IBM Corporation, San Jose, 2003-11-06.
<http://www.wiscorp.com/SQL2003Features.pdf>
3. N.Mattos, H.Darwen, P.Cotton, P.Pistor, K.Kulkarni, S.Dessloch, K.Zeidenstein. SQL99, SQL/MM, and SQLJ: An Overview of the SQL standards.
http://www.wiscorp.com/sql1999_c3.zip
4. How to add style to XML. //W3C papers. <http://www.w3.org/Style/styling-XML>
5. CSS vs. XSL. //W3C papers. <http://www.w3.org/Style/CSS-vs-XSL>
6. XML Path Language (XPath). Version 1.0. //W3C Recommendation, 16 November 1999. <http://www.w3.org/TR/xpath>
7. XML submission. //Web Forms 2.0, Working Draft, 12 October 2006.
<http://www.whatwg.org/specs/web-forms/current-work/#x-www-form-xml>
8. P.Scarponcini. SQL Multimedia and Application Packages - Part 3: Spatial. Bentley Transportation, 2000-03-26. <http://www.wiscorp.com/sqlspat.zip>

Proposals

Hierarchical input-output

[Concept for output](#)

[SQL/XLang output](#)

[Conditions](#)

[Compositions of determinations,
compositions of refinements](#)

[Portion of sorted data](#)

[Aggregates vs. recursion](#)

[Wave algorithm \(on examples\)](#)

[Output, optionally](#)

[Concept for input](#)

[Manually](#)

[Manually into content](#)

[Input, optionally](#)

[Concept for processing](#)

[SQL, XML just over TCP](#)

Distributed query

[Concept for dQ](#)

[Implementation](#)

[Examples of replication](#)

[dQ, optionally](#)

[BLOB](#)

[Nested schemas like former folders](#)

[Repairing of dQ, optionally](#)

Different

[Whole control](#)

[Controllable and usable FTS](#)

[Freezing](#)

[Timer](#)

[Interval table constraints](#)

Model - Window System

[Concept for 3D](#)

[2D-grid in 3D](#)

[3D-triggers](#)

[Mouse triggers](#)

[Operations](#)

[3D-contour, optionally](#)

[Concept for 2D](#)

[2D-triggers](#)

[Mouse triggers](#)

[1D, optionally](#)

[Window format, optionally](#)

[Mixture of dimensions](#)

[Fielders, tablers](#)

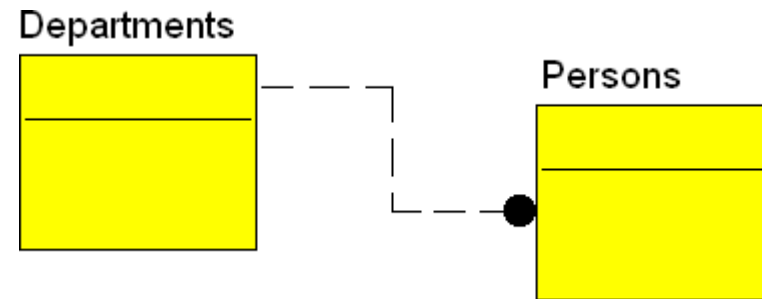
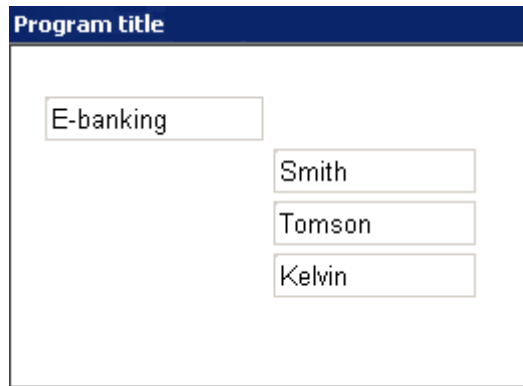
Hierarchical input-output (hIO)

Attention: XML is used instead of binary format only to represent hierarchical data from communication channel onto slides

Concept for output

Screen schema = database schema

Output fields are bound by relations IDEF1. Screen of program has schema, and this schema completely repeats schema of database



Syntax, expressing hierarchy, e.g. "Departments.Persons" is absent in SQL at all. Client cannot separate in Cartesian products, what fields belong to one table, or another. So taking information from database, we lose a part of information!

Tree notation vs. LOOP/FETCH

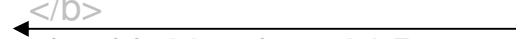
Instead of inserting lost information by machine, we try to make it manually, forcing user to trace work of computer. It occupies up to 50% of working time. Tracing is written in notation LOOP/FETCH. We take one record at a time to find records, bound with it.

Strong decision exists.
It is expressions like
'select * from a.b.c' to
select data from tables
'a', 'b', 'c', already bound
by foreign keys. Let's
name this by term 'XTree'
(in analogy with 'XPath').

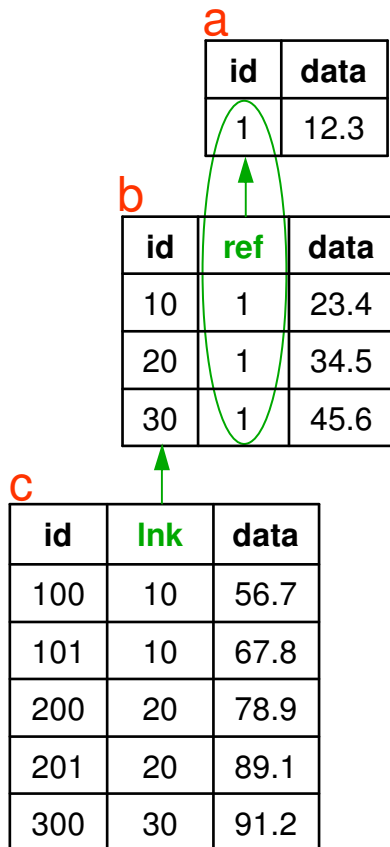
```
<a id=1 data=12.3>  
<b id=10 data=23.4>  
  <c id=100 data=56.7/>  
  <c id=101 data=67.8/>  
</b>  
<b id=20 data=34.5>  
  <c id=200 data=78.9/>  
  <c id=201 data=89.1/>  
</b>  
</a>
```

DBMS

select *
from a.b.c;



Selecting **set**

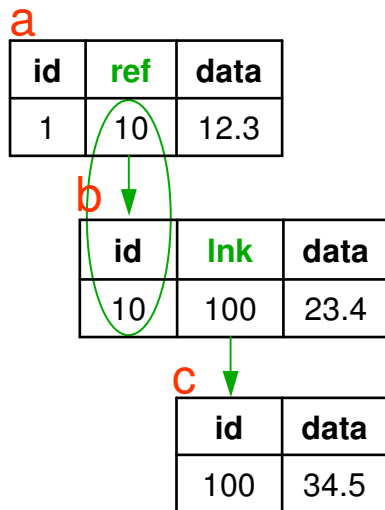


select * from a.b.c;

```
<a id=1 data=12.3>
  <b id=10 data=23.4>
    <c id=100 data=56.7/>
    <c id=101 data=67.8/>
  </b>
  <b id=20 data=34.5>
    <c id=200 data=78.9/>
    <c id=201 data=89.1/>
  </b>
  <b id=30 data=45.6>
    <c id=200 data=91.2/>
  </b>
</a>
```

<!-- used referring fields are **not** extracted -->

Selecting relay-race



select * from a.b.c;

```
<a id=1 data=12.3>
```

```
  <b id=10 data=23.4>
```

```
    <c id=100 data=34.5/>
```

```
  </b>
```

```
</a>
```

```
<!-- used refering fields are not extracted -->
```

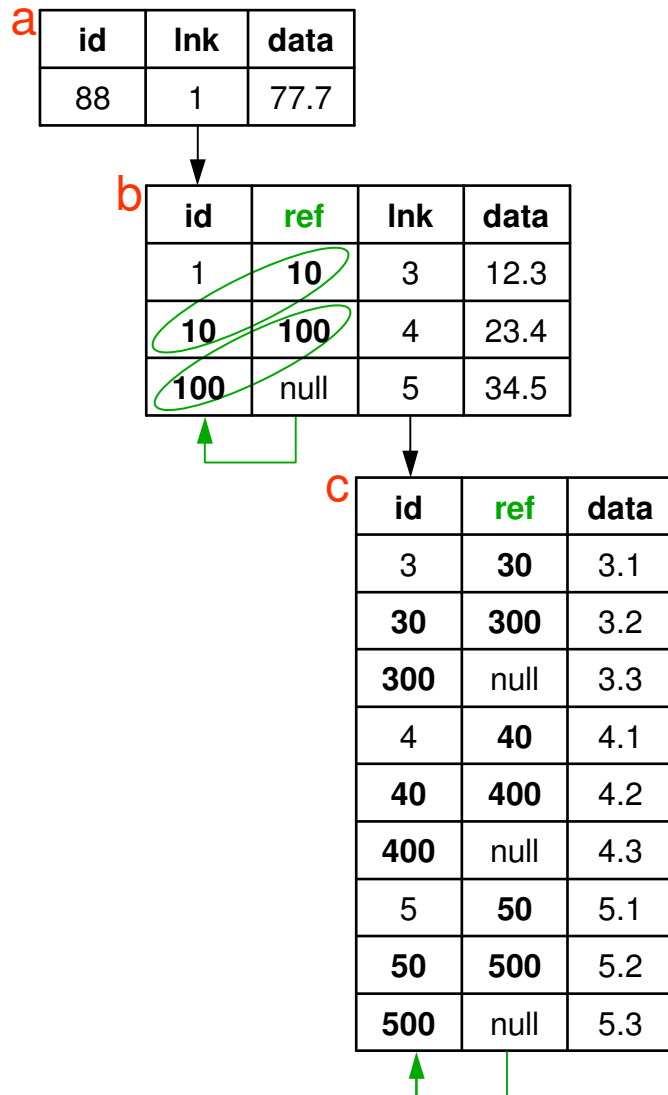
Postulates

- **FK** of table **to itself** is always means **list**, instead of nesting (transforming list to nesting is archived by special statement, [slide 29](#))
- **Previous** element of list refers **to next** element (instead of next element refers to previous)

Restriction to obtain particular **list** instead of all lists may be:

- tree, in which list is (e.g. “select **b** from a.b.c;”), or
- predicate for element of list (e.g. “select * from **b** where first(**b**)/@b1=5;”)

Selecting list



select * from a.b.c;

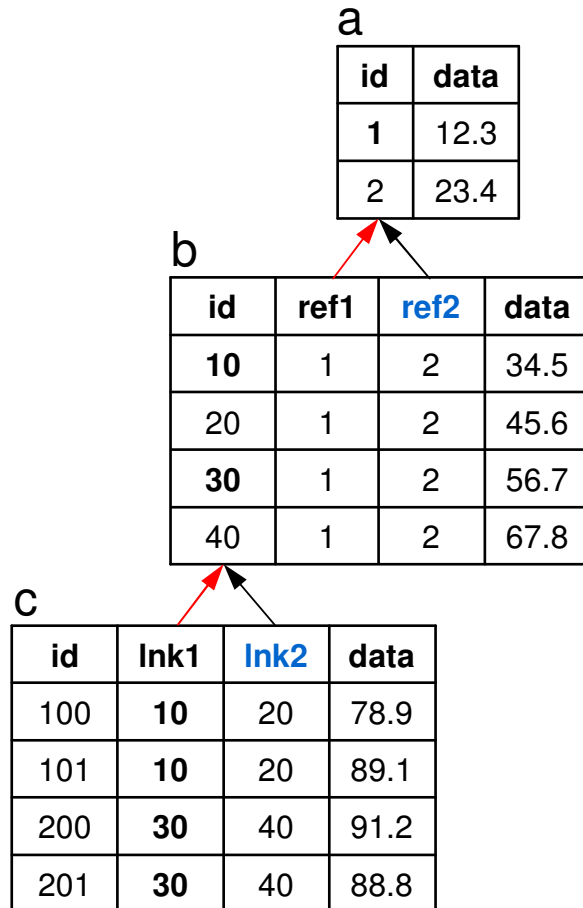
```
<a id=88 data=77.7>
  <b id=1 data=12.3>
    <c id=3 data=3.1/>
    <c id=30 data=3.2/>
    <c id=300 data=3.3/>
  </b>
  <b id=10 data=23.4>
    <c id=4 data=4.1/>
    <c id=40 data=4.2/>
    <c id=400 data=4.3/>
  </b>
  <b id=100 data=34.5>
    <c id=5 data=5.1/>
    <c id=50 data=5.2/>
    <c id=500 data=5.3/>
  </b>
</a>
```

<!-- used referring fields are not extracted -->

Refinement for set and relay-race

If two tables refer to each other, or one table refers to another by several foreign keys, than this ambiguity is solved in query: Sign '#' and name of necessary referring field (not name of constraint, i.e. name of foreign key) are specified after strictly name of table, containing necessary foreign key. It looks like new table name with symbol '#' in the middle of the name. We shall name this specifying of referring field by term 'refinement'.

Refinement for set



```
select * from a.b#ref1.c#Ink1;
```

```
<a      id=1      data=12.3>
  <b#ref1 id=10 ref2=2 data=34.5>
    <c#Ink1 id=100 Ink2=20 data=78.9/>
    <c#Ink1 id=101 Ink2=20 data=89.1/>
  </b>
  <b#ref1 id=30 ref2=2 data=56.7>
    <c#Ink1 id=200 Ink2=40 data=91.2/>
    <c#Ink1 id=201 Ink2=40 data=88.8/>
  </b>
</a>
```

<!-- un-used refering fields are extracted -->

Refinement for relay-race

a

id	ref1	ref2	data
1	10	20	12.3

b

id	lnk1	lnk2	data
20	200	201	34.5
10	100	101	23.4

c

id	data
201	78.9
200	67.8
101	56.7
100	45.6

select * from a#ref1.b#lnk1.c;

```
<a#ref1 id=1 ref2=20 data=12.3>  
  <b#lnk1 id=10 lnk2=101 data=23.4>  
    <c id=100 data=45.6/>  
  </b>  
</a>
```

<!-- un-used referring fields are extracted -->

Refinement for list

Similarly, if table contains a list and refers to self by several foreign keys, than this ambiguity is also solved in query: Sign '\$' and name of necessary referring field (not name of constraint, i.e. name of foreign key) are specified after strictly name of table, containing necessary foreign key. It also looks like new table name with symbol '\$' in the middle of the name. We shall name this specifying of referring field by term 'refinement' too.

Different signs - '#' and '\$' - are used in two different types of refinement, that it would possible to use both types of refinement simultaneously: 'table#field1\$field2'

Refinement for list

a

id	Ink	data
88	1	77.7

b

id	ref1	ref2	Ink	data
1	10	20	3	12.3
10	null	200	4	23.4

c

id	ref3	ref4	data
3	30	70	3.1
30	300	700	3.2
300	null	null	3.3
4	40	80	4.1
40	400	800	4.2
400	null	null	4.3

select * from a.b\$ref1.c\$ref3;

```

<a      id=88      data=77.7>
  <b$ref1 id=1  ref2=20 data=12.3>
    <c$ref3 id=3  ref4=70 data=3.1/>
    <c$ref3 id=30 ref4=700 data=3.2/>
    <c$ref3 id=300      data=3.3/>
  </b>
  <b$ref1 id=10 ref2=200 data=23.4>
    <c$ref3 id=4  ref4=80 data=4.1/>
    <c$ref3 id=40 ref4=800 data=4.2/>
    <c$ref3 id=400      data=4.3/>
  </b>
</a>

```

<!-- un-used refering fields are extracted -->

Another output

```
select tab2/@data2, tab1/@data1  
from tab1, tab2 -- tab1 is first  
where tab2/@fld2=tab1/@fld1;
```

```
<tab1 data2=3 data1=12.3 />  
<tab1 data2=7 data1=23.4 />  
<tab1 data2=10 data1=34.5 />  
<tab1 data2=25 data1=45.6 />
```

Switching designations

set dialect sql5;



```
select a.a1 , b.b1
from schema.tab
where b.b1=5
group by x1, y1
```

```
update d set d1=5
```

```
insert into b
delete from
```

```
create procedure p(v type1)
```

```
references tab(@fld1)
references tab(@fld1, @fld2)
foreign key
not null
```

```
select a/@a1.b/@b1
from schema$tab
where b/@b1=5
group @c by @x1, @y1
group @c by x1, y1 --exists
```

```
update d set @d1=5
update d set d1/@e1=5 --exists
```

```
insert into b (@)
delete * from
```

```
create procedure p (@v type1)
```

```
refers tab/@fld1
refers (tab[@fld1, @fld2])
foreign
notnull
```

set dialect sql4;



Switching and enquiring session parameter

```
set dialect sql5;      -- SQL5
```

```
set dialect sql4;      -- SQL:2003
```

```
show dialect;  ==>>>  <?res code=0 dialect=sql4 /?>
```

SQL/XLang output

XLang

XPath

a/(b | k)/c

a/b*/c

a/b+/c

a/b?/c

a/b[@b1]/c

a/b[@b1=5]/c

a/b[0]/c

a/(b | k)[0]/c

a//b

XTree

a.(b | k).c

a.b*.c

a.b+.c

a.b?.c

a.b[@b1].c

a.b[@b1=5].c

a.b[0].c

a.(b | k)[0].c

a..b

tab[!=5] -- level of nesting

E.g.

'*' means repetition of section (of table) from 0 to infinity times

a.b*.c means **all** following trees

a.c

a.b.c

a.b.b.c

a.b.b.b.c

etc

'?' means presence or absent of section (of table)

a.b?.c means **both** following trees

a.c

a.b.c

'+' means repetition of section (of table) from 1 to infinity times

a.b+.c means **all** following trees

a.b.c

a.b.b.c

a.b.b.b.c

a.b.b.b.b.c

etc

'..' means any intermediate sections (tables) in any quantity

a..c means **all** following trees

a.c

a.m.c

a.n.c

a.n.n.c

a.n.k.c

a.n.k.k.c

etc

p/q/a.b.c.d means the same as p.q.a.b.c.d but p.q are not extracted

Extracted tables

```
select *  
from p/q/a.b.c.d;
```

```
select *  
from p/q/a.b[3,5].c.d;
```

```
select b.d  
from p/q/a.b.c.d;
```

```
select b.d  
from p/q/a.b[3,5].c.d;
```

```
select b[3,5].d  
from p/q/a.b[11-18].c.d;
```

-- b[13,15] to output

```
list  
  <a  a1=v1  a2=v2 >  
    <b  b1=1  b2=2 >  
      <c  c1=w1  c2=w2 >  
        <d  d1=10  d2=20><d  d1=30  d2=40>  
      </c>  
    </b>  
  <b  b1=3  b2=4 >  
    <c  c1=w3  c2=w4 >  
      <d  d1=50  d2=60><d  d1=70  d2=80>  
    </c>  
  </b>  
</a>  
  
<b  b1=1  b2=2 >  
  <d  d1=10  d2=20><d  d1=30  d2=40>  
</b>  
  
<b  b1=3  b2=4 >  
  <d  d1=50  d2=60><d  d1=70  d2=80>  
</b>
```

Extracted fields

```
select b [@b1].d[@d1]
from p/q/a.b.c.d;
-- only these fields
```

```
<b b1=1>
  <d d1=10><d d1=30>
</b>
<b b1=3>
  <d d1=50><d d1=70>
</b>
```

```
select b [@b1 as m].d[@d1 as n]
from p/q/a.b.c.d;
-- only these fields, but with renaming
```

```
<b m=1>
  <d n=10><d n=30>
</b>
<b m=3>
  <d n=50><d n=70>
</b>
```

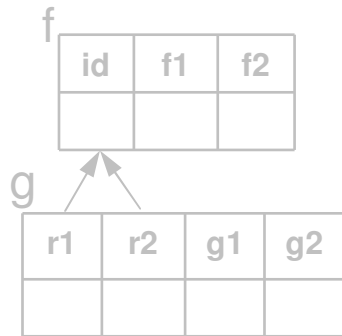

Binding two trees (BTT)

```
select a.b.k.m  
from p/q/a.b ,s/t/k.m  
where b/@b1=k/@k1;
```

Aliases for section

```
select f.g.x
from f x, -- it's impossible 'f.g.x' from 'f' and 'x'
      f.g.x;
```

```
<f f1= f2= >
  <g g1= g2= >
    <x f1= f2= />
  </g>
</f>
```



Aliases for whole tree

```
select *
from f.g k.m; -- rename tree

select f.g as k.m -- rename tree
from f.g k.m;
```

```
<k f1= f2= >
  <m g1= g2= />
</k>
```

Aliases and BTT

```
select k.m.n
  from p/q/c k m n
where k/@c2=m/@c3 and m/@c4=n/@c5;
```

```
select k.m.n
  from p/q/a k, r/s/b m, t/u/c n,
where k/@a2=m/@b2 and m/@b3=n/@c3;
```

```
select k.m.n.v
  from p/q/a.b k.m, r/s/c.d n.v -- renaming trees
where m/@b2=n/@c2;
```

Re-alignment tree to list (only for non-branching tree)

```
select a from p/q/a*;  
select a from p/q/(a.b)*;
```

```
<a a1=v1 a2=v2 />  
<a a1=v3 a2=v4 />  
<a a1=v5 a2=v6 />
```

```
select a.b from p/q/(a.b)*;
```

```
<a a1=v1 a2=v2 >  
  <b b1=w1>  
  <b b1=w2>  
</a>  
<a a1=v3 a2=v4 >  
  <b b1=w3>  
  <b b1=w4>  
</a>  
<a a1=v5 a2=v6 >  
  <b b1=w5>  
  <b b1=w6>  
</a>
```

Realignment list to tree

select **a*** from p/q/**a**;
select **a*** from p/q/**a.b**;

```
<a a1=v1 a2=v2 >  
  <a a1=v3 a2=v4 >  
    <a a1=v5 a2=v6 />  
  </a>  
</a>
```

select **(a.b)*** from p/q/**a.b**;

```
<a a1=v1 a2=v2 >  
  <b b1=w1 >  
    <a a1=v3 a2=v4 >  
      <b b1=w3 >  
        <a a1=v5 a2=v6 >  
          <b b1=w5 >  
            <b b1=w6 >  
          </a>  
        </b>  
      <b b1=w4 >  
    </a>  
  </b>  
</a>
```

Constant as section

```
select a.<u u1=8>.b  
from p/q/a.b;
```

```
<a a1=v1 a2=v2 >  
  <u u1=8>  
    <b b1=1 b2=2 />  
    <b b1=3 b2=4 />  
  </u>  
</a>  
<a a1=w1 a2=w2 >  
  <u u1=8>  
    <b b1=1 b2=2 />  
    <b b1=3 b2=4 />  
  </u>  
</a>
```

```
select a.<u u1=8 >||b||</u>  
from p/q/a.b;
```

```
<a a1=v1 a2=v2 >  
  <u u1=8>  
    <b b1=1 b2=2 />  
  </u>  
  <u u1=8>  
    <b b1=3 b2=4 />  
  </u>  
</a>  
<a a1=w1 a2=w2 >  
  <u u1=8>  
    <b b1=1 b2=2 />  
  </u>  
  <u u1=8>  
    <b b1=3 b2=4 />  
  </u>  
</a>
```

XPath continues XTree

```
select * from p/q/a.b.c
  where   q/t/u/@u1=7           -- 'p/q/t/u/@u1'
         and   b/t/u/@u1=8;     -- 'p/q/a/b/t/u/@u1'

insert into  p/q (
             q/n/k/a.b(@b1).c   -- 'p/q/n/k/a.b.c'
) values (1)
  where     q/t/u/@u1=7         -- 'p/q/t/u/@u1'
         and   b/t/u/@u1=8;     -- 'p/q/n/k/a/b/t/u/@u1'

update      p/q/a.b.c           -- similarly 'p/q/a/b.c'
  set       q/n/k/@k1=1,        -- 'p/q/n/k/@k1'
           b/n/k/@k1=2,        -- 'p/q/a/b/n/k/@k1'

  where     q/t/u/@u1=7         -- 'p/q/t/u/@u1'
         and   b/t/u/@u1=8;     -- 'p/q/a/b/t/u/@u1'

delete      b, d
  from      p/q/a.b.c.d
  where     q/t/u/@u1=7         -- 'p/q/t/u/@u1'
         and   b/t/u/@u1=8;     -- 'p/q/a/b/t/u/@u1'
```

Refinement in any place

(in XPath too)

```
select * from p#ref1/q#lnk1/a.b#k.c#m.d;
```

```
insert into p#ref1/q#lnk1/a (@data) values (88);
```

```
update p#ref1/q#lnk1/a set @data=88;
```

```
update p#ref1/q#lnk1/a (@data) select 88;
```

```
delete * from p#ref1/q#lnk1/a.b#k.c#m.d;
```


Conditions

Restriction only for set and relay-race

```
select * from a.b*.c
```

```
where parent(b)/@b2=child(b)/@b1;
```

-- functions 'parent' and 'child' are used only for set and relay-race,
must be used always simultaneously

```
<a>  
  <b          b2=1>  
    <b b1=1 b2=2>  
      <b b1=2 >  
        <c/>  
      </b>  
    </b>  
  </b>  
</a>
```

Only for set and relay-race

```
select * from a.b*.c.b*  
where root(b)/@b2=3;
```

-- restriction for first 'b' in whole tree

```
select * from a.b*.c.b*  
where terminator(b)/@b2=4;
```

-- cut tree on last 'b' in each branch, field 'b2' of which is equal 4

P.S. 'select terminator(b) from b*' is possible too

```
<a>  
<b b2=3>  
<b b2=4>  
<c>  
<b b2=4>  
<b b2=5 />  
</b>  
</c>  
</b>  
</b>  
</a>
```

```
<a>  
<b b2=3>  
<b b2=4>  
<c>  
<b b2=4 />  
</c>  
</b>  
</b>  
</a>
```

Conditions for fields

~~XPath~~

~~a.b[@b1=5]
a.b[@b1]
a.b.c[@c1=.../@a1]
a.b[position()=1 and @b1=5]*.c
a.b[last() and @b1=5]*.c~~

XTree

from .../a.b where @b1=5
from .../a.b where @b1 is not null
from .../a.b.c where a/@a1=c/@c1
from .../a.b*.c where **first**(b)/@b1=5
-- first in each list
from .../a.b*.c where **last**(b)/@b1=5
-- last in each list
from .../a.b*.c where @a1 in ...

Restriction only for list

```
select * from a.b.c
```

```
where previous(b)/@b2=next(b)/@b1;
```

-- functions 'previous' and 'next' are used only for list,
and must be used always simultaneously

```
<a>  
  <b      b2=1><c/></b>  
  <b b1=1 b2=2><c/></b>  
  <b b1=2      ><c/></b>  
</a>
```

Only for list

```
select * from a.b*.c.b*  
where first(b)/@b2=3;
```

-- restriction for first 'b' in each list

```
select * from a.b*.c.b*  
where last(b)/@b2=4;
```

-- cut tree on last 'b' in each list, field 'b2' of which is equal 4

```
<a>  
<b b2=3><c/></b>  
<b b2=4><c/></b>  
<b b2=5><c/></b>  
</a>
```

```
<a>  
<b b2=3><c/></b>  
<b b2=4><c/></b>  
<b b2=6><c/></b>  
</a>
```

```
<a>  
<b b2=7><c/></b>  
<b b2=4><c/></b>  
</a>
```

```
<a>  
<b b2=8><c/></b>  
<b b2=4><c/></b>  
</a>
```

Compositions of determinations,
compositions of refinements

Nog

```
create table a (  
  id  num  primary key,  
  data float  
);  
create table b (  
  id  num  primary key,  
  ref1 num  references a(id), -- important  
  ref2 num  references a(id), -- ballast  
  ref3 num  references c(id), -- important  
  ref4 num  references c(id), -- ballast  
  data float  
);  
create table c (  
  id  num  primary key,  
  data float  
);
```

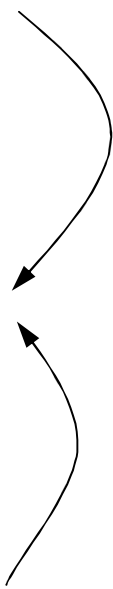


```
<a>  
  <b#ref1:ref3>  
    <c>  
  </b>  
</a>
```

```
.a.b#ref1:ref3.c.  
/a/b#ref1:ref3/c/
```


Buckle

```
create table a (  
  id num primary key,  
  ref1 num references b(id), -- important  
  ref2 num references b(id), -- ballast  
  data float  
);  
create table b (  
  id num primary key,  
  data float  
);  
create table c (  
  id num primary key,  
  ref1 num references b(id), -- important  
  ref2 num references b(id), -- ballast  
  data float  
)
```



**<a#ref1>

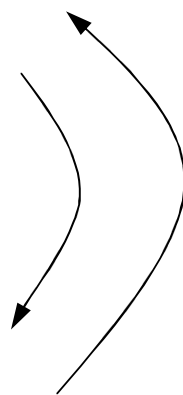
 <c#ref3>

**

**.a#ref1.b.c#ref3.
/a#ref1/b/c#ref3/**

Crossroad

```
create table a (  
  id num primary key,  
  ref num references b(id),  
  data float  
);  
create table b (  
  id num primary key,  
  lnk num references a(id),  
  data float  
);
```



```
<a#ref>  
  <b>  
</a>
```

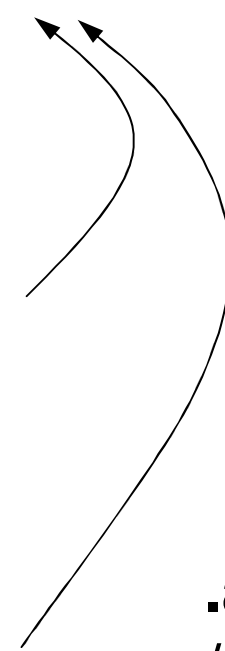
```
<a>  
  <b#lnk>  
</a>
```

```
.a#ref.b.  
/a#ref/b/
```

```
.a.b#lnk.  
/a/b#lnk/
```

Branching-1

```
create table a (  
  id num primary key,  
  data float  
);  
create table b (  
  id num primary key,  
  ref1 num references a(id), -- important  
  ref2 num references a(id), -- ballast  
  data float  
);  
create table c (  
  id num primary key,  
  ref1 num references a(id), -- important  
  ref2 num references a(id), -- ballast  
  data float  
);
```



```
<a>  
  <b#ref1>  
  <c#ref1>  
</a>
```

```
.a.(b#ref1 c#ref1).  
/a/(b#ref1 c#ref1)/  
-- both branches simultaneously
```

```
.a.(b#ref1 | c#ref1).  
-- any branch
```

```
/a/(b#ref1 | c#ref1)/  
-- path must be specified exactly
```

Branching-2

```
create table a (  
  id num primary key,  
  ref1 num references b(id), -- important  
  ref2 num references b(id), -- ballast  
  ref3 num references c(id), -- important  
  ref4 num references c(id), -- ballast  
  data float  
);  
create table b (  
  id num primary key,  
  data float  
);  
create table c (  
  id num primary key,  
  data float  
);
```

<a#ref1+ref3>

<c>

.a#ref1+ref3.(b c).
/a#ref1+ref3/(b c)/
-- both branches simultaneously

.a#ref1^ref3.(b | c).
-- any branch
~~**/a#ref1^ref3/(b | c)/**~~
-- path must be specified exactly

Continuation-1

```
create table b (  
  id num      primary key,  
  data float  
);  
create table c (  
  id num      primary key,  
  data float  
);  
create table d (  
  id num      primary key,  
  ref1 num    references b(id), -- important  
  ref2 num    references b(id), -- ballast  
  ref3 num    references c(id), -- important  
  ref4 num    references c(id), -- ballast  
  data float  
);
```

Sament can't be
inside 2 saments

.(b.d#ref1 c.d#ref3).

/(b/d#ref1 c/d#ref3)/

-- both branches simultaneously

.(b.d#ref1 | c.d#ref3).

-- any branch

~~/(b/d#ref1 | c/d#ref3)/~~

-- path must be specified exactly

Continuation-2

```
create table b (  
  id num primary key,  
  ref1 num references d(id), -- important  
  ref2 num references d(id), -- ballast  
  data float  
);  
create table c (  
  id num primary key,  
  ref1 num references d(id), -- important  
  ref2 num references d(id), -- ballast  
  data float  
);  
create table d (  
  id num primary key,  
  data float  
);
```

Sament can't be
inside 2 saments

.(b#ref1 c#ref3).d.

/((b#ref1 c#ref3)/d/

-- both branches simultaneously

.(b#ref1 | c#ref3).d.

-- any branch

~~**/((b#ref1 | c#ref3)/d/**~~

-- path must be specified exactly

Combined key

```
create table a (  
  id1 num,  
  id2 num,  
  primary key (id1, id2),  
  data float  
);  
create table b (  
  id num primary key,  
  lnk1 num,  
  lnk2 num,  
  foreign key (lnk1, lnk2) references a(id1, id2),  
  data float  
);
```

<a>
<b#lnk1*lnk2>

.a.b#lnk1*lnk2.
/a/b#lnk1*lnk2/

Refinement creates Determination

```
insert into a select * from a.b#lnk;  
insert into a select * from a#ref.b;  
insert into a select * from a.b$lnk;
```

--“select” with refinement creates xml-output with determination,
“insert” uses this determination for correct entering

Portion of sorted data

Next portion

```
select * from a.* where ...  
order by @a2 asc, @a3 desc, @a1 asc  
downward @a1=100 @a2=null @a3="string"  
limit 20;  
-- start from record @a1=100 @a2=null @a3="string",  
and select next 20 records
```

```
select * from a.* where ...  
order by @a2 asc, @a3 desc, @a1 asc  
downward (select * from a.* where ... limit 1 )  
limit 20;  
-- start from record, returned by "select * from a.* where ... limit 1",  
and select next 20 records
```

Previous portion

```
select * from a.* where ...  
order by @a2 asc, @a3 desc, @a1 asc  
upward @a1=100 @a2=null @a3="string"  
limit 20;  
-- start from so record, and select next 20 records,  
   that last selected record would be @a1=100 @a2=null @a3="string"
```

```
select * from a.* where ...  
order by @a2 asc, @a3 desc, @a1 asc  
upward (select * from a.* where ... limit 1 )  
limit 20;  
-- start from so record, and select next 20 records,  
   that last selected record would be equal to  
   returned by "select * from a.* where ... limit 1"
```

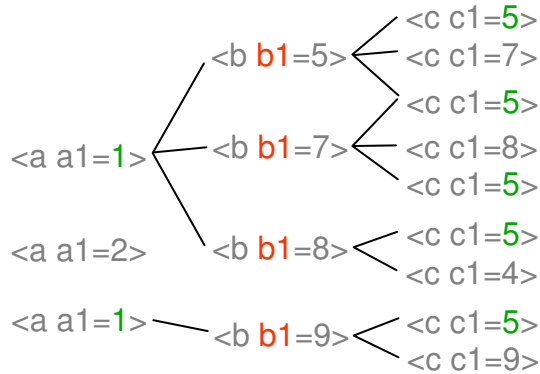
Aggregates vs. recursion

Grouping by non-list for extraction

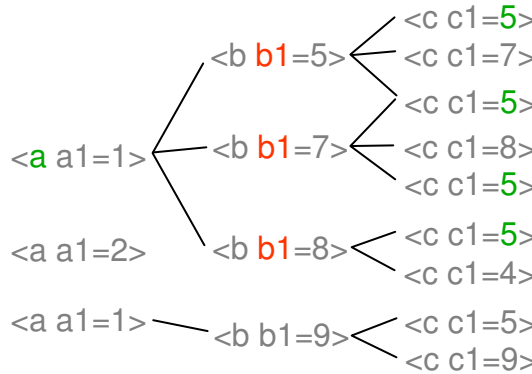
select **sum**(@b1) from p/q/a.b.c ⇒
 -- summarizing @b1, which are only in tree
 <b b1= >
 <b b1= >
 <b b1= >

if 'c' has no self-refering FK (i.e. cannot contain list), else <?res code=1 /?>

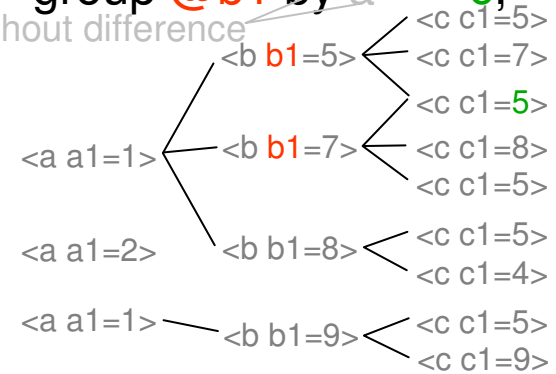
group @b1 by @a1 @c1;



group @b1 by a @c1;

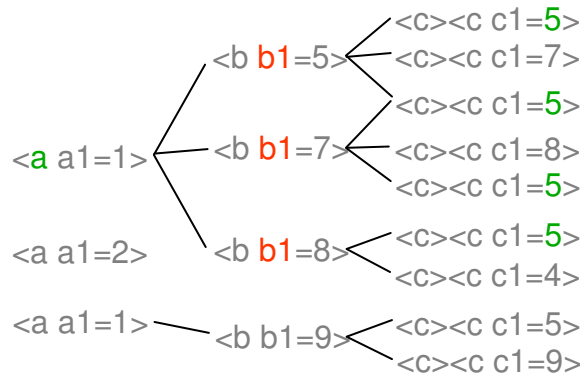
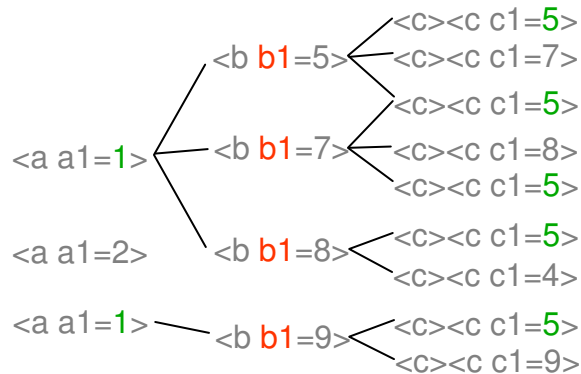


group @b1 by @a1 c;
 group @b1 by a c;
 without difference

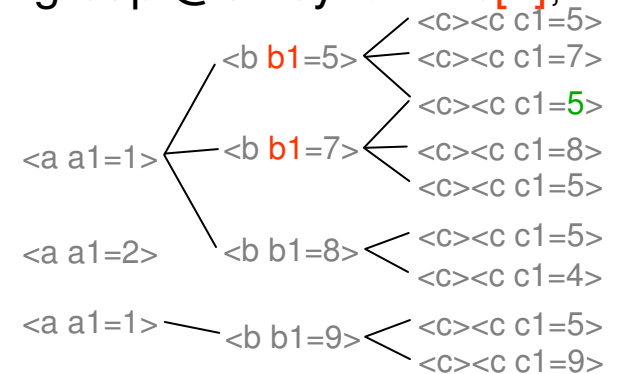


if 'c' has self-refering FK (i.e. can contain list)

group @b1 by @a1 c[2]/@c1; group @b1 by a c[2]/@c1;

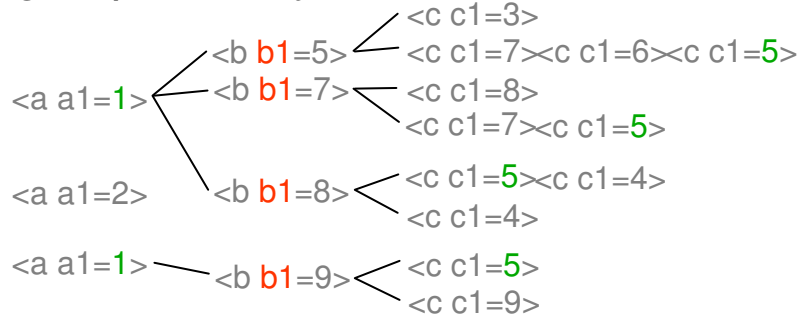


group @b1 by @a1 c[2];
 group @b1 by a c[2];



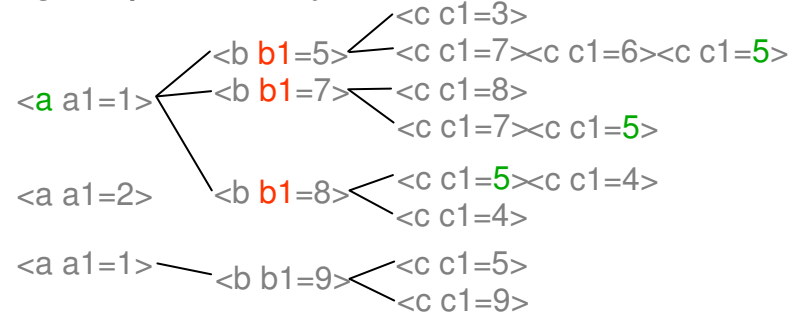
Group by list for extraction

group @b1 by @a1 some @c1;

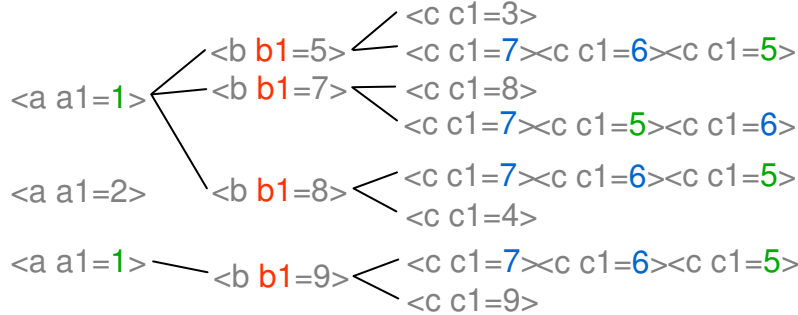


-- by any list element

group @b1 by a some @c1;

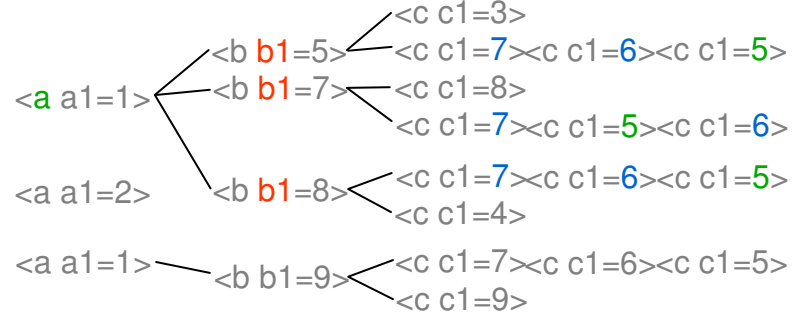


group @b1 by @a1 all @c1;

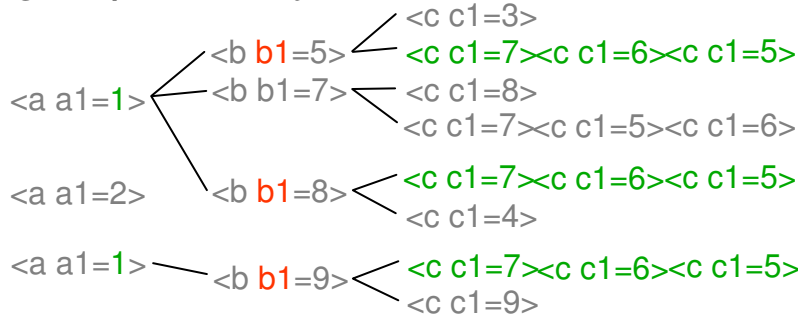


-- by all list elements

group @b1 by a all @c1;

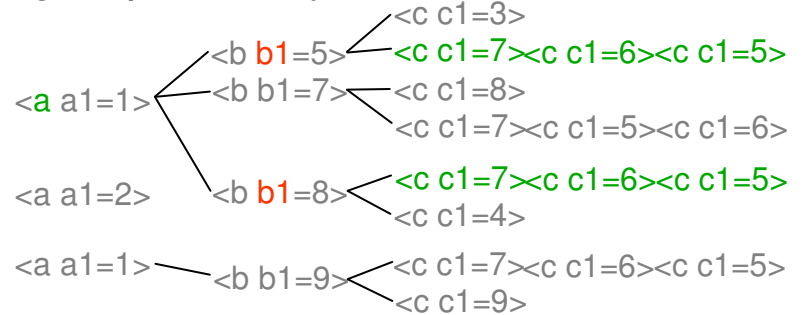


group @b1 by @a1 whole @c1;



-- by whole list

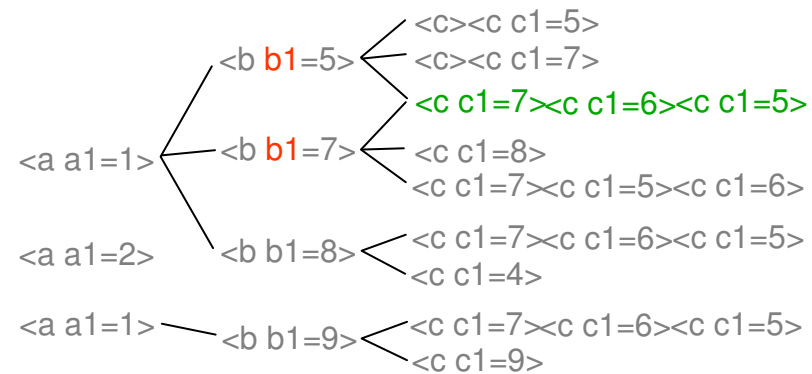
group @b1 by a whole @c1;



Group by list for extraction (continuation)

```
select sum(@b1) from p/q/a.b.c
```

```
group @b1 by a some c;  
group @b1 by a all c;  
group @b1 by a whole c;
```



Grouping for further operations

update a.b.c.d.e => <?res code=0 ?>

set @b1=sum(@d1)

-- assignment or comparison must obtain only one meaning from aggregate

group @d1 by b;

-- so group only by section of assignment or comparison (i.e. 'b'),

group @d1 by a;

-- or by any section, preceding it (i.e. 'a')

group @d1 by @b2; group @d1 by @a2;

-- or by their fields

set @b1=5

where sum(@d1)>10

for 'select', 'insert', 'delete' too

set @b1=sum(@c1)+sum(@d1)

-- left operand of arithmetical (e.g. '+') or comparison (e.g. '<>')
must correspond with only one right operand, and vice versa

group @c1, @d1 by b; group @c1, @d1 by a;
group @c1, @d1 by @b2; group @c1, @d1 by @a2;

-- so both aggregates in operation must have identical grouping

set @b1=5

where sum(@c1)<>sum(@d1)

as well as 'some(sum())', 'all(sum())'

set @b1=max(sum(@d1) as @x)

set @b1=5

where max(sum(@d1) as @x)>10

group @x by b, @d1 by e; group @x by a, @d1 by e;
group @x by @b2, @d1 by e; group @x by @a2, @d1 by e;

-- only first aggregate (i.e. 'max') has restriction above,
all next aggregates (i.e. 'sum') have no restrictions

Aggregate processes all sections with identical name

```
update a.b.c.b.c  
update a.b.b.c.c  
update a.(b.c)*    => <?res code=0 ?>  
update a.b*.c*
```

```
set @a1=sum(@c1)                                     set @a1=5  
-- aggregate always processes (e.g. summarizes)   where sum(@c1)>10  
all sections with identical name (i.e. all 'c' of different level '!')
```

```
group @c1 by @b;   group @c1 by @b1;  
-- grouping is always by first section with identical names (i.e. by 'b' with level !=2)
```

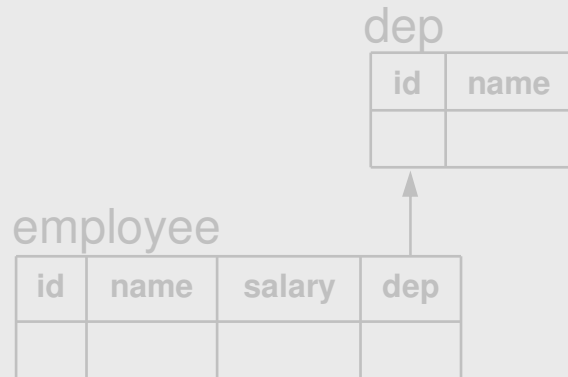
Default grouping

If aggregate is used, but 'group by' is absent,
then grouping is executed
by first section (record) in tree

```
select *
from city.(flight.city)*
where sum(flight/@price) = (select min(sum(@price) as @x)
                           from city.(flight.city)*
                           group @price, @x by city)
group @price city;
```

Wave algorithm (on examples)

Select employee



```
select *
  from dep.employee
 where @salary > (
    select avg(@salary)
    from employee e
    where e/@dep=dep/@id
  ) order by @salary desc;
```

Output:

```
<dep      id="1" name="Technical">
  <employee id="54" name="Fraud"  salary="3200"/>
  <employee id="72" name="Kitter"  salary="3100"/>
  <employee id="31" name="Tomson" salary="3000"/>
</dep>
<dep      id="2" name="Marketing">
  <employee id="25" name="Johnson" salary="4100"/>
  <employee id="64" name="Smith"   salary="4000"/>
</dep>
```

Select direct flights



select flight

from city[@name="a"].flight#c1:c2.city[@name="b"];

Output:

```
<flight t1="5.30" t2="8.30" day="sunday" price="12"/>
<flight t1="7.30" t2="10.30" day="monday" price="11"/>
<flight t1="9.30" t2="12.30" day="tuesday" price="10"/>
<flight t1="15.30" t2="18.30" day="wednesday" price="13"/>
<flight t1="17.30" t2="20.30" day="thursday" price="9"/>
<flight t1="19.30" t2="22.30" day="friday" price="11"/>
<flight t1="7.30" t2="10.30" day="saturday" price="10"/>
```

Select in-direct pathes

```
select (flight[@t1 @t2 @price].city)*.flight
from city[@name="a"].(flight#c1:c2[@day="monday"].city)*.flight.city
-- expression under 'select' clause must be substring of expression under 'from' clause
where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
-- (flight.city)*.flight.city is equal (flight.city)* for 'where' clause
and terminator(city)/@name="b";
```

*how to reach city "b"
from city "a" on Monday,
time for change flight is
not less 30 minutes*

Output:

```
<flight t1="5.30" t2="5.50" price="2">
  <city name="C">
    <flight t1="7.00" t2="11.00" price="7"/>
  </city>
</flight>
<flight t1="5.10" t2="12.10" price="11">
  ...
</flight>
```

Select most cheap path as tree

```
select city*
  from city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b"
     and sum(@price) =
      -- specifying 'flight' before '@price' is redundant, because other tables dont contain '@price'
select min(sum(@price))
  from city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b";
```

*what is the most cheap path
from city "a" to city "b",
time for change flight is
not less 30 minutes*

Output:

```
<city name="C">
  <city name="D">
    <city name="B">
      </city>
    </city>
```

Select most cheap path as list

```
select town
  from city town,
  city[@name="a"].(flight#c1:c2.town)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b"
     and sum(@price) =
select min(sum(@price))
  from city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b";
```

not satisfy

satisfies

-- to align tree to list

*what is the most cheap path
from city "a" to city "b",
time for change flight is
not less 30 minutes*

Output:

```
<town name="C"/>
<town name="D"/>
<town name="B"/>
```


Select cost of most cheap path

```
select sum(@price) as @cost
  from city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b"
     and sum(@price) =
select min(sum(@price))
  from city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b";
```

*what is cost of the most cheap path
from city "a" to city "b",
time for change flight is
not less 30 minutes*

Output:

<flight cost="8"/>

Select less bumpy path

```
select town
  from city town,
       city[@name="a"].(flight#c1:c2.town)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
       and terminator(city)/@name="b"
       and sum(@t2-@t1) =
select  min(sum(@t2-@t1))
  from  city[@name="a"].(flight#c1:c2.city)*
  where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
       and terminator(city)/@name="b";
```

*what is the less bumpy path
from city "a" to city "b",
time for change flight is
not less 30 minutes*

Output:

```
<town name="G"/>
<town name="H"/>
<town name="K"/>
<town name="B"/>
```

Select less boring path

```
select town
from city town,
     city[@name="a"].(flight#c1:c2.town)*
where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator(city)/@name="b"
     and sum(next(flight)/@t1 - previous(flight)/@t2) =
select min(sum(next(flight)/@t1 - previous(flight)/@t2)
from city[@name="a"].(flight#c1:c2.city)*
where parent(flight)/@t1 - child(flight)/@t2 > 0/00.30
     and terminator (city)/@name="b";
```

*what is the less boring path
from city "a" to city "b",
time for change flight is
not less 30 minutes*

Output:

```
<town name="J"/>
<town name="K"/>
<town name="B"/>
```

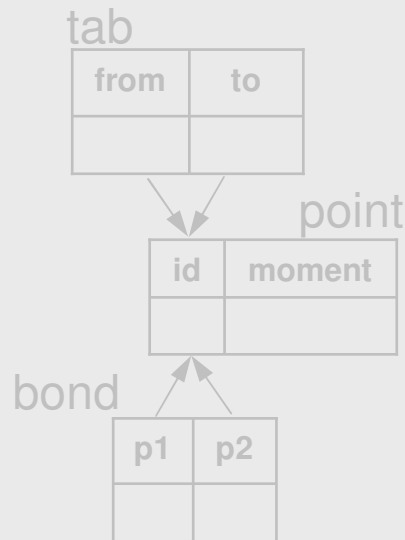
Visit all cities

```
select (flight[@t1 @t2 @price].city)*
from city[@name="a"].(flight#c1:c2.city[@name as @n])*
where root(flight)/@day="monday"
      and terminator(flight)/@day="friday"
      and @n~=("b", "c", "d", "a")
      and parent(flight)/@t1 –
          child(flight)/@t2 > 0/00.30
      and terminator(city)/@name="a"
      and terminator(city)/!=9
```

*from city "a" on Monday,
back into "a" on Friday,
through "b", "c", "d",
not more than one flight per day*

```
<flight t1="5.30" t2="5.50" price="2">
<city n="C">
<flight t1="7.00" t2="11.00" price="7">
<city n="D">
<flight t1="12.00" t2="13.20" price="4">
<city n="B">
<flight t1="14.00" t2="15.10" price="4">
<city n="A"/>
</flight>
</city>
</flight>
</city>
</flight>
</city>
</flight>
<flight t1="5.10" t2="12.10" price="11">
...
<city n="A"/>
...
</flight>
```

Find shortest path

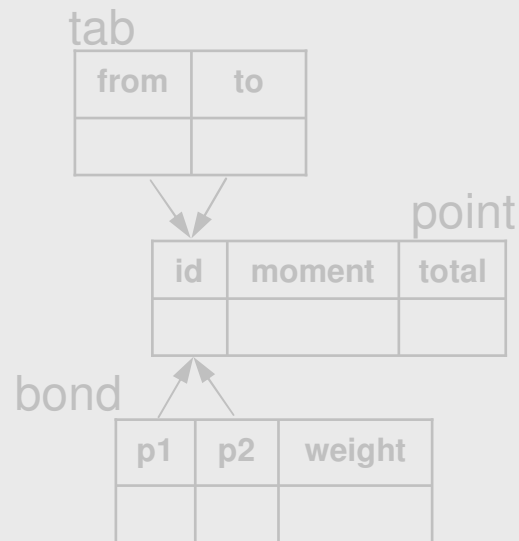


Output:

```
<point id="3">
<point id="4">
...
  <point id="11"/>
...
</point>
</point>
```

```
update point set @moment=null;
update tab#from.point.(bond#p1:p2.point[@moment=null])* .tab#to
  set @moment=!; -- wave of calculation runs from root to ends of branches
if tab#to.point/@moment is not null then
  select point.(point)*
    from tab#from.point.(bond#p1:p2.point[@moment=!])* .tab#to;
else select '<p>Pass does not exist</p>';
end if;
```

Find weighted shortest path



Output:

```
<point id="3">
<point id="4">
...
<point id="11"/>
...
</point>
</point>
```

```
update point set @moment=null, @total=null;
update tab#from.point.(bond#p1:p2.point[@moment=null or @total>../../@total+./@weight])* .tab#to
  set @moment=!, @total=../../@total+./@weight;
if tab#to.point/@moment is not null then
  select point.(point)*
  from tab#from.point.(bond#p1:p2.point[@moment=! and @total=../../@total+./@weight])* .tab#to;
else select '<p>Pass does not exist</p>';
end if;
```

Output, optionally

Title

comment on column a/@data is “title”;
comment on column b/@data is “caption”;
comment on column c/@data is “desc”;

```
select *  
from a.b.c
```

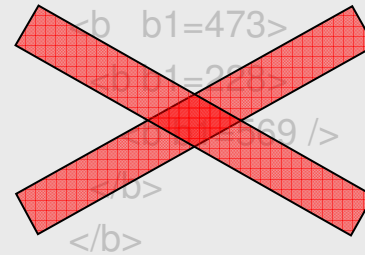
commenting a, c

```
-- extracts comments as first xml-element,  
-- appending service attribute _=comment
```

```
<a id=null data=title _=comment>  
<a id=1 data=12.3>  
  <b id=10 data=23.4>  
    <c id=null data=desc _=comment/>  
    <c id=100 data=56.7/>  
    <c id=101 data=67.8/>  
  </b>  
  <b id=20 data=34.5>  
    <c id=null data=desc _=comment/>  
    <c id=200 data=78.9/>  
    <c id=201 data=89.1/>  
  </b>  
</a>
```


Cartesian product without 'where'

```
select @b1 from p/q/a.b*.c;
```



```
<b b1=473 />  
<b b1=228 />  
<b b1=569 />
```

```
select @b1.@a1 from p/q/a.b.c;
```

-- only specified fields

```
<a b1=1 a1=v1 />  
<a b1=3 a1=v1 />
```

```
select @ from p/q/a.b.c;
```

-- all fields

```
<a a1=v1 a2=v2 b1=1 b2=2 c1=w1 c2=w2>  
<a a1=v1 a2=v2 b1=3 b2=4 c1=w3 c2=w4>
```

Selecting field as element

```
select ^b1.^d1
from p/q/a.b.c.d;
-- ^b1, ^d1 are fields
```

```
<b1>1</b1 >
  <d1>10</d1> <d1>30</d1>
<!-- </b> -->
<b1>3</b1>
  <d1>50</d1> <d1>70</d1>
<!-- </b> -->
```

```
select ^b1.<e>.^d1.</e>
from p/q/a.b.c.d;
```

```
<b1>1</b1 >
  <e> <d1>10</d1> <d1>30</d1> </e>
<b1>3</b1>
  <e> <d1>50</d1> <d1>70</d1> </e>
```

```
select ^b1.<e>||^d1||</e>
from p/q/a.b.c.d;
```

```
<b1>1</b1 >
  <e><d1>10</d1></e> <e><d1>30</d1></e>
<b1>3</b1>
  <e><d1>50</d1></e> <e><d1>70</d1></e>
```

View

```
create view vname as select a.b.c;
```

Concept for input

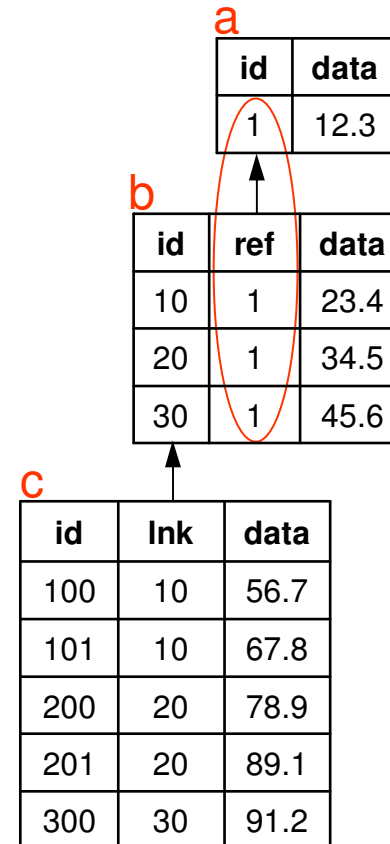
No adjustment

It's necessary to provide possibility to install DBMS and immediately use it, like user install and use programs "Teleport", "FlashGet", browser, etc. DBMS itself must accept XML, and place data from it into tables under some agreements:

- **xml-element** is saved in a table with *identical name* (i.e. tag name coincides with table name), **xml-attribute** is saved in field with *identical name* (i.e. attribute name coincides with field name). Primary key of new record is filled by trigger
- during creating two records, corresponding to **surrounding xml-element and directly enclosed** into it, primary key of one record *is copied into foreign key* of another record
- during creating two records, corresponding to **two consecutively located xml-elements** with the same names, *also* primary key of one record *is copied into foreign key* of another record

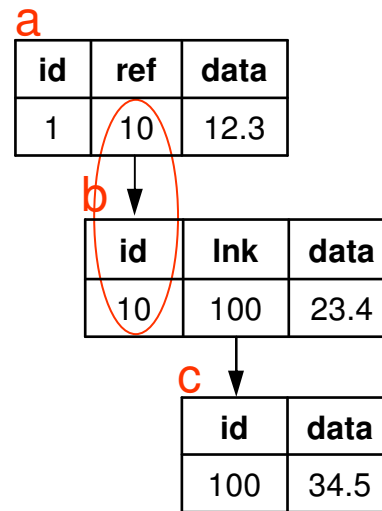
Auto-inserting as set

```
<a data=12.3>  
  <b data=23.4>  
    <c data=56.7/>  
    <c data=67.8/>  
  </b>  
  <b data=34.5>  
    <c data=78.9/>  
    <c data=89.1/>  
  </b>  
  <b data=45.6>  
    <c data=91.2/>  
  </b>  
</a>
```



Auto-inserting as relay-race

```
<a data=12.3>  
  <b data=23.4>  
    <c data=34.5/>  
  </b>  
</a>
```

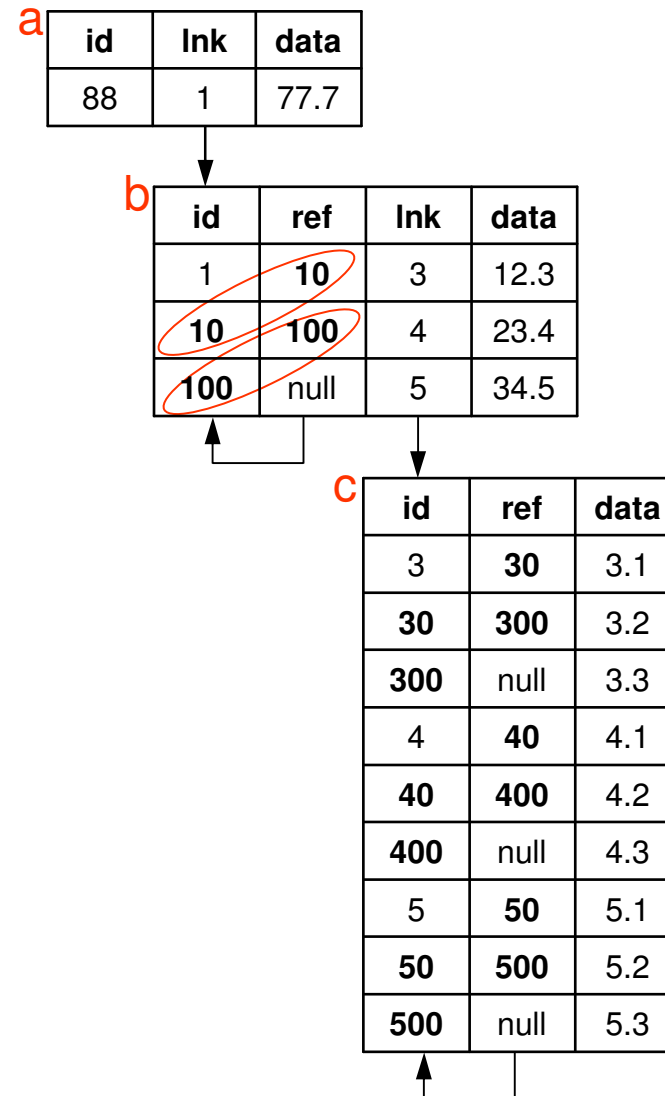


Auto-inserting as list

 <b data=12.3>
 <c data=3.1/>
 <c data=3.2/>
 <c data=3.3/>

 <b data=23.4>
 <c data=4.1/>
 <c data=4.2/>
 <c data=4.3/>

 <b data=34.5>
 <c data=5.1/>
 <c data=5.2/>
 <c data=5.3/>

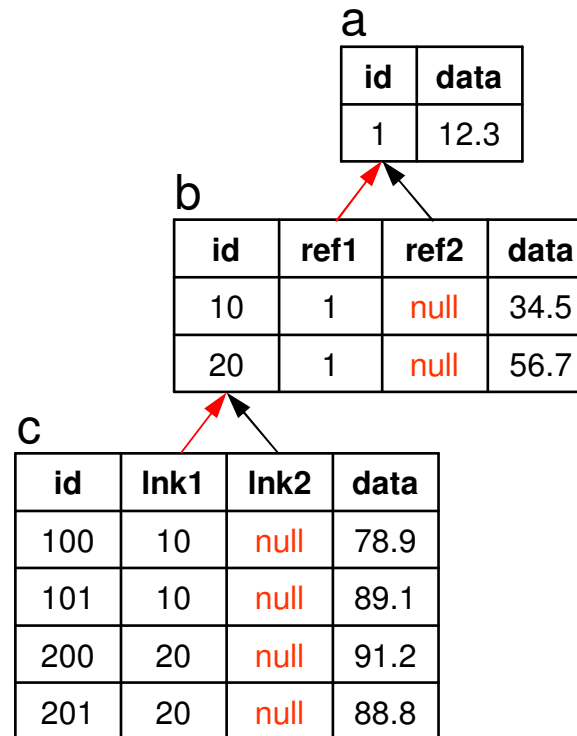


Determination

If two tables refer to each other, or one table refers to another by several foreign keys, than this ambiguity is solved in name of opening xml-tag: Sign '#' and name of necessary referring field (not name of constraint, i.e. name of foreign key) are specified after strictly name of table, containing necessary foreign key. It looks like new tag name with symbol '#' in the middle of the name. We shall name this specifying of referring field by term 'determination'.

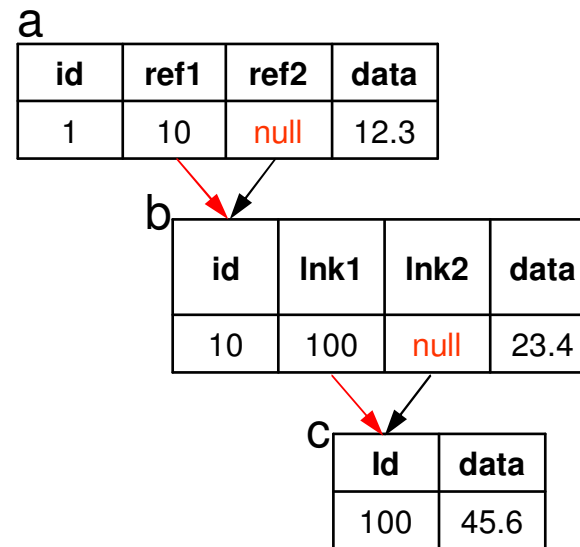
Determination for set

```
<a data=12.3>  
  <b#ref1 data=34.5>  
    <c#lnk1 data=78.9/>  
    <c#lnk1 data=89.1/>  
  </b>  
  <b#ref1 data=56.7>  
    <c#lnk1 data=91.2/>  
    <c#lnk1 data=88.8/>  
  </b>  
</a>
```



Determination for relay-race

<a#ref1 data=12.3>
<b#lnk1 data=23.4>
<c data=45.6/>



Determination too

If table refers to self by several foreign keys, than this ambiguity is also solved in name of opening xml-tag: Sign '\$' and name of necessary referring field (not name of constraint, i.e. name of foreign key) are specified after strictly name of table, containing necessary foreign key. It also looks like new tag name with symbol '\$' in the middle of the name. We shall name this specifying of referring field by term 'determination' too. Determination must be specified in each of consecutively located xml-elements with the same name - including in the first element (that user think less).

Different signs - '#' and '\$' - are used in two different types of determination, that it would possible to use both types of determination simultaneously: 'tag#field1\$field2'.

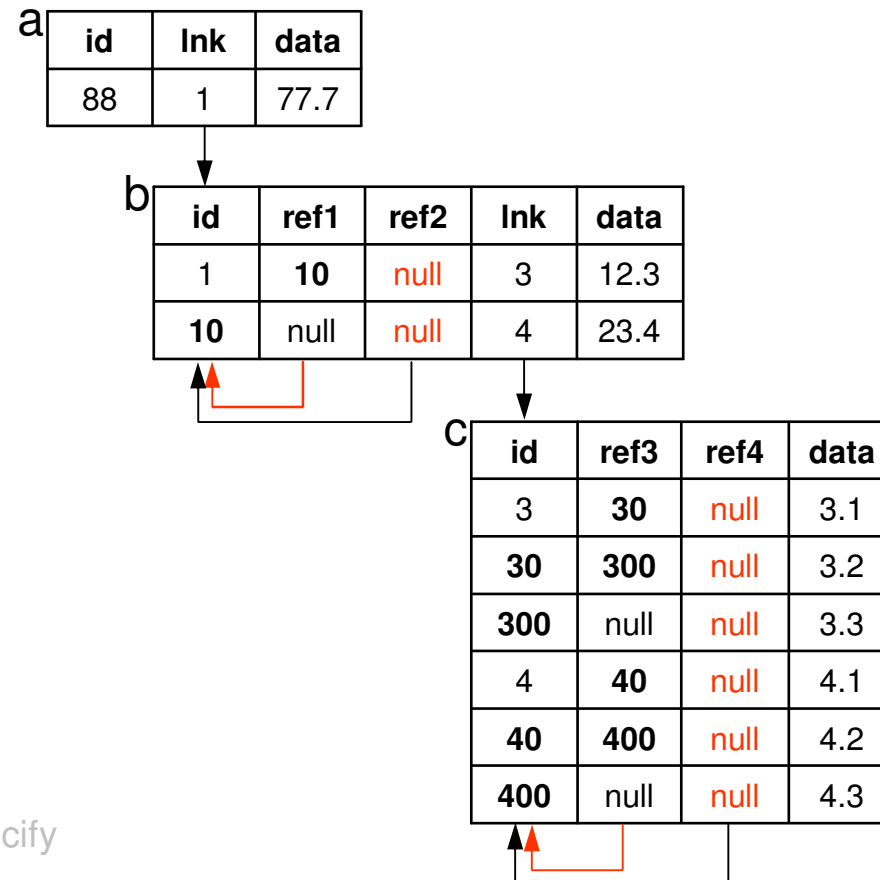
Determination for list

```

<a data=77.7>
  <b$ref1 data=12.3>
    <c$ref3 data=3.1/>
    <c$ref3 data=3.2/>
    <c$ref3 data=3.3/>
  </b>
  <b$ref1 data=23.4>
    <c$ref3 data=4.1/>
    <c$ref3 data=4.2/>
    <c$ref3 data=4.3/>
  </b>
</a>

```

- '\$' instead of '#', that it will possible to specify
- list and set simultaneously, as well as
- list and relay-race simultaneously



Presence of PK attribute means update instead of insert

a

pk	data
1	12.3

a

pk	data
1	45.7

Selection into LAN & WAN stops transaction

Client (e.g. browser) submits XML, DBMS enters it into tables, triggers react on it. When DBMS sends XML by '**select ... ;**' ('select ... union ... select ... ;'), client can use it as definition of new screen.

To all screens can be built by **one stored procedure** (instead of each next trigger for each next answer), extraction stops transaction, until client submits new XML or command '**prolong;**'.

Manually

Manually inserting as Set

insert into * values

```
<a data=12.3>  
  <b data=23.4/>  
  <b data=34.5/>  
  <b data=45.6/>  
</a>  
<a data=0.7>  
  <b data=56.7/>  
  <b data=67.8/>  
  <b data=78.9/>  
</a>
```

;

a

id	data
10	12.3
11	0.7

b

id	Ink	data
100	10	23.4
101	10	34.5
103	10	45.6
104	11	56.7
105	11	67.8
106	11	78.9

Manually inserting as Relay-race

insert into * values

```
<a data=12.3>  
  <b data=23.4/>  
</a>  
<a data=56.7>  
  <b data=78.9/>  
</a>
```

;

a

id	Ink	data
10	100	12.3
11	101	56.7

b ↓

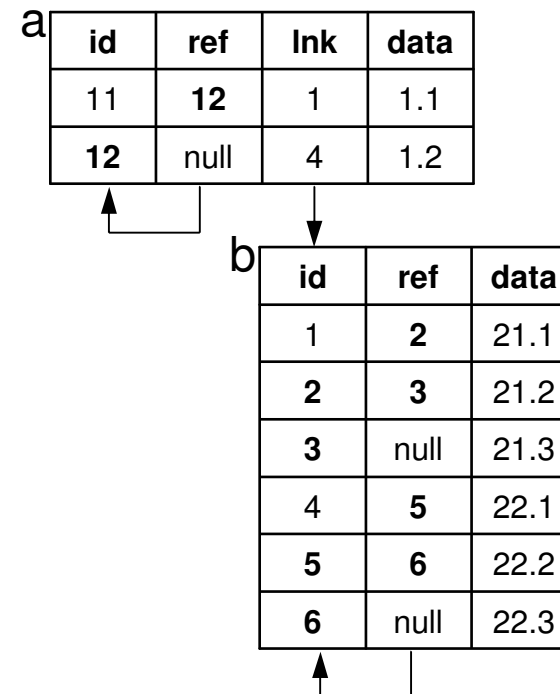
id	data
100	23.4
101	78.9

Manually inserting as List

insert into * values

```
<a data=1.1>  
  <b data=21.1/>  
  <b data=21.2/>  
  <b data=21.3/>  
</a>  
<a data=1.2>  
  <b data=22.1/>  
  <b data=22.2/>  
  <b data=22.3/>  
</a>
```

;



Manually: Determination

insert into * values

```
<a#ref data=12.3>  
  <b data=23.4/>  
  <b data=34.5/>  
  <b data=45.6/>  
</a>
```

;

insert into * values

```
<a data=12.3>  
  <b#lnk data=23.4/>  
  <b#lnk data=34.5/>  
  <b#lnk data=45.6/>  
</a>
```

;

Manually into content

Manually into content: Set

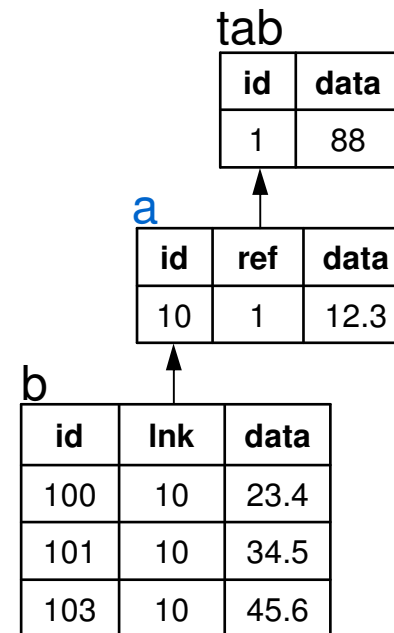
insert into tab (@data, @μ) values (88,

<b data=23.4/>

<b data=34.5/>

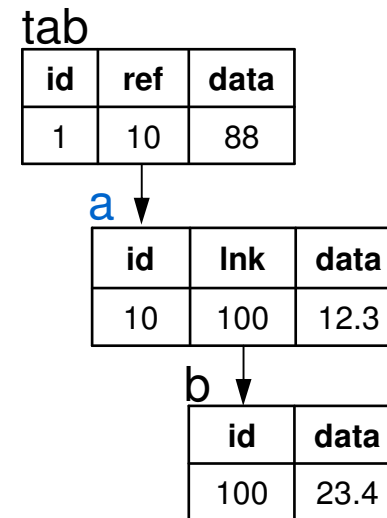
<b data=45.6/>

);



Manually into content: Relay-race

```
insert into tab (@data, @μ) values (88,  
  <a data=12.3>  
    <b data=23.4/>  
  </a>  
);
```



Manually into content: List

insert into tab (@data, @μ) values (88,

<b data=21.1/>

<b data=21.2/>

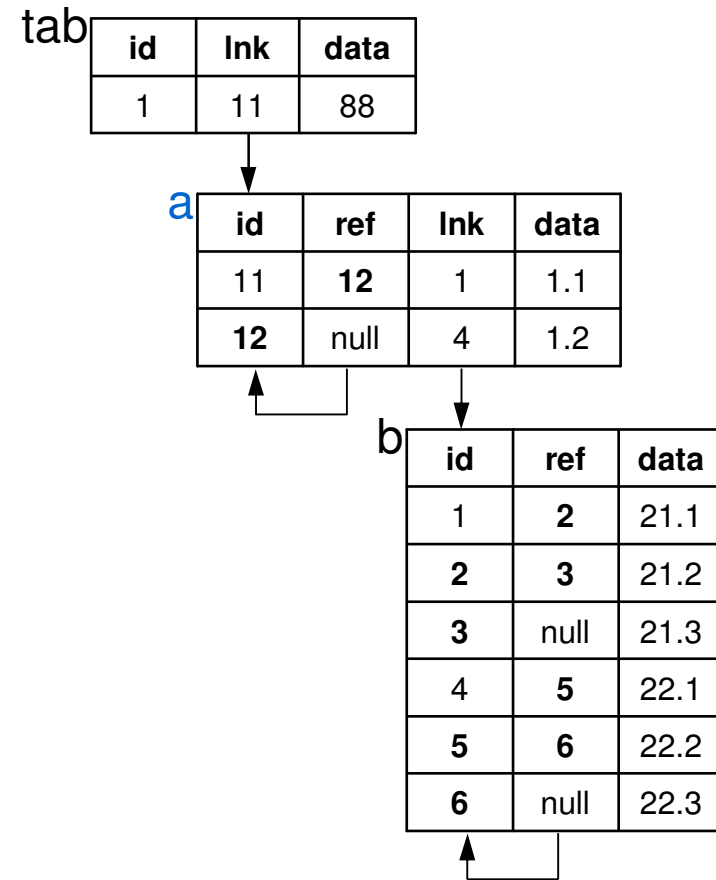
<b data=21.3/>

<b data=22.1/>

<b data=22.2/>

<b data=22.3/>

);



Set: determination

insert into tab (@μ) values (

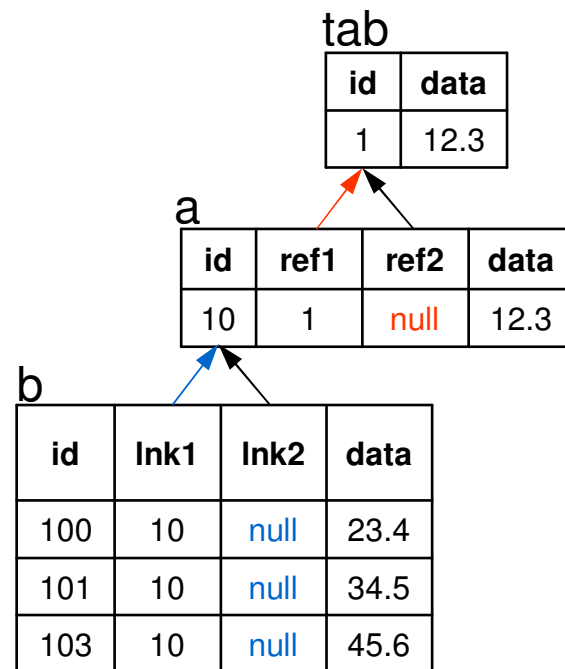
<a#ref1 data=12.3>

<b#lnk1 data=23.4/>

<b#lnk1 data=34.5/>

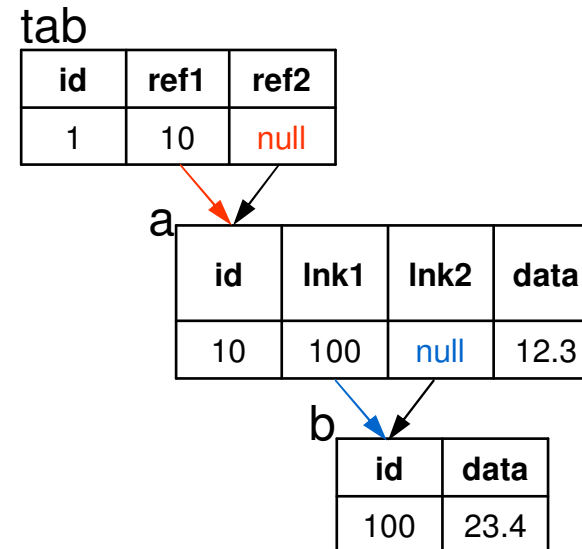
<b#lnk1 data=45.6/>

);



Relay-race: Determination

```
insert into tab#ref1 (@μ) values (  
  <a#lnk1 data=12.3>  
  <b data=23.4/>  
  </a>  
);
```



List: Determination

insert into tab (@μ) values (

<a\$ref1 data=1.1 >

<b\$ref3 data=21.1/>

<b\$ref3 data=21.2/>

<b\$ref3 data=21.3/>

<a\$ref1 data=1.2 >

<b\$ref3 data=22.1/>

<b\$ref3 data=22.2/>

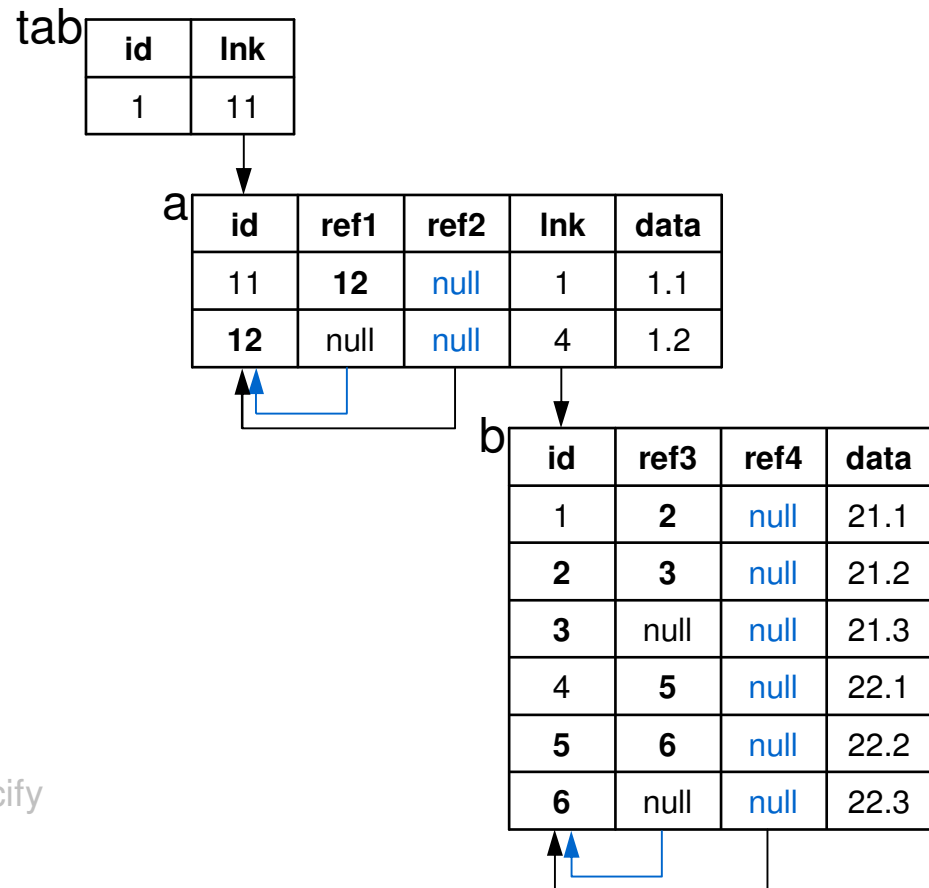
<b\$ref3 data=22.3/>

);

-- '\$' instead of '#', that it will possible to specify

-- list and set simultaneously,

-- list and relay-race simultaneously



Inserting into tree

Into set or relay-race

```
insert into p/q/a.b(@μ).c.d
```

```
-- if 'b' is list, inserting under first element of list 'b'
```

```
insert into p/q/a.b[3,5](@μ).c.d
```

```
-- inserting under 3th and 5th elements of list 'b'
```

```
...values <c c1=10><k k1=20/></c>;
```

```
...select * from c.k;
```

```
-- once more nested 'c.k' - in addition to existing 'c.d'
```

Into list

insert into p/q/a.b(@b1, @b2).c ...

insert into p/q/a.b(@).c ...

-- inserting after first element of list 'b'

insert into p/q/a.b[3,5](@b1, @b2).c[7,8] ...

insert into p/q/a.b[3,5](@).c[7,8] ...

-- inserting after 3th and 5th elements of list 'b' – only if list 'c' has 7th and 8th elements

... values (1, 2);

... select @b1, @b2, @c1, @c2 from b, c where @b1=@c1;

Separation

Select only inserted
by outside agent

```
select ... outside;
```

Permission depends of
activity of trigger

```
grant ... depends TriggerName;
```


Saved at break of connection

savewhole; -- by default

```
<a data=12.3>
  <b data=23.4>
    <c data=56.7/>
    <c data=67.8/>
  </b>
  <b data=34.5>
    <c data=78.9/>
    <c data=89.1/>
  </b>
  <b data=45.6>
    <c data=91.2/>
  </b>
</a>
```

-- nothing is saved
-- '' is necessary

savepart;

```
<a data=12.3>
  <b data=23.4>
    <c data=56.7/>
    <c data=67.8/>
  </b>
  <b data=34.5>
    <c data=78.9/>
    <c data=89.1/>
  </b>
  <b data=45.6>
    <c data=91.2/>
  </b>
</a>
```

-- 'a', two 'b', two 'c'
-- are saved

← breakdown →

Concept for processing

Partly deleting of tree

(referring fields obtain 'null')

delete **b, d** from p/q/a.**b.c.d**;

delete **b** from p/q/a.*****; -- in all places

delete **b.*** from p/q/a.**b.c.d**;

delete ***** from p/q/**a.b.c.d**;

delete **a.b.c.d** from p/q/*****;

delete **a.b*.c*.d** from p/q/*****;

Partly deleting of list

(referring fields obtain new values)

```
delete b[3,5].* from p/q/a.b.c;
```

```
delete b[3,5].* from p/q/a.b[11-18].c;
```

-- deleting 13th and 15th elements

Insertion of tree

insert into p/q/a.b(@μ).c ...

insert into p/q/a.b[3,5](t/u/@μ).c[7,8] ...

...values <n n1=10><k k1=20/></n>;

Replace operator 'merge'

by separate 'update'
and separate 'insert'

```
update a (@a1, @a2) select @a1, @a2 from k  
where @a3=@k3;
```

```
update a (@a1, @a2) select @k1 as @a1, @k2 as @a2 from k  
where @a3=@k3;
```

Replace 'merge' for list

`update` p/q/a.b(@b1, @b2).c(@c1, @c2) ...

`update` p/q/a.b(@).c(@) ...

`update` p/q/a.b[3,5](@b1, @b2).c[7,8](@c1, @c2) ...

`update` p/q/a.b[3,5](@).c[7,8](@) ...

... `select` @b1, @b2, @c1, @c2 from b, c where @b1=@c1;

Updating = deleting + insertion

```
update tab set @data=88, @μ = <a      data=12.3>...</a>;  
update tab set @data=88, @μ = <a#ref1 data=12.3>...</a>;  
update tab set @data=88, @μ = <a$ref1 data=12.3>...</a>;
```

```
-- delete all trees, nested into table "tab",  
-- insert specified tree(s) as nested into "tab"
```


Update several tables at once

```
update p/q/v.w*.z* set @w1=5, @z1=10;  
update p/q/v.w*.z* (@w1=5, @z1) select 5, 10;
```

Sub-tree like field

```
update p/q set a.* = <k k1=3><m m1=4></k>;  
update p/q set a.b.c = <k k1=3><m m1=4></k>;  
update p/q set a.* = (select * from k.m.n);  
update p/q set a.b.c = (select * from k.m.n);
```

'where' for tree

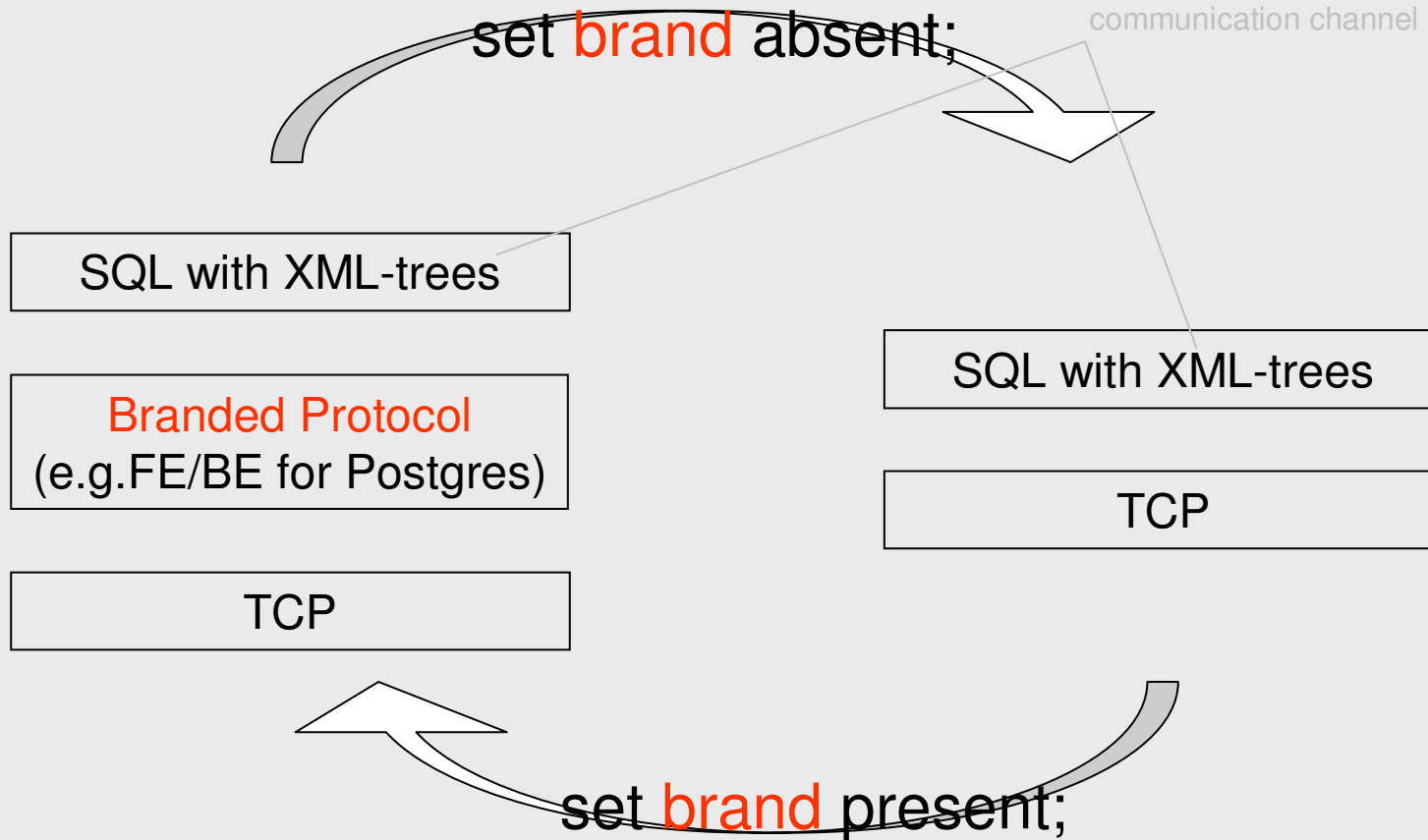
select * from p/q/a.b.c ...
insert into p/q (
q/n/k/a.b(@b1).c ...
update p/q/a.b.c ...
delete b, d from p/q/a.b.c.d ...

where q/t/u/@μ = <n n1=10><k k1=20/></n>
and b/t/u/@μ in <m m1=30><v v1=40/></m>

XML just over TCP,
optionally

Switching protocol stack

XML is mentioned instead of binary format only to represent from communication channel onto slides



Switching and enquiring session parameter

set brand present; -- to enter Branded Protocol

set brand absent; -- to withdraw Branded Protocol

show brand; \Longrightarrow <?res code=0 pb=absent /?>

Distributed query

Concept for dQ

SQL vs. branded programs

It's necessary to note, that SQL would be more flexible and convenient for distributed query (**gathering** data from several databases and **scattering** them into several databases), than branded programs; including more convenient for **replication**, than branded programs. But there is no necessary syntax:

- database has '**nickname**' (nickname is specified in query before table name through colon)
- group of databases is named '**society**' (society is also specified before table name through colon, and means nicknames of all databases of group)
- '**default**' database is database, in which all nicknames and societies are stored

Nickname, 'all', restriction

```
connect          address=remote.bz;  
create nick db1 address=site.com;  
create nick db2 address=data.net;  
create nick db3 address=store.org;  
create nick db4 address=place.ws;
```

default database

target database
≠

source database

```
insert into      tab select * from db1:tab;  
insert into db3:tab select * from db1:tab;  
insert into all:tab select * from db1:tab;  
insert into %all:tab select * from %db1:tab;  
insert into %default:tab select * from %all:tab;
```

The same or different

One sql-statement with society means a great number of sql-statements with nicknames. Besides this, that nicknames, several societies or several mentions of one society don't specify the same database simultaneously, we place symbol '%' before them; and that several mentions of one society always synchronously specify the same database, we place **any word (identical) and symbol '%'** before this mentions.

Society, 'all', restriction

```
create society s;  
create society c;  
append db1, c, default to s;  
refuse db2, c from s;
```

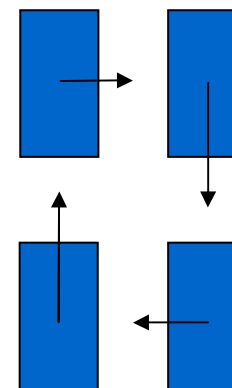
```
insert into          tab select * from s:tab;  
insert into db1:tab select * from s:tab;  
insert into (db1+db4):tab select * from s:tab;  
insert into s:tab select * from s:tab;  
insert into (s-c):tab select * from s:tab;  
insert into all:tab select * from s:tab;  
insert into %all:tab select * from %s:tab;  
insert into %c:tab select * from %s:tab;  
insert into %default:tab select * from %c:tab;
```

target database
≠
source database

Society as oriented graph

```
create society s;  
append db1 [ into db7 ], db2 [ into db8 ] to s;  
refuse db1 [ into db7 ], db2 [ into db8 ] from s;  
append (db1, db2) [ into (db7,db8) ] to s;  
refuse (db1, db2) [ into (db7,db8) ] from s;  
refuse all from s;
```

```
create society c;  
append db1 into db2, db2 into db3,  
db3 into db4, db4 into db1 to c;  
insert into c:tab select * from c:tab;
```



Control of societies

```
create society s, c;  
drop society s, c;  
drop society all; -- drop all societies  
set society none; -- forbid all societies  
set society s, c; -- allow two societies  
set society all; -- allow all societies  
set society all except s, c;  
  
grant append on society s to u;  
grant refuse on society s to u;  
revoke drop on society s to u;  
revoke set on society s to u;
```

Marker

```
insert into identical%all tab  
select * from %all:tab  
where @fld > (select @f from identical%all:tab);
```

different
societies

```
insert into m%s:tab  
select * from %s:tab  
where @fld > (select @f from m%s:tab);
```

always the same database

Prefix terms

prefix

restricted prefix

marked prefix

nick db

restricted nick %db

society s

restricted society %s

marked society i%s

all all

restricted all %all

marked all i%all

restricted default %default

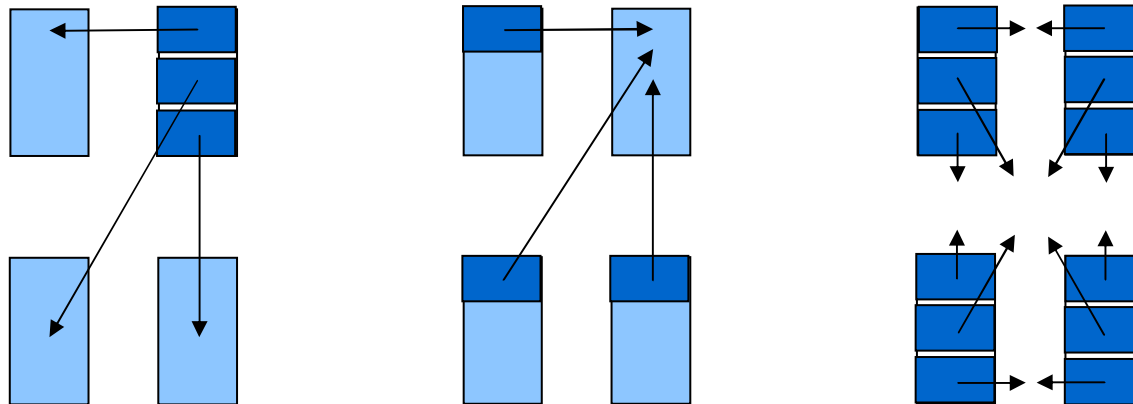
restricted you %you

Nick identifier (NID)

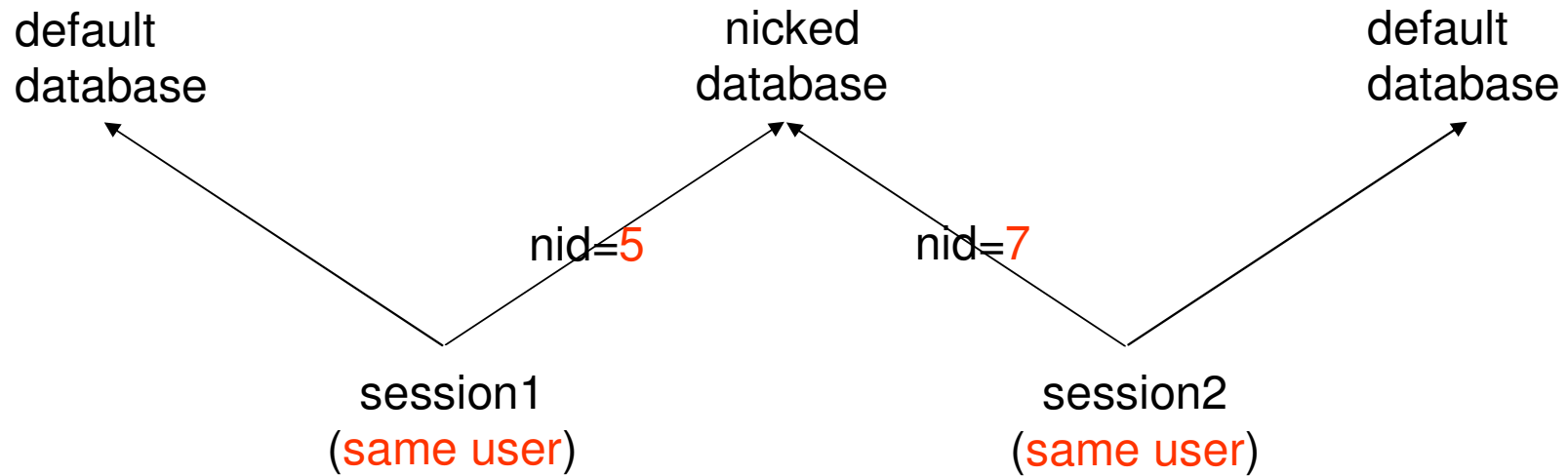
```
create nick [ db1 ] address=www.ws [ nid=7 ];  
alter  nick db1 [ address=site.com ] [ nid=1 ];  
drop  nick db1;  
drop  nick all; -- of one user
```

NID

```
insert into all:t (@d, @s) select @d, %% from db1:t where @x=all:%%;  
insert into db1:t (@d, @s) select @d, %% from all:t where @x=db1:%%;  
insert into i%all:t (@d, @s) select @d, %% from %all:t where @x=i%all:%%;
```



NID is session-specific



Crossing FK (cFK)

```
connect address=www.site.com user=Smith;  
create table b (  
  b1 xml,  
  b2 number,  
  b3 xml  
);
```

```
connect address=www.remote.bz user=Rothschild;  
create nick db1 address=site.com;  
create table a (  
  a1 number references db1:b(@b1/k/m/@m1),  
  a2 xml, foreign key (@a2/p/q/@q1) references db1:b(@b2),  
  a3 xml, foreign key (@a3/p/q/@q1) references db1:b(@b3/k/m/@m1)  
);
```

Implementation

Mistrust & simplicity

In purpose of security, distributed query must satisfy some requirements - concept of whole mistrust of one database to another:

- database does **not store login** of other database (not to give foreign login at unauthorized access)
- database does **not edit** (update, insert, delete) data of other database (temporary access to one database, obtained at unauthorized access, can't destroy other database)
- if it's possible, database does **not obtain** data from other database (data from another database does not become accessible at unauthorized access)

And concept of client simplicity:

- client does **not know SQL** (can't simplify or decompose SQL)

Isolation

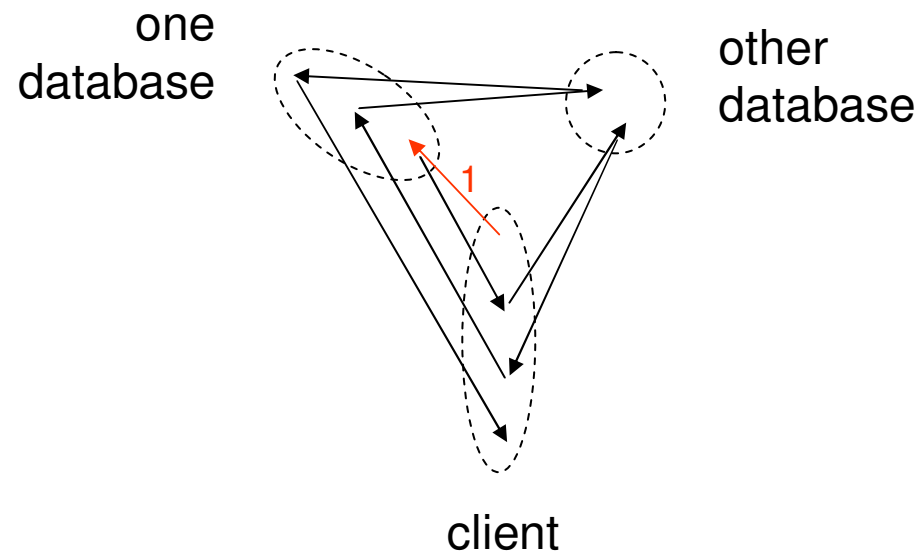
So one database can't create and enter sql-command into other database neither directly, nor indirectly (asking client to forward command). And client can't construct new sql-command on base of parsing of entered sql-command.

First database sends special command to client, which force client to make substitution inside last sql-command, sent to first database and memorized in client stack, and to send result of transformation into second database. Special commands must be so limited, that to not allow appearance of sql-command, harm for second database.

- `<?ry/?>`
- `<?finish/?>`
- `<?iiv/?>`
- `<?reader/?>`

Participation of user

User submits only first (1) query, all next parcels are sent by client and DBMS-es automatically, independently of user.



Select distributed union (dU)

1. select * from all:tab;

1. select * from s:tab;

1. select * from tab

union

select * from db1:tab;

3. connect addr=site.com;

select * from %you:tab;

connect addr=data.net;

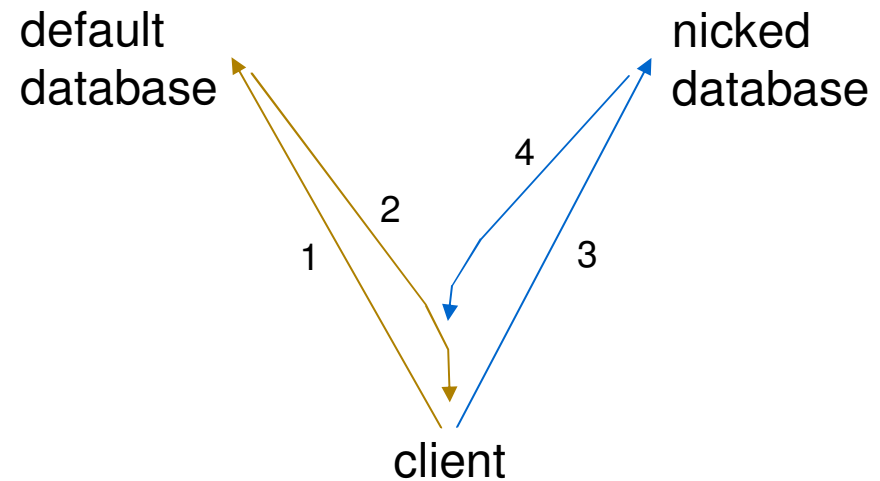
select * from %you:tab;

connect addr=store.org;

select * from %you:tab;

2. Bag
Baribute

```
<?ry addr=site.com /?>  
<?ry addr=data.net /?>  
<?ry addr=store.org /?>  
<tab id=3 data1=12.3 data2=aaa>  
<tab id=7 data1=23.4 data2=bbb>  
<tab id=10 data1=34.5 data2=ccc>  
<tab id=25 data1=45.6 data2=ddd>
```



Exactly '%you' is used in all similar cases, reasons are on the slides [155](#), [158](#)

Insert/update/delete dU

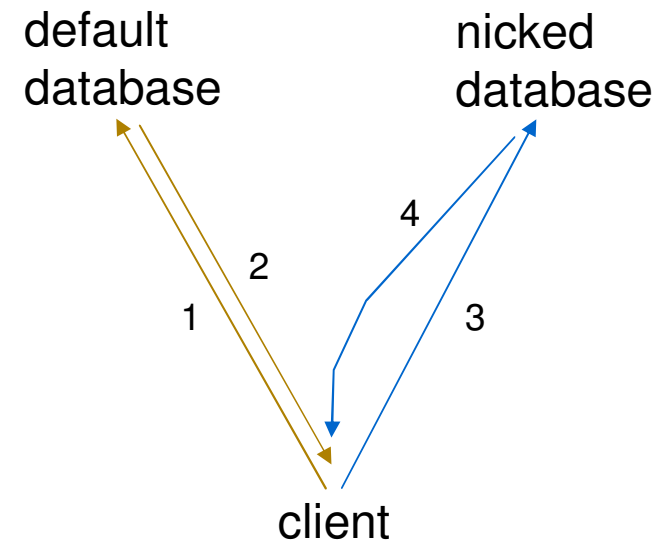
1. insert into **all**:tab values (3, 5);
insert into **s**:tab values (3, 5);

1. insert into **db1**:tab values (3, 5);

3. connect addr=site.com;
insert into **%you**:tab values (3,5);
connect addr=data.net;
insert into **%you**:tab values (3,5);
connect addr=store.org;
insert into **%you**:tab values (3,5);

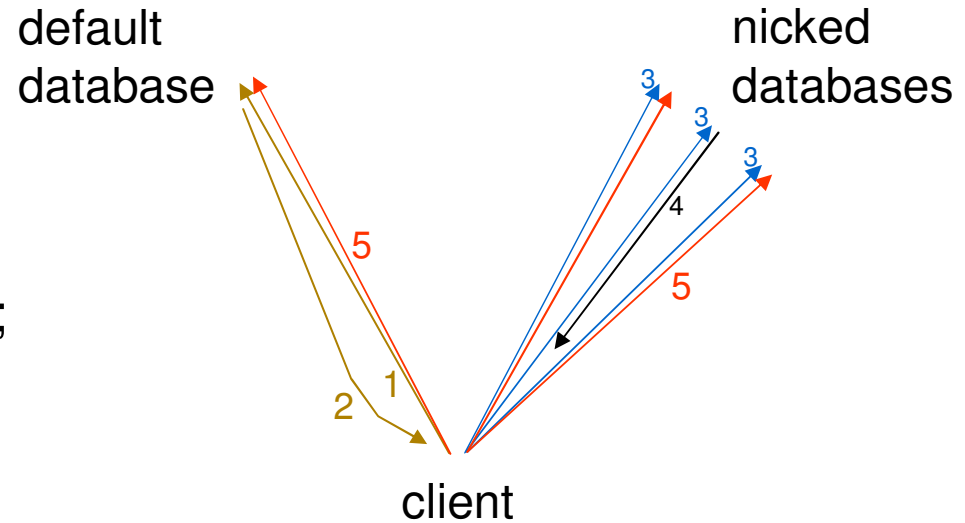
2. `<?ry addr=site.com /?>`
`<?ry addr=data.net /?>`
`<?ry addr=store.org /?>`

2. `<?ry addr=site.com /?>`

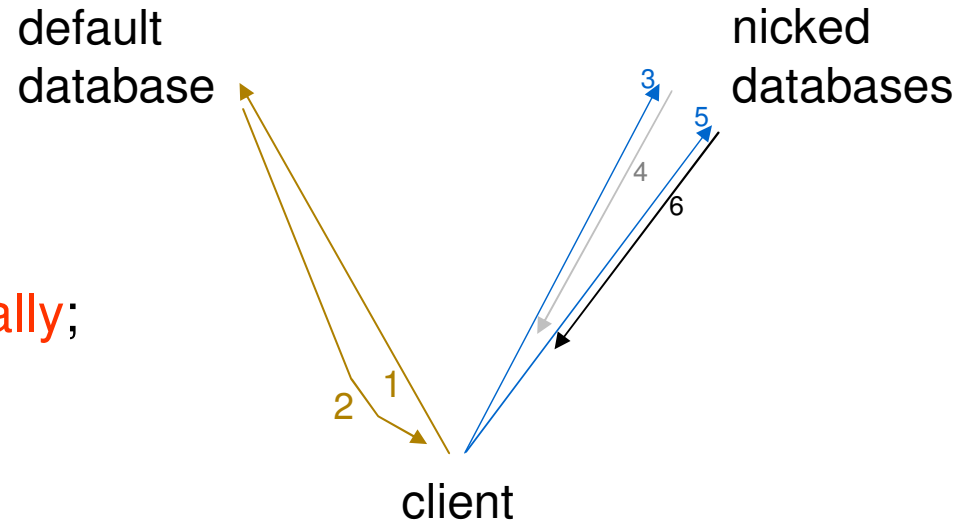


Search parallelly

1. select **all:func()** **parallelly**;
2. `<?ry addr=site.com /?>`
`<?ry addr=data.net /?>`
`<?ry addr=store.org /?>`
3. `connect addr=site.com;`
 `select %you:func();`
`connect addr=data.net;`
 `select %you:func();`
`connect addr=store.org;`
 `select %you:func();`
4. `<tab data1=12.3 data2=aaa>`
5. **cancel;**

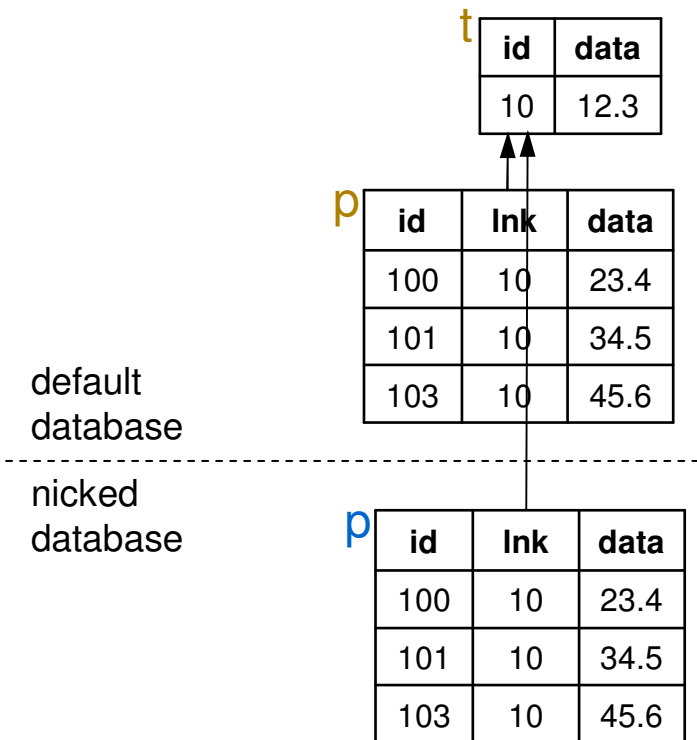


Search consecutively

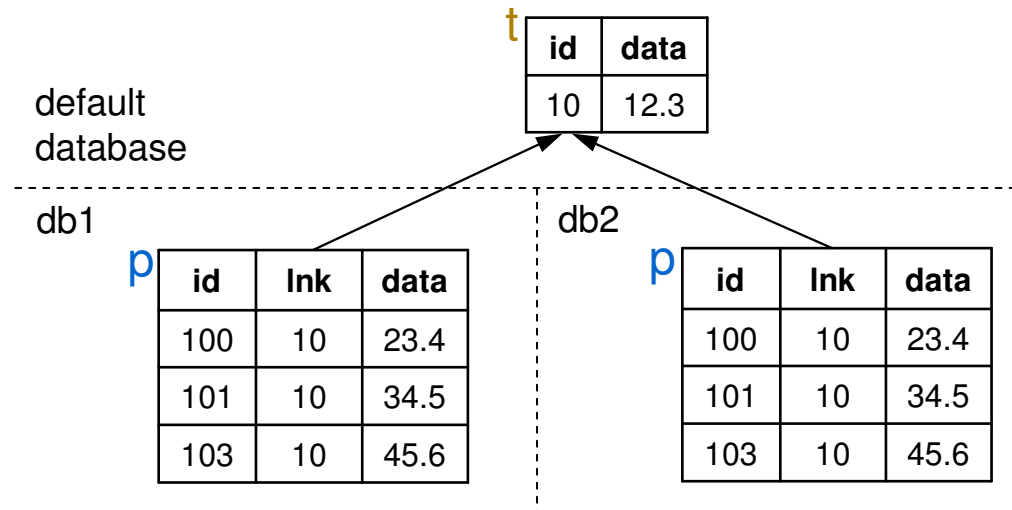


1. select **all:func()** **sequentially**;
2. `<?ry addr=site.com /?>`
`<?ry addr=data.net /?>`
`<?ry addr=store.org /?>`
3. connect addr=site.com;
select **%you:func()**;
4. `<?res code=2 /?>` -- absence
5. connect addr=data.net;
select **%you:func()**;
6. `<tab data1=12.3 data2=aaa>` -- store.org will be not used

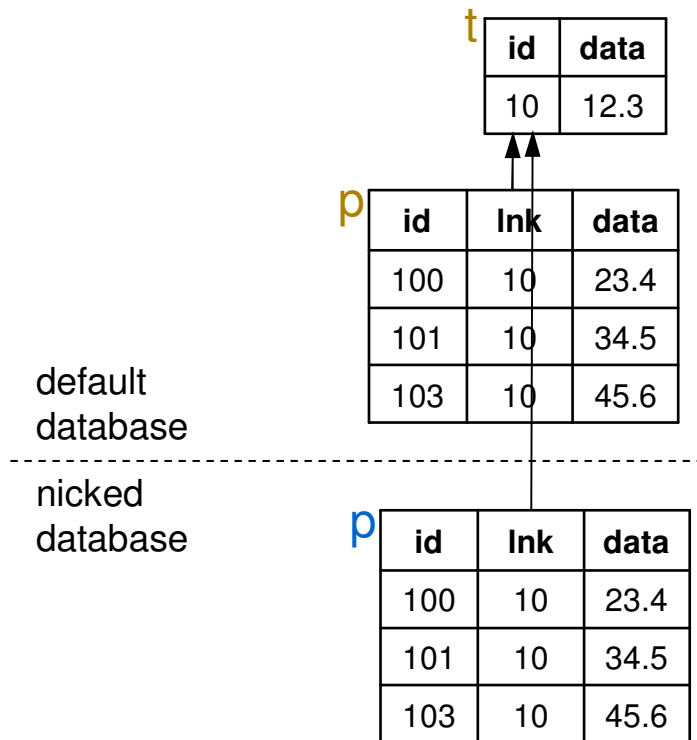
Ambiguity of cFK for Set



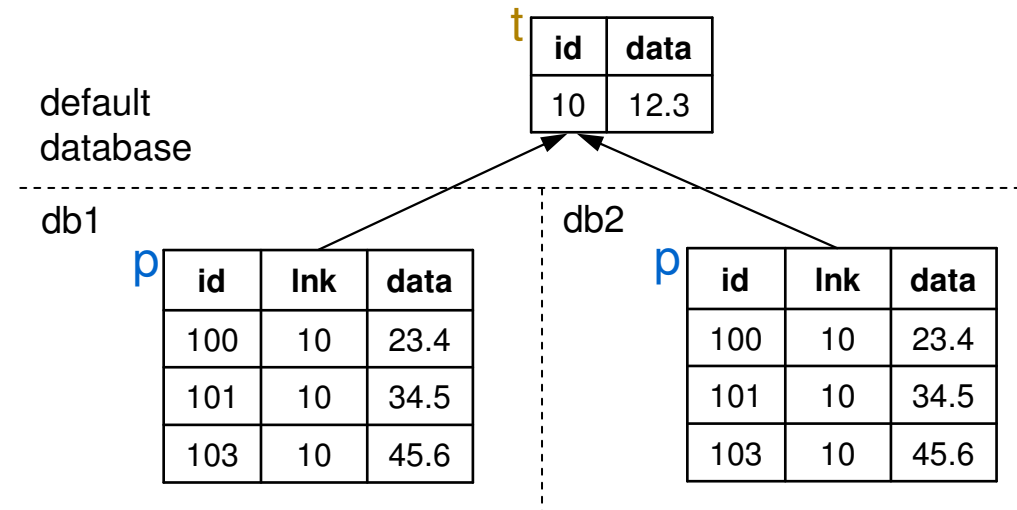
1. `select * from t.*;`
 1. `insert into t(@μ) values (<p><q/></p>);`
 1. `update t set p/q/r/@fld=5;`
 1. `delete * from t/p/q;`
2. `<?res code=3 /?> <!--mistake-->`



Constraint also does not work

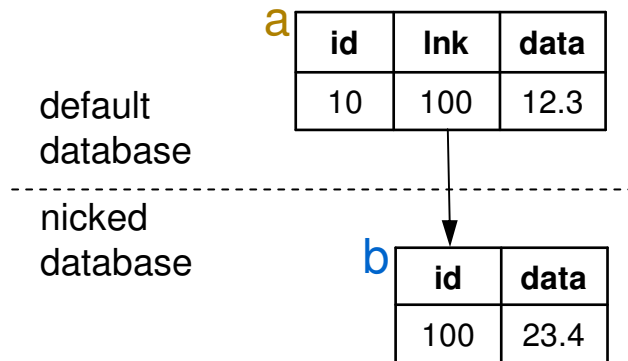


```
1. create table db1:p (
    id num,
    Ink num references default:t(id)
        on delete cascade,
        -- on update cascade
        -- on update set null
    data num
);
```



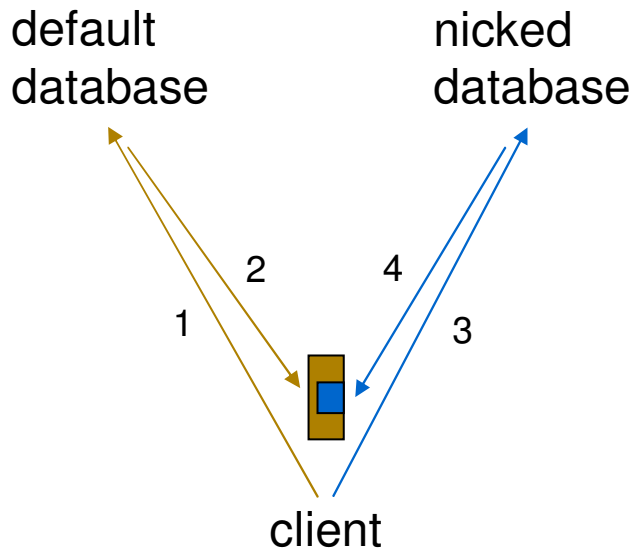
2. `<?res code=3 /?>`
`<!--mistake-->`

Select cFK of Relay-race & BTT



1. `select * from a.*;`

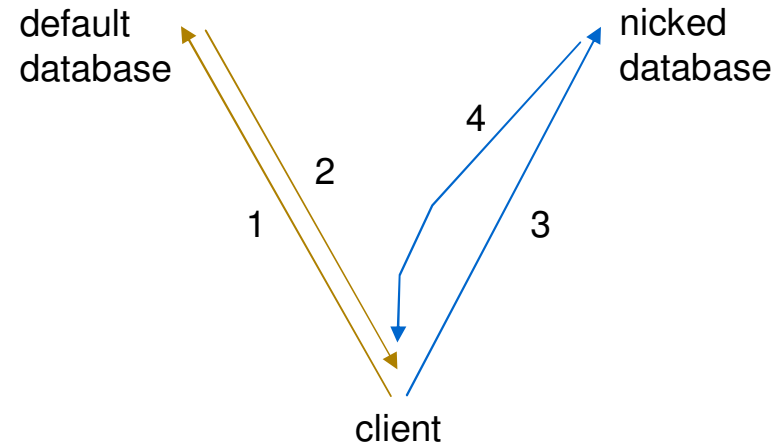
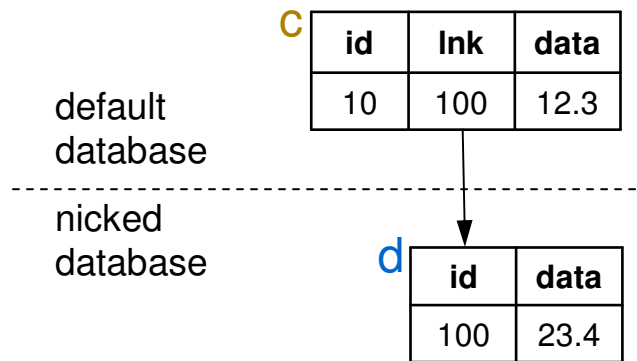
2. `<?finish addr=site.com
ques="select * from b.* where @b2=" /?>
<a a1=3 a2=23.4> <?f b2=453 /?>
<a a1=7 a2=23.4> <?f b2=748 /?>
<a a1=10 a2=34.5> <?f b2=295 /?>
<a a1=25 a2=45.6> <?f b2=817 /?> `



3. `connect addr=site.com;`

`select * from b.* where @b2=453;`
`select * from b.* where @b2=748;`
`select * from b.* where @b2=295;`
`select * from b.* where @b2=817;`

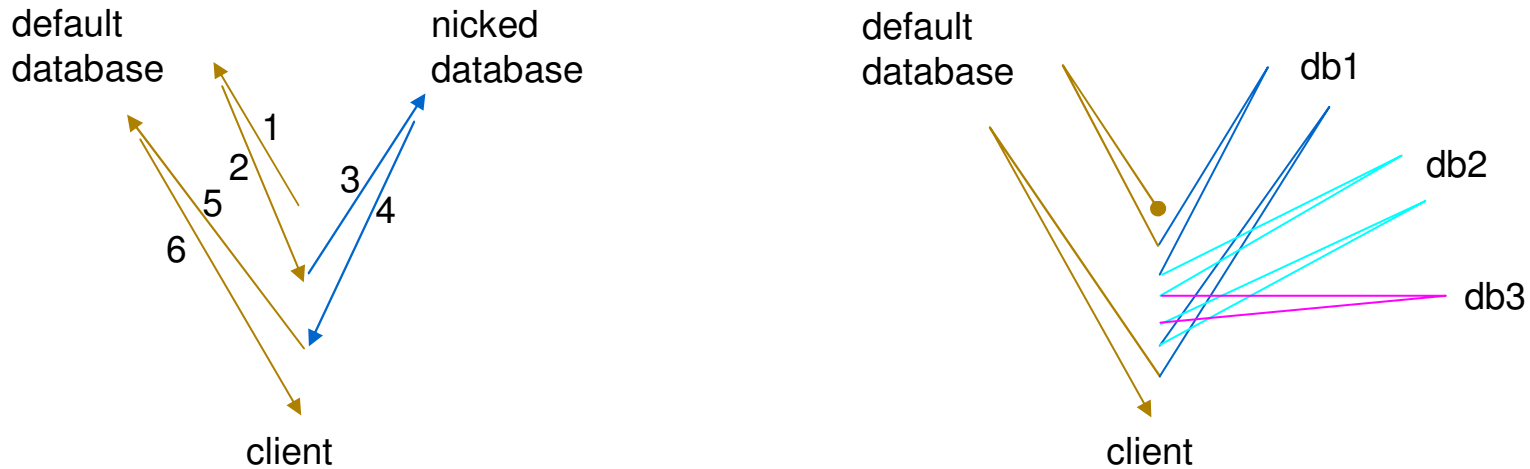
Sel/ins/upd/del via cFK:Relay-race



- | | | |
|--|--|------------------------|
| 1. select | <code>%db1:b/c/d/e</code> | where @b1=3 and @d1=5; |
| 1. insert into | <code>%db1:b/c/d/e</code> values (10,20) | where @b1=3 and @d1=5; |
| 1. update | <code>%db1:a</code> set b/c/d/e/@e1=5 | where @b1=3 and @d1=5; |
| 1. delete * | from <code>%db1:b/c/d/e</code> | where @b1=3 and @d1=5; |
| 2. <code><?ry shift="b/c" addr=site.com /?></code> | | |
| 3. select | <code>%you:d/e</code> | where @b1=3 and @d1=5; |
| 3. insert into | <code>%you:d/e</code> values (10,20) | where @b1=3 and @d1=5; |
| 3. update | d set e/@e1=5 | where @b1=3 and @d1=5; |
| 3. delete * | from d/e | where @b1=3 and @d1=5; |

desirable { 1. ... where b/@b1=3 and @d1=5;
3. ... where @d1=5;

Insert Relay-race: fractal



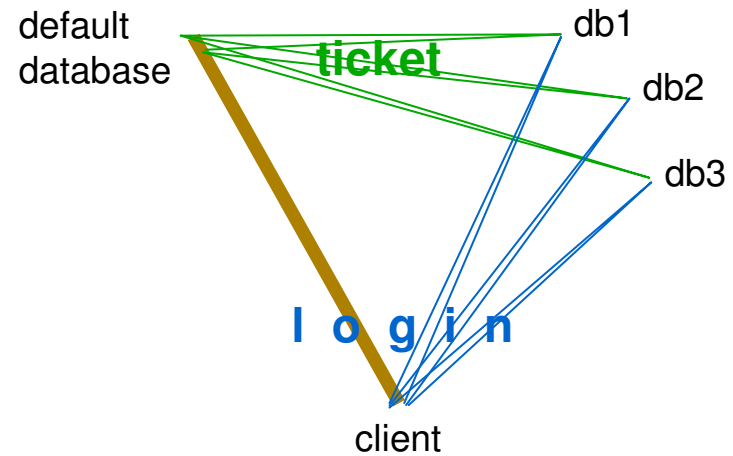
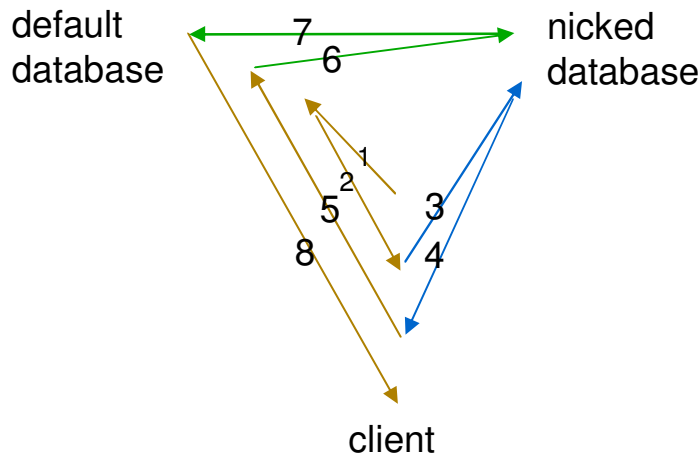
1. insert into a values <a> <c><d></d></c> ;
 2. <?iiv place="4" addr=nicked.com /?>
 3. insert into b values <c><d></d></c>;
 4. <?res code=0 pk=100 /?>
 5. insert into a values <a><? b 100 /?>;
 6. <?res code=0 pk=10 /?>
- <!-- OK -->
-- from stack
<!-- OK -->

Insert Relay-race: neighbour



1. insert into a values ⁴<a> ¹¹ ¹⁸<c></c> <d></d> ;
2. <?iiv place="4" addr=site.com /?>
<?iiv place="11" addr=data.net /?>
<?iiv place="18" addr=store.org /?>
3. <u>insert into b values ;
<u>insert into c values <c></c>;
<u>insert into d values <d></d>;
4. <?res code=0 pk=100/?> <?res code=0 pk=200/?> <?res code=0 pk=300/?>
5. insert into a values <a><? b 100 /?><? c 200 /?><? d 300 /?>;
6. <?res code=0 pk=10 /?>

Select distributed joining (dJ)



1. `select * from all:tab as ta, all:tab as tb where ta/@t1 = tb/@t2;`
2. `<?reader addr=default.edu ?>`
`<?store nick=db1 addr=nicked.com /?>`
`<?/reader?>`
3. `ticket default.edu`
`template select * from all:tab as ta, all:tab as tb where ta/@t1 = tb/@t2;`
4. `<?res code=0 ticket=xbwid ?>`
5. `upon db1=xbwid`
`select * from all:tab as ta, all:tab as tb where ta/@t1 = tb/@t2;`
6. `select ...` `-- particular 'select'`
7. `<tab t1=88 t2=99 data=12.3>` `<!-- particular data -->`
8. `<ta data1=3 data2=12.3>` `<!-- final data -->`

Ticket

Ticket is temporary **Username/Password**, which

- only for *'select'* operations
- only for concrete **address** of reader
- has *timeout* to connect
- *auto-removed* after disconnect (disposable)
- *disconnected and auto-removed* at first *'select'*, not derived from **template**
- under *privileges* of asking login (some records will be invisible like for asking login – [slide 185](#))

```
ticket reader.com  
template select *  
  from all:tab as ta, all:tab as tb  
  where ta/@t1= tb/@t2;
```

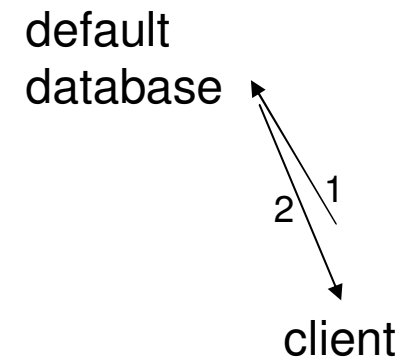
```
<?res code=0 ticket=xbwid /?>
```

Insert/update/delete dJ

1. insert into x values (1, 2, 3);

2. ~~<?dup view="@a1, @b1, @c1 from db1:a, db2:b, db3:c"?>
<?to nick=db1 addr=www.com?>
<?to nick=db2 addr=data.net?>
<?to nick=db3 addr=store.org?>
<?/dup?>~~

2. <?res code=4 /?> <!--mistake-->



Insertion, updating, deleting of Distributed Joining are **forbidden**, because default DBMS must tell definition of view and can **cheat** about names of **tables and fields**, and condition of joining

Select distributed intersection or exception (dI, dE)

Like select dJ:

1. `select * from tab intersect select select * from db1:tab;`
2. `<?reader addr=default.edu /?>`
`<?store nick=db1 addr=nicked.com /?>`
`<?/reader?>`
3. `ticket default.edu`
`template select * from tab intersect select select * from db1:tab;`
4. `<?res code=0 ticket=xbwid /?>`
5. `upon db1=xbwid`
`select * from tab intersect select select * from db1:tab;`
6. `select ...` `-- particular 'select'`
7. `<tab t1=88 t2=99 data=12.3>` `<!-- particular data -->`
8. `<ta data1=3 data2=12.3>` `<!-- final data -->`

Communication service info

select/insert/update/delete

<?ry shift="b/c" addr=more.com /?>
<?res code=2 /?>

select

<?cancel /?>
<?finish addr=more.com ques="select ..." /?>
<a> <?f b2=453 ?>
<?reader addr=default.edu ?>
<?store nick=db1 addr=nicked.com /?>
<?/reader?>

Bag "more"

insert

<?iiv place="5" addr=nicked.com /?>

insert/update/delete

<?res code=0 pk=10 /?>
<?res code=3 /?>

insert ... values (
 '... <? tab 100 ?> ...'
);

Kag "more"

<?res code=0 pk=10 /?>

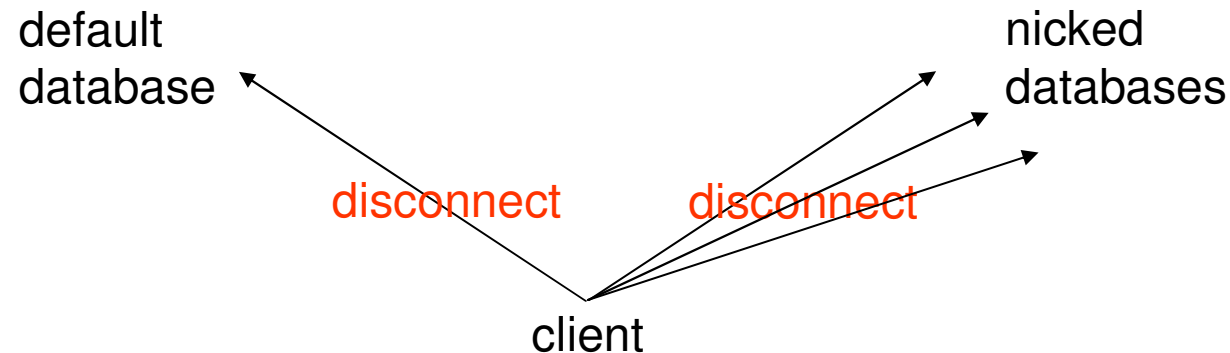
Baribute

Bag

Terms for XML and its analogue

	in document	in traffic
<element>	Element Tag Attribute	Sament (SAG eleMENT) Sag (Sent tAG) Saribute (SAG attRIBUTE)
<?element/?>	Kament Kag Karibute	Bament Bag Baribute

Disconnect default database
implies disconnect
all nicked databases



Sub-array

```
insert into i%all:t ( @fld )
```

```
  select sum( @arr[ i%all:%% ] ) from %all:tab;
```

```
select @arr[1] from tab; -- particular 'select' from db1:
```

```
select @arr[1] from tab; -- particular 'select' from db2:
```

```
select @arr[1] from tab; -- particular 'select' from db3:
```

```
  <tab arr[1]=5 />
```

```
  <!-- particular data from db1: -->
```

```
  <tab arr[1]=7 />
```

```
  <!-- particular data from db2: -->
```

```
  <tab arr[1]=8 />
```

```
  <!-- particular data from db3: -->
```

```
select @arr[ i=2:6:2, 2:i:2 ] from t;
```

```
<t arr[ i=2:6:2, 2:i:2 ]="1 2 3 4 5 6" />
```

```
select @arr[ 2:6:2, 2:6:2 ] from t;
```

```
<t arr[ 2:6:2,2:6:2 ]="1 2 3 0 4 5 0 0 6" />
```

```
select @arr[ (1,6,4),(2,3) ] from t;
```

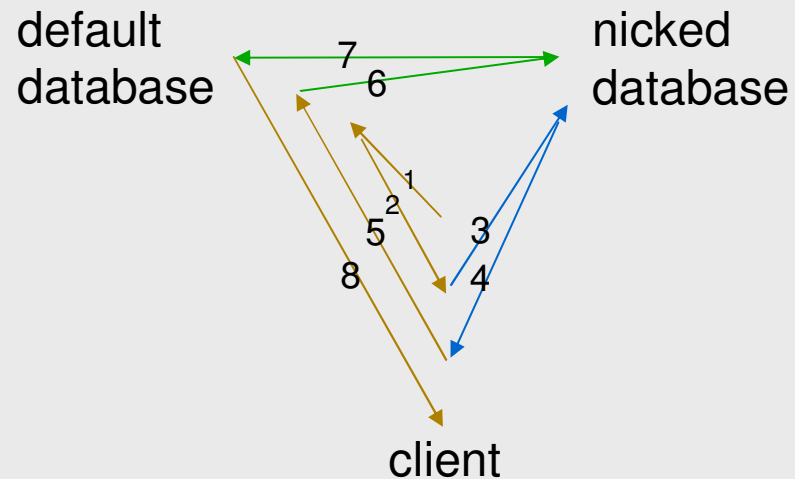
```
<t arr[ (1,6,4),(2,3) ]="0 3 2 0 0 0" />
```

```
select @arr[ 1 , : ] from t;
```

```
<t arr[ 1 , : ]="0 0 0 0 0 0" />
```

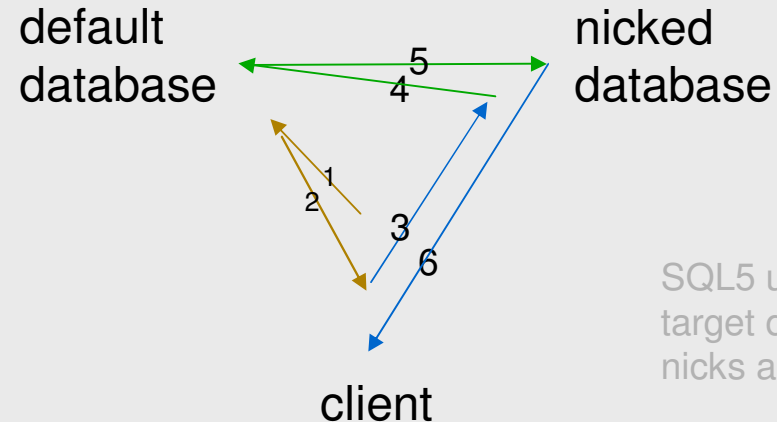

Examples of replication

Insert-select-1



1. insert into tab select * from db1:tab;
2. `<?reader addr=default.edu ?>`
`<?store nick=db1 addr=nicked.com ?>`
`<?/reader?>`
3. `ticket default.edu`
`template insert into tab select * from db1:tab;`
4. `<?res code=0 ticket=xbwid ?>`
5. `upon db1=xbwid`
`insert into tab select * from db1:tab;`
6. `select ...`
7. `<tab t1=88 t2=99 data=12.3>`
8. `<?res code=0 done=15 ?>`

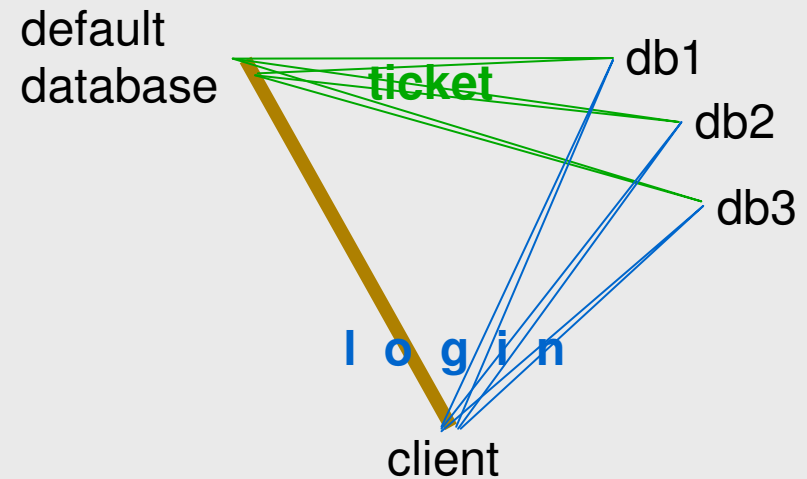
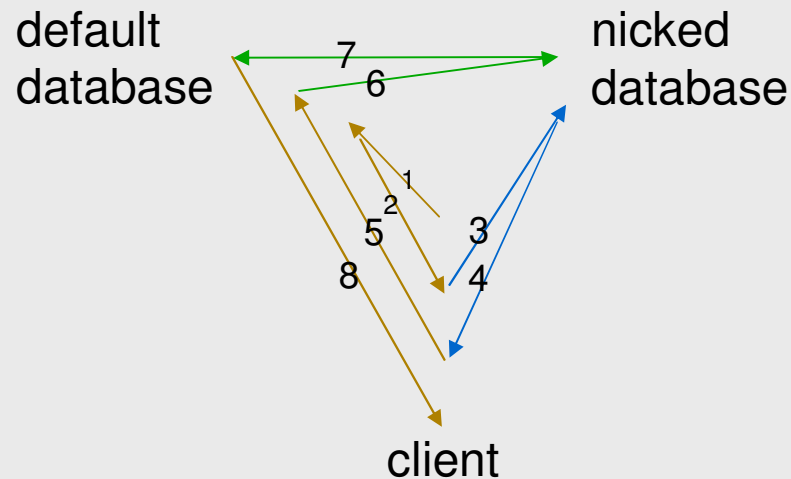
Insert-select-2: reason of 'YOU'



SQL5 uses prefix 'YOU' to say target database to not use own nicks and societies

1. insert into db1:tab select * from tab;
2. <?res code=0 ticket=xbwid ?>
<?ry addr=nicked.com ?>
3. upon default=xbwid
insert into %you:tab select * from tab;
4. select ...
5. <tab t1=88 t2=99 data=12.3>
6. <?res code=0 done=15 ?>

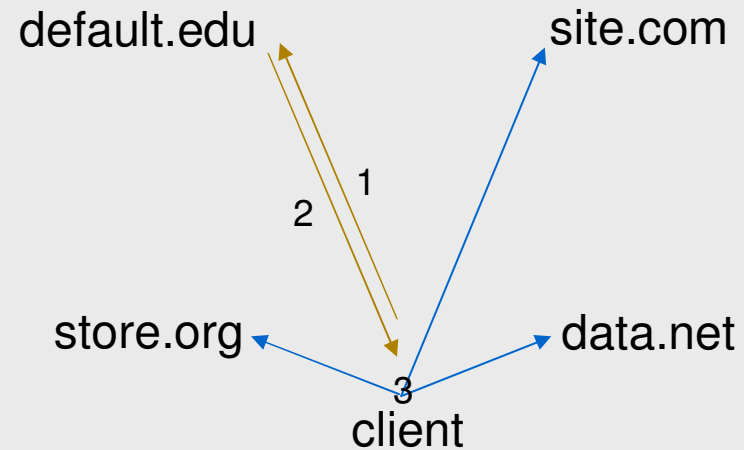
Update-select



1. `update tab set @fld= (select @a1 from db1:a);`
2. `<?reader addr=default.edu ?>`
`<?store nick=db1 addr=nicked.com ?>`
`<?/reader?>`
3. `ticket default.edu`
`template update tab set @fld= (select @a1 from db1:a);`
4. `<?res code=0 ticket=xbwid ?>`
5. `upon db=xbwid`
`update tab set @fld= (select @a1 from db1:a);`
6. `select ...` -- particular 'select'
7. `<a a1=88 >` <!-- particular data -->
8. `<?res code=0 done=15 ?>`

All-to-all and its traffic

1. insert into %all:tab select * from %all:tab;
2. <?reader addr=default.edu ?>
 <?store nick=db1 addr=site.com ?>
 <?store nick=db2 addr=data.net ?>
 <?store nick=db3 addr=store.org ?>
 <?/reader?>
 <?reader addr=site.com ?>
 <?ticket it=ccc ?>
 <?store nick=db2 addr=data.net ?>
 <?store nick=db3 addr=store.org ?>
 <?/reader?>
 <?reader addr=data.net ?>
 <?store nick=db1 addr=site.com ?>
 <?ticket it=nnn ?>
 <?store nick=db3 addr=store.org ?>
 <?/reader?>
 <?reader addr=store.org ?>
 <?store nick=db1 addr=site.com ?>
 <?store nick=db2 addr=data.net ?>
 <?ticket it=ooo ?>
 <?/reader?>

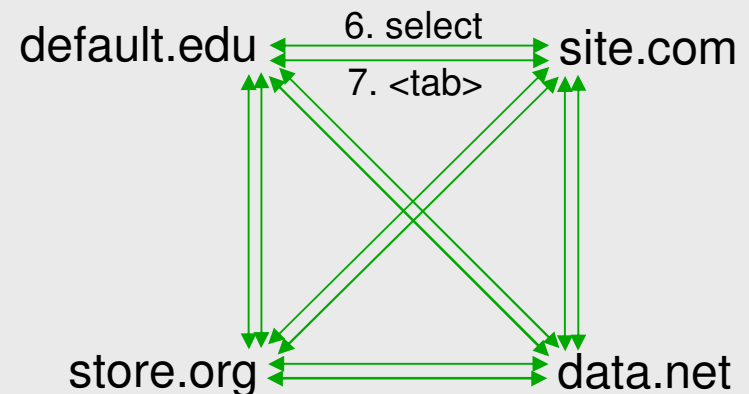
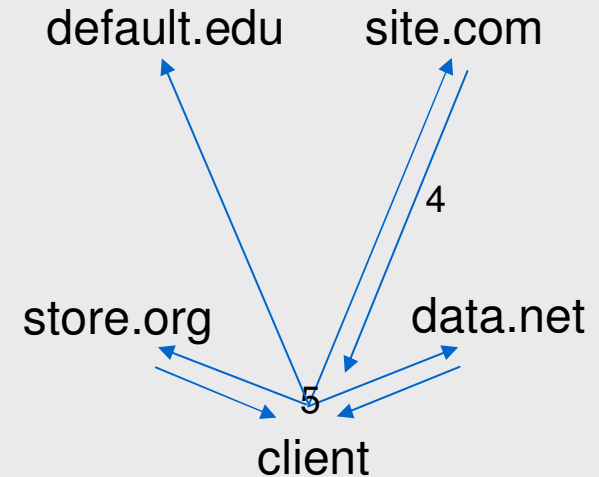


3. connect data.net;
 ticket site.com
 template insert into %all:tab
 select * from %all:tab;
connect store.org;
 ticket site.com
 template insert into %all:tab
 select * from %all:tab;

Reason of restriction of 'YOU'

4. `<?res ticket=222 ?><?res ticket=333 ?>`
...
5. `upon db1= and db2= and db3=
insert into %you:tab select * from %all:tab;
upon default=ccc and db2=222 and db3=333
insert into %you:tab select * from %all:tab;
upon db1= and default=nnn and db3=
insert into %you:tab select * from %all:tab;
upon db1= and db2= and default=ooo
insert into %you:tab select * from %all:tab;`
6. `select ... ; select ... ; select ... ;`
...
7. `<tab t1= t2= data= >
<tab t1= t2= data= >
<tab t1= t2= data= >`
...

SQL5 uses RESTRICTED prefix 'YOU'
to exclude target database from source
database-s



NID: whole <?ry?>, <?reader?>

```
select @fld, %% from all:a/b/tab;  
insert into all:a/b/tab (@d, @s) values (10, %%);  
update all:a/b/tab set @s=%%;  
delete * from all:a/b where @s=%%;
```

```
<?ry shift=b/c addr=site.com nid=1 ?>  
<?ry shift=b/c addr=data.net nid=2 ?>  
<?ry shift=b/c addr=store.org nid=3 ?>
```

```
insert into i%all:tab (d,s) select d,%% from s%all:tab where i%all:%%=s%all:%% +1;
```

```
<?reader addr=reader.com ?>  
  <?store nick=db addr=library.com ?> <!-- library.com to reader.com -->  
  <?ticket it=ccc ?> <!-- default.com to reader.com -->  
  <?sys nick=db1 addr=site.com nid=1 ?>  
  <?sys nick=db2 addr=data.net nid=2 ?>  
  <?sys nick=db3 addr=store.org nid=3 ?>  
<?/reader?>
```

dQ, optionally

Forward insertion & updating from default database into nicked databases

```
create predicate a/x/y/@y1=s:%% [and ...or ...]  
[for UserName]  
[parallely | consecutively]; -- only for select
```

```
-- only first proper predicate will be executed
```

DBMS automatically repeats query at break of connection

```
timein;  
insert into a  
  select *  
  from db1:a.*
```

```
timeout;
```

```
repeat  
  insert into a  
  select *  
  from db1:a.*  
  where db1:a/@pk<>a/@pk  
until TIMEOUT='false';
```

DBMS continues shapshot at break of connection

```
insert into db1:* select * from * -- snapshot  
controlling a; -- 'a' is snapshot mark
```

```
insert into db1:* select * from * -- after break of connection  
continuing a; -- submit tail of snapshot 'a'
```

Manually into nicked database

insert into s:* values

<b data=23.4/>

<b data=34.5/>

<b data=45.6/>

<b data=56.7/>

<b data=67.8/>

<b data=78.9/>

;

BLOB

Downloading BLOB

Connection can be broken, computer with DBMS can be rebooted or switched off (e.g. notebook). So it is necessary to save in field:

- **address of BLOB** on remote computer to continue
- **downloaded parts** of BLOB
- **offsets** of beginning and of end for each part to continue

And we need operators, returning:

- **BLOB**
- **percents of completeness** of downloading
- **address of BLOB** on remote computer (to inform, downloading is not finished, and where it is possible to obtain BLOB, if DBMS will be destroyed)
- **starting downloading** in background mode

Accepting client submission

If anybody selects BLOB during accepting, what must DBMS respond? Answering NULL, it notes, that it knows nothing about BLOB, what is false, because accepting is only matter of time. Thus field must save:

- **accepted parts** of BLOB
- meaning "**ACCEPTING**", similar to NULL, but not equal to it

Obtaining "ACCEPTING" during replication, may third database save it in own BLOB-field? No, because it disinforms users of third database, that it itself is accepting BLOB. Thus once more meaning, similar to NULL, into which "ACCEPTING" is turned during replication.

- meaning "**NOTGOT**", similar to NULL, but not equal to it

It is still a file

To choose routine to process BLOB, which does not contain filetype in own content, we need to save filetype in BLOB-field. If filetype is in the content, and DBMS understands this format, than must copy filetype from content into BLOB-field at each operation of coping BLOB

- file type

BLOB-field contains

- meaning "NULL"
 - meaning "ACCEPTING"
 - file type
 - beginning of BLOB
 - file type
 - whole BLOB
 - meaning "NOTGOT"
 - address on remote computer
 - file type
 - parts of BLOB
 - offsets of borders of parts

Operators

SELECT fld ...

to extract BLOB

SELECT **PERCENT**(fld) ...

completeness of downloading

SELECT **TYPE**(fld) ...

to extract file type of BLOB

SELECT **DOWNLOAD**(fld) ...

in background mode, returns boolean

SELECT **STOP**(fld) FROM * WHERE fld=542

to stop downloading BLOB

BLOBs are after all attributes

d1	d2
4	3652435
5	accepting
7	data.net/7254843
8	null

BLOBs are transferred not in xml-attributes, but after all xml-elements. Otherwise they would block reaction of acceptor at fast transferred nonBLOB attributes

select @d1, @d2 from d;

<d d1=4 d2="3652435"/>
<d d1=5 d2="accepting"/>
<d d1=7 d2="data.net/7254843"/>
<d d1=8 d2="null"/>
<?file id="3652435" type="mpg" size="3">Y29</?file>

If transfer of BLOB is broken

d

d1	d2
5	3652
7	accepting
8	data.net/7254843
9	null

```
select @d1, @d2  
from db1:d;
```

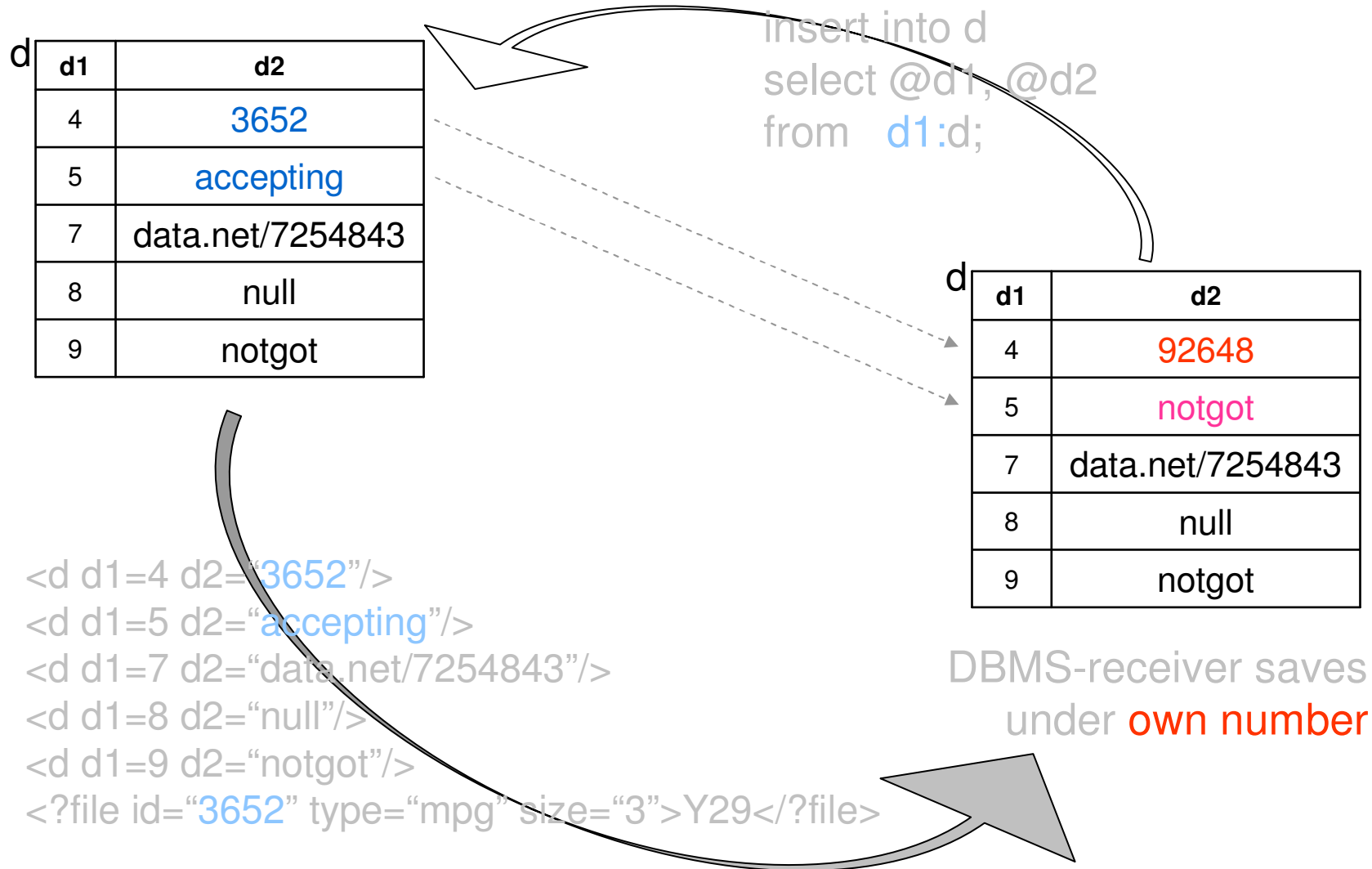
If transfer of BLOB is broken at 1000nd byte,
then **client automatically repeats** query for
the tail of BLOB

```
select @d2[1000:]  
from db1:d  
where @d2=3652;
```

```
<d d2[1000:]="3652"/>
```

```
<?file id="3652" size="3">udG</?file>
```

Copying BLOB from DB to DB



Nested schemas like
former folders

Instead of many databases

```
connect address=site.com schema=sn1/sn2;
create schema sn5         inside  sn3/sn4         uni=goods; -- uni is UNlque name
connect address=site.com schema=sn1/sn2/sn3/sn4/sn5;
connect address=site.com                               uni=goods;
connect                               schema=../..;    -- go two schemes above
connect                               schema=./sn3/sn4;  -- go into sub-schema
-- root schemas with name of one letter are external storage like DVD/BluRay
```

```
alter schema sn1/sn2/sn3/sn4/sn5 rename to sn7;
alter schema sn1/sn2/sn3/sn4/sn7 [copy|move] to [address=data.net] uni=my;
alter schema sn1/sn2/sn3/sn4/sn7 [copy|move] to [address=data.net] schema=sn1/sn2;
```

-- after move

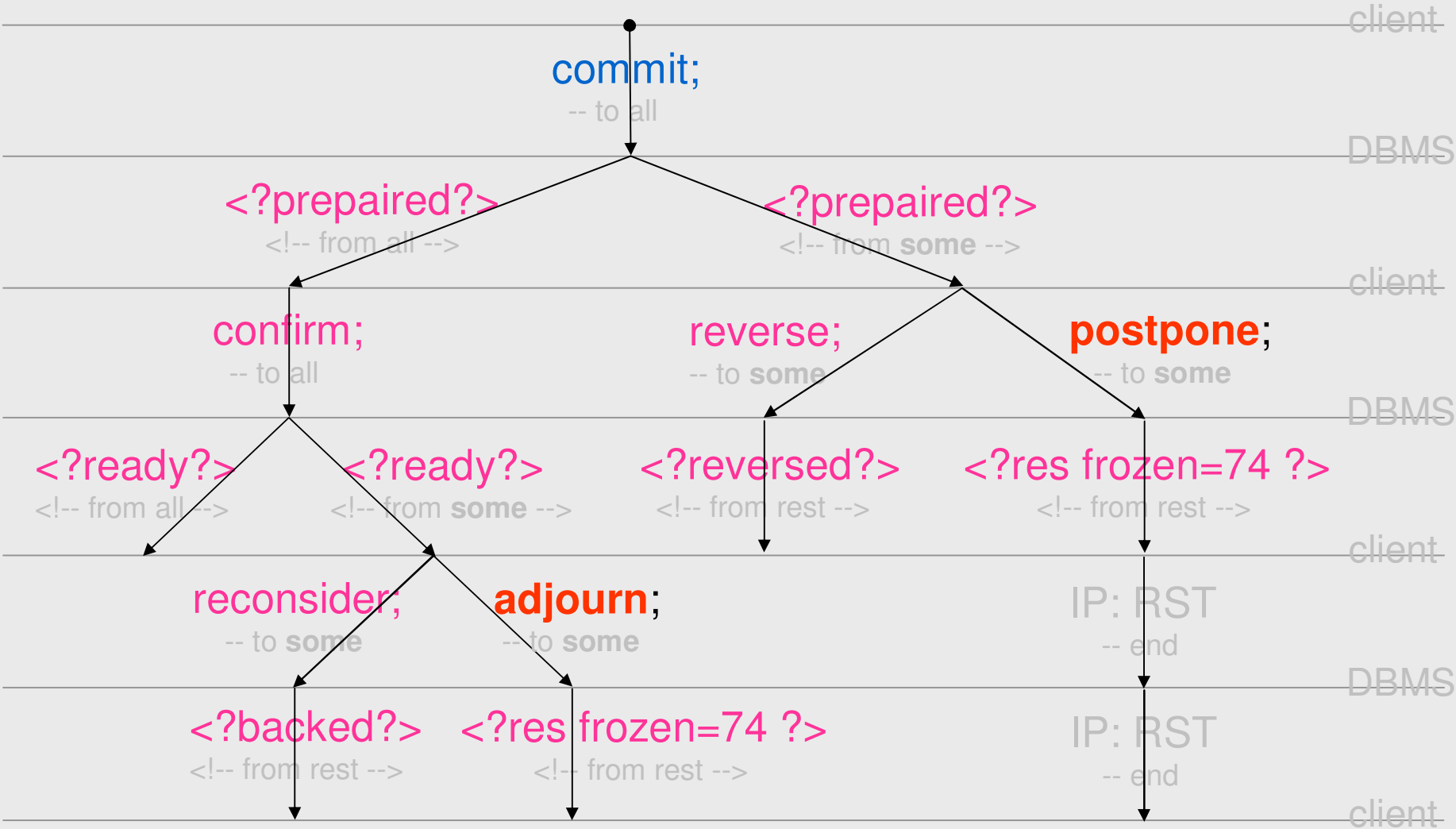
```
alter schema sn1/sn2/sn3/sn7 uni to chernobyl;
create nick db1 address=data.net uni=my;
alter nick db1 address=site.com uni=chernobyl; -- only uni, never schema
```

Repairing of dQ ,
optionally

2PC, 3PC

Operator 'POSTPONE' to freeze 2- and 3-phase 'COMMIT' (2PC and 3PC) on second phase, and operator 'ADJOURN' to freeze 3-phase 'COMMIT' (3PC) on third phase – to repair **only failed** databases after from their log-files

Freezing stages of 2PC and 3PC



Different

Permutation

Permutation for fields

from **a**, x

where **@a1**~=@x1 and **@a2**~=@x1 and **@a3**~=@x1;

where **@a1**<>**@a2** and **@a2**<>**@a3** and **@a1**<>**@a3**
and **@a1** in (select @x1 from x)
and **@a2** in (select @x1 from x)
and **@a3** in (select @x1 from x)

Permutation for cartesian product or tree

```
from a, b, c, x -- cartesian product of 'a', 'b', 'c'  
where @a1~=@x1 and @b1~=@x1 and @c1~=@x1;
```

```
from a.b.c, x -- tree 'a.b.c'  
where @a1~=@x1 and @b1~=@x1 and @c1~=@x1;
```

```
where @a1<>@b1 and @b1<>@c1 and @a1<>@c1  
and @a1 in (select @x1 from x)  
and @b1 in (select @x1 from x)  
and @c1 in (select @x1 from x)
```

Permutation for '*', '+'

from a.b*.c, x where @b1 ~=@x1;

```
from a.b[@b1 as k1].b[@b1 as m1].b[@b1 as n1].b[@b1 as
      p1].b[@b1 as q1].b[@b1 as r1].b[@b1 as s1]. ... .c
where @k1<>@m1 and @k1<>@n1 and @k1<>@p1 and @k1<>@q1 and @k1<>@r1 and @k1<>@s1 and ...
      and
      @m1<>@n1 and @m1<>@p1 and @m1<>@q1 and @m1<>@r1 and @m1<>@s1 and ...
      and
      @n1<>@p1 and @n1<>@q1 and @n1<>@r1 and @n1<>@s1 and ...
      and
      @p1<>@q1 and @p1<>@r1 and @p1<>@s1 and ...
      and
      @q1<>@r1 and @q1<>@s1 and ...
      and
      @r1<>@s1 and ...
and @k1 in (select @x1 from x)
and @m1 in (select @x1 from x)
and @n1 in (select @x1 from x)
and @p1 in (select @x1 from x)
and @q1 in (select @x1 from x)
and @r1 in (select @x1 from x)
and @s1 in (select @x1 from x)
and ... ;
```

Whole control

Rights for each record

User enters not separate records into database, but whole graphs. Graphs of different users (different roles) are cross little, so all records of database are decomposed into little interacting classes (**departments**).

```
create department dp;
```

It is reasonable to give identical rights on all records of one department to user (to role), and to specify department name in records themselves in fields of special datatype “**legal**”

```
create table a (... , a5 legal, ...);  
insert into a values (... , dp, ...);  
bestow select on dp for usr;  
bestow update on dp for rolename;
```

Hiding and protection

We shall withdraw rights on department by following operator

```
VANISH select ON dp FOR usr;
```

```
VANISH update ON dp FOR rolename;
```

- if user has no rights to some operation with record, than record does not exist for this user (he does not see it in this operation)
- if record has no field of datatype **legal**, or this field is equal NULL, than user has all rights on this record
- field **legal** is accessible for updating on the same basis, as whole record

Use case

We shall specify for user convenient (and for administrating database too) at creating user, by which department he marks all inserted or updated records by default. We shall specify this department in parameter **trace**.

```
create user          u1
                    identified by p1
                    trace          dp;
```

Controllable and usable
full text search
(cuFTS)

Table, nested in textual field

S

pk	s1	s2
1	10	In the morning, dog comes, cat comes home too. Continue in the NEXT issue

@token	@sn	@beginning	@end
In	1	1	2
the	2	4	6
morning	3	8	14
dog	4	17	19
comes	5	21	24
cat	6	27	29
comes	7	31	34
home	8	36	39
too	9	41	43
Continue	10	46	53
in	11	55	56
the	12	58	60
NEXT	13	62	65
issue	14	67	71

Syntactical access:

@s2/@token

@s2/@sn

@s2/@beginning

@s2/@end

insert into s (@s2/@token, @s2/@sn) values ("new", 15)
update s SET @s2/@token = "" || @s2/@token || "";
delete from s where @s2/@beginning >= 100;


Solution of collisions

S

pk	s1	s2
1	10	In the morning, dog comes, cat comes home too. Continue in the NEXT issue

```
select @s1, @s2/@token
from s
where @s2/@sn in (
  select distinct @s2/@sn
  from s, (
    select @s2/@sn as fn
    from s
    where @s2/@token in "comes next"
  )
)
where abs(@sn-fn) <= 1
);
```

propagated group



s1	s2	sys_clue
10	dog comes, cat ... the NEXT issue	1
10	cat comes home ... the NEXT issue	2

Operations with nested column

Concatenation of two nested columns
is
UNION for them,
and gives new nested table

Surrounding by tags

pk	s1	s2
1	10	In the morning, dog comes , cat comes home too. Continue in the NEXT issue

```

select @s1, ("<b>" || @s2/@token as @f1 || "</b>") ||
          ("<em>" || @s2/@token as @f2 || "</em>")
from s
where @f1 in "comes next"
and @f2 in (
  select distinct(@s2/@token, @s2/@sn) @s2/@token
  from s, (
    select @s2/@sn as fn
    from s
    where @s2/@token in "comes next"
  )
  where abs(@s2/@sn-fn)=1
);

```

alias for function argument

concatenation

s1	s2	SYS_CLUE
10	dog comes, cat ... the NEXT issue	1
10	cat comes home ... the NEXT issue	2

Table of tokens

tokenize s(@s2) into tokens delimiting delimiters;
copy tokens(@idlexeme, @token) from c:/lexeme.txt

tokens

idtoken	token	idlexeme
1	in	1
2	the	2
3	morning	3
4	dog	4
5	comes	5
12	come	5
6	cat	6
7	home	7
8	too	8
9	continue	9
10	next	10
11	issue	11

Table with
blank, tab,
/n, /r,
marks of punctuation

leaves 'idlexeme' equal null

} all grammatical forms

```
create index i1 on tokens( @idtoken );  
create index i2 on tokens( @token );  
create index i3 on tokens( @idlexeme );
```

Table of items

itemize s(@s2) into items delimiting delimiters tokenizing tokens;
set nomenclature items; -- for operations 'in', etc

items

idfield	pk	idtoken	own name	abbr	sn	beginning	end
505	1	1	yes		1	1	2
505	1	1			11	55	56
505	1	2			2	4	6
505	1	2			12	58	60
505	1	3			3	8	14
505	1	4			4	17	19
505	1	5			5	21	24
505	1	5			7	31	34
505	1	6			6	27	29
505	1	7			8	36	39
505	1	8			9	41	43
505	1	9	yes		10	46	53
505	1	10		yes	13	62	65
505	1	11			14	67	71

Side positive effect –
new sub-fields:

@s2/@idtoken

@s2/@idlexeme

@s2/@idfield


id of table column (unique in whole database)

```
create index i4 on items( @idfield, @pk, @idtoken );  
create index i5 on items( @idfield, @pk, @sn );
```

Freeezing

Frozen transaction

create user u identified by p waited 1.0/0; -- yy.mm.dd/hh.mm.ss;

freeze;  <?res code=0 frozen=7482 ?> <!-- from 'default.edu' -->
<?res code=0 frozen=8726 ?> <!-- from 'site.com' -->
<?res code=0 frozen=9278 ?> <!-- from 'data.net' -->
<?res code=0 frozen=3825 ?> <!-- from 'data.net': second transaction -->
<?res code=0 frozen=6384 ?> <!-- from 'store.org' -->
<?res code=5 ?> <!-- from 'place.ws': database is broken -->

unfreeze address=site.com user=Smith pwd=ncwhif safe=8726;

unfreeze address=data.net user=Darvin pwd=qxuwb safe=9278;

To switch-on and switch-off notebook several times during one long transaction, lasting days or even weeks, it's enough to enter operator 'FREEZE (similar to 'DISCONNECT'), which saves transaction in current state; and operator 'UNFREEZE (similar to 'CONNECT'), which continues transaction from frozen state (instead starts new transaction, as 'CONNECT'). 'FREEZE' returns identifier of frozen transaction, which should be specified in 'UNFREEZE'

Useful before freeze

```
commit savepoint a;  
freeze;
```

Timer

Timer is

procedure, which is actual from moment, specified in parameter **'start'**, and to moment, specified in parameter **'end'**. Whole gap of time from 'start' to 'end' is divided into equal intervals of time, duration of which is specified in parameter **'per'** (quantity of intervals 'per' can be not integer of gap from 'start' to 'end'). On each this interval (including last and not equal 'per' itself), function is automatically executed in moments, lagging from beginning of interval to distance, specified in parameter **'schedule'**.

Syntax

```
create timer TimerName
  start      yy.mm.dd/hh.mm.ss
  end        yy.mm.dd/hh.mm.ss
  schedule (yy.mm.dd/hh.mm.ss, yy.mm.dd/hh.mm.ss, ...)
  per        yy.mm.dd/hh.mm.ss
  as begin ... end;
```

```
create timer t1 schedule (01/0, 03/0, 10/0) per 01.00/0 as ... ;
-- 1st, 3rd, 10th days of each month
create timer t2 schedule (0/0.05, 0/0.10, 0/0.20) per 0/01 as ... ;
-- 5th, 10th, 20th minutes of each hour
create timer t3 schedule (0) per 1/0 as ... ;
-- each day
```


Interval table constraints

Notrifi, Notcross, Sole

Either for `array[2]`, or for data types,
composed of **only two elements** of the **same** data type

- **only one** array or composed data type under constraint
- **first** element must be **less or equal** second element
- only data types
natural, enum, UUID, integer, float, money, or datetime

Notrift, Notcross

```
create type t as (beg datetime, end datetime);
```

```
create table a ( -- for intervals [@a1/@beg, @a1/@eng]
  a1 t,
  notcross (a1), -- must not cross in whole table
  notrift (a1) -- must be continuous line without gaps
);
```

```
create table aa ( -- for intervals [@aa1[1], @aa1[2]]
  aa1 datetime array[2],
  notcross (aa1), -- must not cross in whole table
  notrift (aa1) -- and must be continuous line without gaps
);
```

```
create table b ( -- for intervals [@b1/@beg, @b1/@eng]
  b1 t,
  b2 number,
  b3 number,
  notcross (b1, b2, b3), -- must not cross within records with identical '@b2, @b3'
  notrift (b1, b2, b3) -- must be continuous line within records with identical '@b2, @b3'
);
```

```
create table bb ( -- for intervals [@bb1[1], @bb1[2]]
  bb1 datetime array[2],
  bb2 number,
  bb3 number,
  notcross (bb1, bb2, bb3), -- must not cross within records with identical '@b2, @b3'
  notrift (bb1, bb2, bb3) -- must be continuous line within records with identical '@b2, @b3'
);
```

Sole

```
create type t as (beg datetime, end datetime);
```

```
create table c ( -- for intervals [@c1/@beg, @c1/@eng]
  c1 t,
  sole (c1) -- must be sole in whole table
);
```

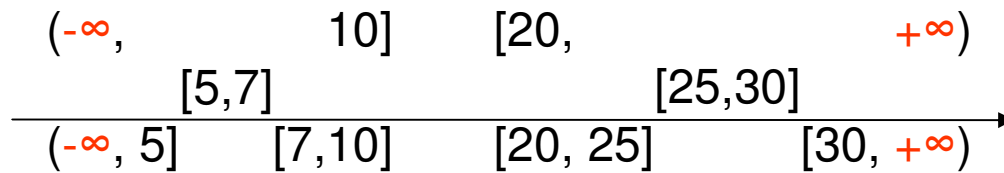
```
create table cc ( -- for intervals [@cc1[1], @cc1[2]]
  cc1 datetime array[2],
  sole (cc1) -- must be sole in whole table
);
```

```
create table d ( -- for intervals [@d1/@beg, @d1/@eng]
  d1 t,
  d2 number,
  d3 number,
  sole (d1, d2, d3) -- must be sole within records with identical '@d2, @d3'
);
```

```
create table dd ( -- for intervals [@dd1[1], @dd1[2]]
  dd1 datetime array[2],
  dd2 number,
  dd3 number,
  sole (dd1, dd2, dd3) -- must be sole within records with identical '@d2, @d3'
);
```

'Notcross' changes values 'eternity' ($-\infty$), 'infinity' ($+\infty$)

necessary for arithmetic calculations



```
create table g (  
  g1 number  
  g2 number array[2],  
  notcross (g2)  
);
```

```
insert into g values (1, array[eternity, 10]);  
insert into g values (2, array[20, infinity]);
```

```
insert into g values (3, array[5, 7]);  
insert into g values (4, array[25, 30]);  
select * from g;
```

```
1, [eternity, 5]  
3, [ 5, 7]  
1, [ 7, 10]  
2, [20, 25]  
4, [25, 30]  
2, [30, infinity]
```

Model into Window System

Concept for 3D

Office technologies stopped

It has occurred because user, capable to write primitive programs, can't:

- display 3-dimensional data in window of own program
- move 3-dimensional objects by commands

There is a need for:

- 3D-objects, written as triangles (which are understandable for users) instead of constructor of objects in SQL, because user does not master formulating and considering of own problems via constructors

Figure by triangles

```
create table concourse (id      number,  
                        color   number array[3] color,  
                        offset   number array[3] shift, -- displacement of concourse  
                        turn     number array[3] krylov, -- rotation of figure in e.g. Krylov angles  
                        distance number array[2] span, -- around geom.center of concourse  
                        note     varchar);  
  
create table figure (id      number,  
                    conc    number      references concourse(@id),  
                    color   number array[3] color,  
                    on 2 or more levels  
                    higher bottom 'dot' { disp  number array[3] shift, -- displacement of figure  
                    rot      number array[3] euler, -- rotation of figure in e.g. Euler angles  
                    range   number array[2] span, -- around geometric center of figure  
                    note     varchar);  
  
create table triangle (fig    number      references figure(@id),  
                      vertex  number array[3] references point(@id),  
                      more    varchar);  
  
create table point (id      number,  
                   coord   number array[3] dot, -- coordinates of point  
                   time    datetime array[2] period, -- more usable than 'select...period'  
                   note     varchar);
```

data type visual type

New step in tech: DBMS

And a need for:

- DBMS, returning result of operator SELECT in format X11, that program could send obtained data to the X-server without any own calculations or changes (without referencing to OpenGL, DirectX, which user never will masters!)

```
select * from concouse.* UNION select * from gathering.*
```

```
PROJECTING [perspectively|planarly]
```

```
[ camera (0, 0, 0) ] -- location of camera
```

```
[ width (3, 4) ] -- width of camera
```

```
[ look (1, 1, 1) ] -- direction of camera
```

```
;
```

Usability

3D-construction is presented as hierarchy of points, triangles, figures (groups of triangles), *groups of figures*, *groups of groups*, etc. Record, containing these objects, must be bound by foreign key. To extract all triangles of a figure or a group, it's necessary to select by operator 'SELECT' not only record, being this figure or group, but also all records, located below it in hierarchy - up to records, which present 3D-points. For this, tables of these records are listed through point - **"SELECT * FROM group3.group2.group1.triangle.point PROJECTING"** - as this is accepted in classical object-oriented languages. Names of tables are *not important*, because *points are always expected to be located on the most lower level* of hierarchy of each branch of tree, triangles - on one level above, and records of other levels are not visualized. Before transformation into format X11, invisible triangles will be removed from result of query, and partly visible triangles will be cut to their visible parts. So any program displays contents of a database on screen by function **'printg("SELECT * FROM group3.group2.group1.triangle.point PROJECTING")'**.

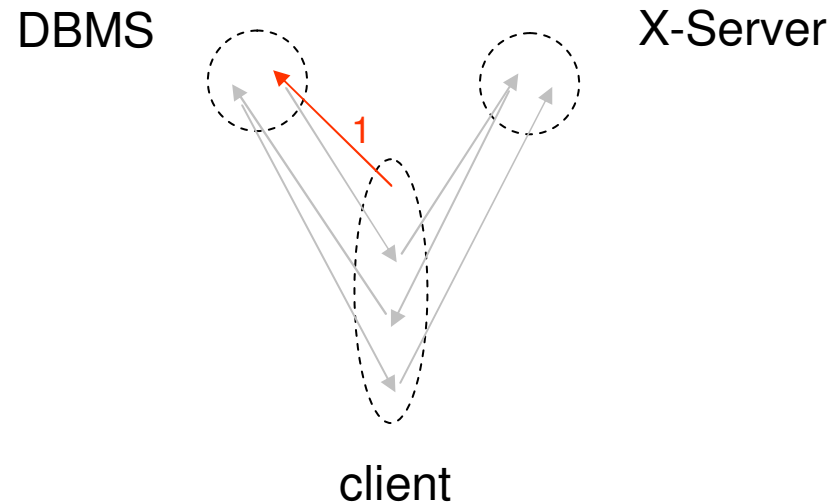
Driven scene

It will be enough for user to call one function in OS, we shall name it as 'printg', in which user specify what 3D-objects from database must appear on a screen (this function being called, automatically forwards all mouse motion commands of objects and motion of camera into DBMS; all messages about motion of objects, obtained from DBMS, automatically forwarded into X-server).

It will be also enough to call this function, i.e. 'printg', to send order to change coordinates on SQL (change immediately is displayed on screen, because messages from DBMS are automatically forwarded to X-server after first call of 'printg'). And 'printg' will be in library of standard input/output, and can be called in any program.

Participation of user

User submits only first (1) query, all next parcels (mouse motion commands of objects and of camera, correction X11-data because UPDATE in DBMS) are sent by client library, DBMS, X-Server automatically, independently of user.



Example: Overview of all figures

create function geodia (**arg** table) returns (

 @geocenter num array[3],

 @diameter num array[3]

) as begin

 select (@mi+@ma)/2 as @geocenter, @ma-@mi as @diameter

 from (

 select min(@coord) as @mi, max(@coord) as @ma

 from **arg**.*.point

);

end;

select * from a.*

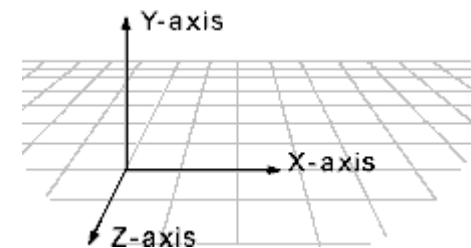
 projecting

 eye (

 select @geocenter + [0,0, @diameter[3]/2+max(@diameter[1],@diameter[2])]

 from geodia (**a**)

) look (0,0,-1);



New step in tech: Ways

And also a need for:

■ two ways to change the coordinates (in field of numeric datatype, marked by **visual type “dot”**), both ways create commands in format X11 to control X-Server, which are immediately forwarded to the X-server:

- by operator UPDATE (that allow a program to change coordinates)
- accepting events in format X11, which are also commands to change coordinates, but are send by mouse (that allows to change coordinates by mouse)

```
update a.* shift (5, 5, 5);
```

```
update a.* krylov (45, 45, 45) center (0, 0, 0); -- rotate
```

```
update a.* euler (45, 45, 45) center (0, 0, 0); -- rotate
```

```
update a.* sprain (5, 5, 5) center (0, 0, 0);
```

Commands from program

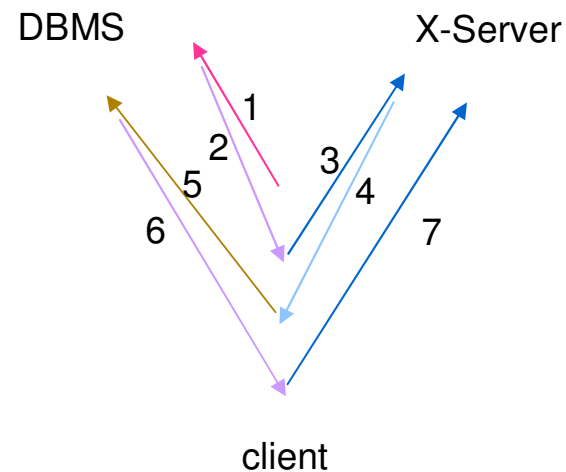
Similarly it's also possible to specify whole hierarchy of records from figure or group to 3D-points in operator UPDATE –
"update group3.group2.group1.triangle.point set ...". So any program changes position of an object in database and on screen by function 'printg ("update group3.group2.group1.triangle.point set ...")'.

```
update point set @coord=mouse; -- value for array[3]  
-- point is always in place, where mouse is
```

```
update point set @coord=default;  
-- see 'create table point (coord number array[3] default ... )'
```


DBMS as Visualization server

Functionality of OpenGL
must be implemented in DBMS



1. query:
 - select ... **PROJECTING** ...
 - insert, update, delete

2, 6 `<?xd size=345>...</?xd>`

3, 7 ...

4 ...

5 `<?xe size=345>...</?xe>`

`<!-- data for X11-server, wrapped in XML -->`

`<!-- data itself for X11-server-->`

`<!-- event itself from X11-server -->`

`<!-- event from X11-server, wrapped in XML -->`

Shift and rotation on levels to refer to the same figure several times

```
create table concourse (id          number,
                        color       number array[3] color,
                        offset      number array[3] shift, -- displacement of concourse
                        turn        number array[3] krylov, -- rotation of figure in e.g. Krylov angles
                        distance    number array[2] span, -- around geom.center of concourse
                        note        varchar);

create table figure (id          number,
                    conc         number          references concourse(@id),
                    color       number array[3] color,
                    on 2 or more levels
                    higher bottom 'dot' } disp      number array[3] shift, -- displacement of figure
                    rot         number array[3] euler, -- rotation of figure in e.g. Euler angles
                    range       number array[2] span, -- around geometric center of figure
                    note        varchar);

create table triangle (fig         number          references figure(@id),
                      vertex      number array[3] references point(@id),
                      more        varchar);

create table point (id          number,
                   coord        number array[3] dot, -- coordinates of point
                   time         datetime array[2] period, -- more usable than 'select...period'
                   note         varchar);
```

Time and distance constraints

Movie

```
create table concourse (id          number,  
                        { color    number array[3] color,  
                          offset  number array[3] shift, -- displacement of concourse  
                          turn    number array[3] krylov, -- rotation of figure in e.g. Krylov angles  
                          distance number array[2] span, -- around geom.center of concourse  
                        note      varchar);  
create table figure      (id          number,  
                        conc         number          references concourse(@id),  
                        { color    number array[3] color,  
                          disp    number array[3] shift, -- displacement of figure  
                          rot     number array[3] euler, -- rotation of figure in e.g. Euler angles  
                          range   number array[2] span, -- around geometric center of figure  
                        note      varchar);  
create table triangle   (fig         number          references figure(@id),  
                        vertex     number array[3] references point(@id),  
                        more       varchar);  
create table point      (id          number,  
                        coord      number array[3] dot, -- coordinates of point  
                        time       datetime array[2] period, -- more usable than 'select...period'  
                        note      varchar);
```

'update' creates new record and thus line by this point in 4D space

Part of movie in real time scale

```
select * from concouse.*
```

```
where -- all time without 'where' for time
```

```
point/@time & -- '&' is operator of intersection of intervals
```

```
[09.01.01/05.00, 09.01.01/10.00]
```

```
-- select [@time1,@time2] from ...
```

```
-- select @timeinterval from ...
```

```
PROJECTING [cycled|not cycled];
```

```
point/@time & 09.01.01/05.00
```

```
point/@time & now
```

```
-- one movie's frame
```

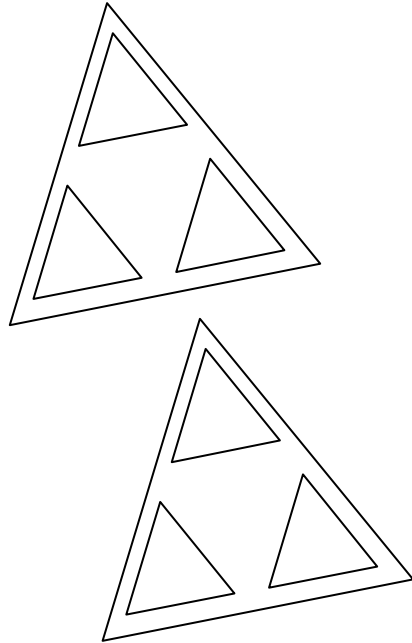
```
PROJECTING;
```

Distance

```
create table concourse (id      number,  
                        color   number array[3] color,  
                        offset  number array[3] shift, -- displacement of concourse  
                        turn    number array[3] krylov, -- rotation of figure in e.g. Krylov angles  
                        distance number array[2] span, -- around geom.center of concourse  
                        note    varchar);  
  
create table figure (id      number,  
                    conc    number      references concorse(@id),  
                    color   number array[3] color,  
                    disp    number array[3] shift, -- displacement of figure  
                    rot     number array[3] euler, -- rotation of figure in e.g. Euler angles  
                    range   number array[2] span, -- around geometric center of figure  
                    note    varchar);  
  
create table triangle (fig    number      references figure(@id),  
                      vertex  number array[3] references point(@id),  
                      more    varchar);  
  
create table point (id      number,  
                   coord   number array[3] dot, -- coordinates of point  
                   time    datetime array[2] period, -- more usable than 'select...period'  
                   note    varchar);
```

on 2 or more levels
higher bottom 'dot'

Visibility of details depends



If figure contains

6 small triangles with value of **span** field inside [10, 100], and
2 big triangles with value of **span** field inside [100, 9999], then

- geometrical center of **all 8** triangles is calculated
- the following is shown at distance between **camera** and **geometrical center**:
 - nothing, at distance less 10
 - only 6 small triangles, at distance from 10 to 100
 - only 2 big triangles, at distance from 100 to 9999
 - nothing, at distance farther 9999

Session parameters

```
set 3Dcamera (0, 0, 0) ; -- move camera  
set 3Dlook   (1, 1, 1) ; -- turn camera  
set 3Dwidth  (3, 4) ; -- change width of camera
```

Particularities

Branches:

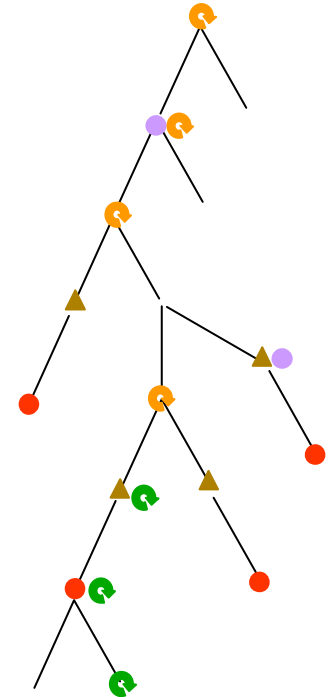
- operation *ignores* branches, not specified in query
- operation *ignores* branches, not satisfied to conditions for sections and fields

Sections:

- table names (section names) are *un-important*
- bottom 'dot' of each branch contains **point**
- one section above of each branch contains **triangles**
- other sections are *invisible*

Fields:

- 'dot' is *ignored*: if not 3 elements in array, if not sole 'array[3] dot', if is **above** other 'array[3] dot'
- 'color, shift, euler/krylov, span' are *ignored*: in triangles, in points, in sections **below**, if not sole in record
- **triangle** is *ignored*, if ambiguity because no refinement ([slide 13](#))
- all other fields are *ignored*



2D-grid in 3D-space

Grid with texture

```

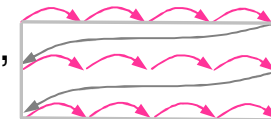
create table concourse (id          number, ...);
create table figure      (id          number,
                          cc          number          references concourse(@id),
                          color       number array[3] color, -- ignored, if texture is not null
                          disp        number array[3] shift, -- displacement of figure
                          rot         number array[3] euler, -- rotation of figure in e.g Euler angles
                          range       number array[2] span, -- around geometric center of figure
                          side        number          references quadrangle(@id),
                          pic         blob           texture, -- rectangular picture for whole grid
                          d           number         diffuse,
                          rf          number         reflect,
                          rd          number         radiate,
                          note        varchar);
create table quadrangle(id          number,
                        next         number          references quadrangle(@id),
                        vertex       number array[4] references point(@id),
                        note        varchar);
create table point      (id          number,
                        coord        number array[3] dot, -- coordinates of point
                        time        datetime array[2] period, -- more usable than 'select...period'
                        note        varchar);

```

on 2 or more levels higher bottom 'dot'

also on 2 or more levels higher bottom 'dot'

-- 2D grid in 3D space



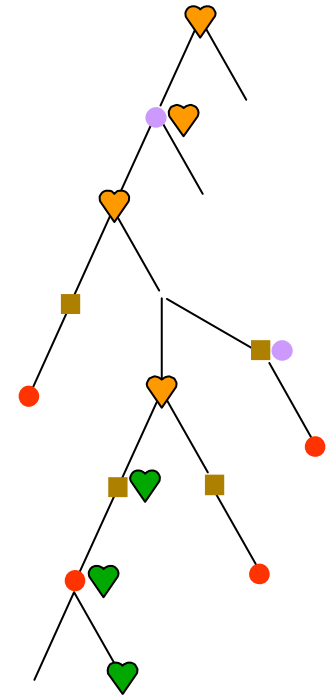
Difference in Particularities

Sections:

- one section above 'dot' of each branch contains quadrangles

Fields:

- 'texture, diffuse, reflect, radiate' are *ignored*: in quadrangles,
- in points, in sections **below**, if not sole in record
- quadrangle is *ignored*, if ambiguity because no refinement ([slide 13](#))



3D-triggers

Trigger for shift, sprain

```
create trigger t on tab
after  shift
when @offset.x=      -- distance of shift
  and @distab=       -- table name, on which 'tab' is dropped
  and @dispk=        -- value of primary key, on which 'tab' occurs
as begin ...
  ... where @fld=@offset.z;
```

```
create trigger t on tab
after  sprain
when @coef.x= -- is normalized
  and @center.x=
  and @distab=
  and @dispk=
as begin ...
  ... where @fld=@coef.z;
  ... where @fld=@center.z;
```

Trigger for rotation

```
create trigger t on tab
after rotation
when @krylov.x= -- Krylov angle of rotation
 [or @euler.x= ] -- Euler angle of rotation
and @center.x=
and @distab= -- table name, on which 'tab' is dropped
and @dispk= -- value of primary key of object,
 -- on which 'tab' is dropped

as begin ...
... where @fld=@krylov.z;
... where @fld=@euler.z;
```

Mouse triggers

Mouse operations terms

worker – object, obtained focus. Always point (first point of two ends of contour segment or triangle or quadrangle border, at clicking on them), triangle, or quadrangle (at clicking in empty space between them)

focus – event; first click on object, accepting click (object becomes *worker*)

shift, rotation – event; motion on screen by mouse, taking object itself

sprain – event; motion on screen by mouse, taking dotted square around object

destiner – object, lost focus

First accepting object obtains focus

```
create table a (  
  id number,  
  focus – figure 'a' does not obtain focus  
);  
create table b (  
  id number,  
  lnk number          references a(@id),  
  focus – figure 'b' does not obtain focus  
);  
create table c (  
  id number,  
  ref number          references b(@id),  
  v number array[3] references d(@id)  
  focus – triangle does not obtain focus  
);  
create table d ( -- click is here  
  id number,  
  x number array[3] dot  
  focus -- point obtains focus  
);
```

```
create table a (  
  id number,  
  focus – figure 'a' does not obtain focus  
);  
create table b (  
  id number,  
  lnk number          references a(@id),  
  focus – figure 'b' obtains focus  
);  
create table c (  
  id number,  
  ref number          references b(@id),  
  v number array[3] references d(@id)  
  – triangle does not obtain focus  
);  
create table d ( -- click is here  
  id number,  
  x number array[3] dot  
  – point does not obtain focus  
);
```



Focus

```
create table figure (  
  id          number,  
  footnote    varchar,  
  [constraint c] focus [accept|reject]'  
);
```

Trigger

```
create trigger t on a  
after focus
```

Programmably

```
select focus (tablename)  
where @pk=3;
```

Trigger for click, double-click

create trigger t on tab

after **click**

-- other meaning is 'double'

when **@self=yes**

-- i.e. worker=distiner; other is 'no'

and **@distab=**

-- table name of 'distiner'

and **@dispk=**

-- value of primary key of 'distiner'

as begin ...

Programmably

select **click** (tablename) where @pk=3;

select **double** (tablename) where @pk=3;

Operations

Operations: 3D-comparisons

-- return boolean

... where a.* = b.*;

-- the same dots

... where a.* \sim = b.*;

-- the same volume and shape

... where a.* in b.*;

-- b.* contains a.*

... where a.* in= b.*;

-- (a.* in b.*) or (a.* = b.*)

... where a.* in^ b.*;

-- a.* in b.* and touch b.*

... where a.* out^ b.*;

-- a.* not in b.* and touch b.*

... where a.* back in b.*;

-- b.* in a.*

... where a.* back in= b.*;

-- b.* in= a.*

... where a.* back in^ b.*;

-- b.* in^ a.*

~~... where intersects(a.*, b.*);~~

~~-- is already occupied for intersection of 'select'~~

Operations: 3D-calculations

... where **measure**(a.*)=5; -- volume

-- return new triangles and points under 3rd level figures of left tree (a.*)

... where a.* **&** b.*; -- common records (triangles, points) in 2 low levels

... where a.* **~&** b.*; -- geometrical overlap of figures (3rd level from bottom)

... where a.* **^** b.*; -- geometrical place of touch of triangles (but not cross)

... where a.* **-|** b.*; -- geometrical place of cross of triangles (but not touch)

... where a.* **^-|** b.*; -- (a.* ^ b.*) union (a.* -| b.*)

... where a.* **<|** b.*; -- closest point of left tree (a.*) to right tree (b.*)

... where a.* **|>** b.*; -- b.* |> a.*

... where a.* **<->** b.*; -- distance between figures

3D-contour

Broken line

```
create table concourse (id          number,
                        color       number array[3] color,
                        offset      number array[3] shift, -- displacement of concourse
                        turn        number array[3] krylov, -- rotation of figure in e.g. Krylov angles
                        distance    number array[2] span, -- around geom.center of concourse
                        note        varchar);

create table figure (id          number,
                    conc         number          references concourse(@id),
                    color       number array[3] color,
                    on 1 or more levels
                    higher bottom 'dot' { disp      number array[3] shift, -- displacement of figure
                                         rot        number array[3] euler, -- rotation of figure in e.g. Euler angles
                                         range     number array[2] span, -- around geometric center of figure
                                         line      number          references point(@id),
                                         note     varchar);

create table point (id          number,
                   next         number          references point(@id),
                   coord        number array[3] dot, -- coordinates of point
                   time         datetime array[2] period, -- more usable than 'select...period'
                   note         varchar);
```

- **contour** is broken line, which **first and last points coincide**
- surface is never described by any contour

Operations: calculations

... where `measure(a.*)=5;` -- length, perimeter

-- return new lines under 2nd level figures of left tree (a.*)

... where a.* `&` b.*; -- common records (lines) in 1st low levels

... where a.* `~&` b.*; -- geometrical overlap of figures (2nd level from bottom)

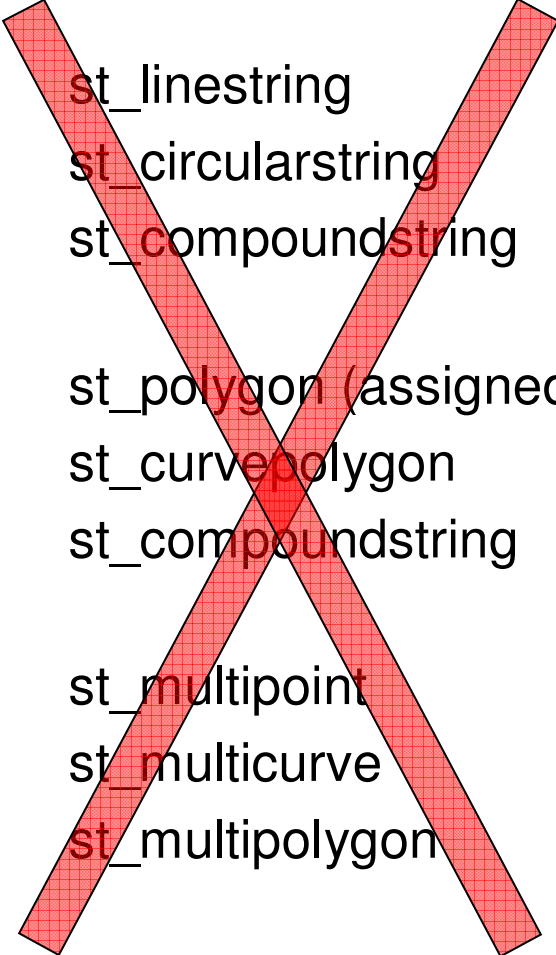
... where a.* `^` b.*; -- geometrical place of touch of lines (but not cross)

... where a.* `-|` b.*; -- geometrical place of cross of lines (but not touch)

... where a.* `^-|` b.*; -- (a.* ^ b.*) union (a.* -| b.*)

Concept for 2D

Current way is not usable



~~st_linestring
st_circularstring
st_compoundstring
st_polygon (assigned "st_linestring")
st_curvepolygon
st_compoundstring
st_multipoint
st_multicurve
st_multipolygon~~

We are forced to learn sophisticated SQL to use redundant datatypes, where we can use usual schema of data

Array instead of new datatype

```
create table intersection (  
  id num,  
  location st_point  
);
```

```
create table intersection (  
  id num,  
  location num array[2] dot  
);
```

Array instead of them

```
insert into intersection values (  
  1, new st_point(0,0)  
);
```

```
<intersection id="1" location="0 0">;
```

Schema instead of new datatypes

```
create table street (  
  name varchar,  
  geo st_curve  
);
```

```
create table street (  
  name varchar,  
  first num references segment(id)  
);  
create table segment (  
  id num,  
  next num references segment(id),  
  point num array[2] dot  
);
```

Hierarchical data instead of them

```
insert into street values (  
  'maple',  
  new st_linestring (  
    array [  
      new st_point( 0,0),  
      new st_point(600,0)  
    ]  
  )  
);
```

```
<street name="maple">  
  <segment point=" 0 0">  
  <segment point="660 0">  
</street>  
;
```

Once again, schema instead

```
create table parcel (  
  name varchar,  
  geo st_curvepolygon  
);
```

```
insert into parcel values (  
  'maple',  
  new st_polygon (  
    new st_linestring (  
      array [  
        new st_point(200, 25),  
        new st_point(400, 25),  
        new st_point(400,400),  
        new st_point(200,400),  
        new st_point(200, 25)  
      ]  
    )  
  )  
);
```

```
create table parcel (  
  name varchar,  
  first num references segment(id)  
);  
create table segment (  
  id num,  
  next num references segment(id),  
  point num array[2] dot  
);
```

```
<parcel name="maple">  
  <segment point="200 25">  
  <segment point="400 25">  
  <segment point="400 400">  
  <segment point="200 400">  
  <segment point="200 25">  
</parcel>  
;
```

Instead of pretentious datatypes

```
create table concourse (id      number,
                        depth   int      layer,
                        color   number array[3] color, -- both for border and for surface
                        offset  number array[2] shift, -- displacement of concourse
                        turn    number      rotation, -- rotation of concourse
                        distance number array[2] span, -- around geom.center of concourse
                        note    varchar);

create table figure (id      number,
                   conc     number      references concourse(@id),
                   depth   int      layer,
                   color   number array[3] color, -- both for border and for surface
                   disp    number array[2] shift, -- displacement of figure
                   rot     number      rotation, -- rotation of figure
                   range   number array[2] span, -- around geometric center of figure
                   line    number      references point(@id),
                   note    varchar);

create table point (id      number,
                  next     number      references point(@id),
                  coord    number array[2] dot, -- coordinates of point
                  time     datetime array[2] period, -- more usable than 'select...period'
                  note     varchar);
```

on 1 or more levels
higher bottom 'dot'

-- contour and surface are not differ in operations
-- contour is broken line, which first, last points coincide

Operations: Projection

```
select * from concouse.* UNION select * from gathering.*  
where point/@time & [09.01.01, 09.01.01]
```

PROJECTING

```
[ eye (0, 0) ] -- location of camera
```

```
[ width (3, 4) ] -- width of camera
```

```
[cycled|not cycled];
```

```
-- un-visible surfaces and broken lines are ignored
```

```
-- partly un-visible surfaces and broken lines are cut  
before putting into output
```

```
-- wholly visible surfaces and broken lines are put into output
```

Session parameters

```
set 2Dcamera (0, 0) ; -- move camera
```

```
set 2Dwidth (3, 4) ; -- change width of camera
```

Operations: Motion, deformation

update a.* **shift** (5, 5);

update a.* **rotate** (45) **center** (0, 0);

update a.* **sprain** (5, 5) **center** (0, 0);

update a.* **shift** (select ...);

update a.* **rotate** (select ...) **center** (select ...);

update a.* **sprain** (select ...) **center** (select ...);

Operations: 2D-comparisons

-- return boolean

... where a.* = b.*;

-- the same dots

... where a.* \sim = b.*;

-- the same area and shape

... where a.* in b.*;

-- b.* contains a.*

... where a.* in= b.*;

-- (a.* in b.*) or (a.* = b.*)

... where a.* in^ b.*;

-- a.* in b.* and touch b.*

... where a.* out^ b.*;

-- a.* not in b.* and touch b.*

... where a.* back in b.*;

-- b.* in a.*

... where a.* back in= b.*;

-- b.* in= a.*

... where a.* back in^ b.*;

-- b.* in^ a.*

~~... where intersects(a.*, b.*);~~

~~-- is already occupied for intersection of 'select'~~

Operations: 2D-calculations

... where `measure(a.*)=5;` -- area

-- return new lines under 2nd level figures of left tree (a.*)

... where a.* `&` b.*; -- common records (lines) in 1st low levels

... where a.* `~&` b.*; -- geometrical overlap of figures (2nd level from bottom)

... where a.* `^` b.*; -- geometrical place of touch of lines (but not cross)

... where a.* `-|` b.*; -- geometrical place of cross of lines (but not touch)

... where a.* `^-|` b.*; -- (a.* ^ b.*) union (a.* -| b.*)

... where a.* `<|` b.*; -- closest point of left tree (a.*) to right tree (b.*)

... where a.* `|>` b.*; -- b.* |> a.*

... where a.* `<->` b.*; -- distance between figures

Special value 'mouse'

```
update point set @coord=mouse; -- value for array[2]  
-- end of line is always in place, where mouse is
```

```
update point set @coord=default;  
-- see 'create table point (coord number array[3] default ... )'
```

2D-triggers

Trigger for shift, spain

```
create trigger t on tab
after  shift
when @offset.x=      -- distance of motion
  and @distab=      -- table name, on which 'tab' is dropped
  and @dispk=      -- value of primary key, on which 'tab' occurs
as begin ...
  ... where @fld=@offset.y;
```

```
create trigger t on tab
after  sprain
when @coef.x= -- is normalized
  and @center.x=
  and @distab=
  and @dispk=
as begin ...
  ... where @fld=@coef.y;
  ... where @fld=@center.y;
```

Trigger for rotation

```
create trigger t on tab
after  rotation
when @angle=           -- angle of rotation
    and @center.x=
    and @distab=       -- table name, on which 'tab' is dropped
    and @dispk=        -- value of primary key, on which 'tab' occurs
as begin ...
    ... where @fld=@angle;
    ... where @fld=@center.y;
```


First object obtains mouse focus

```
create table a (  
  id number,  
  focus – figure 'a' does not obtain focus  
);  
create table b (  
  id number,  
  lnk number references a(@id),  
  focus – figure 'b' does not obtain focus  
);  
create table c (  
  id number,  
  ref number references b(@id),  
  v number references d(@id)  
  focus – figure 'c' does not obtain focus  
);  
create table d ( -- click is here  
  id number,  
  r number references d(@id),  
  x number array[2] dot  
  focus -- point obtains focus  
);
```

```
create table a (  
  id number,  
  focus – figure 'a' does not obtain focus  
);  
create table b (  
  id number,  
  lnk number references a(@id),  
  focus – figure 'b' obtains focus  
);  
create table c (  
  id number,  
  ref number references b(@id),  
  v number references d(@id)  
  – figure 'c' does not obtain focus  
);  
create table d ( -- click is here  
  id number,  
  r number references d(@id),  
  x number array[2] dot  
  – point does not obtain focus  
);
```



Window format,
optionally

Switching and enquiring session parameter

set ws microsoft; -- to Microsoft window system

set ws x11; -- to X11 window system

show ws; \Longrightarrow <?res code=0 ws=x11 /?>

Mixture of dimensions

3D point with 2D object

If 3D point refers to 2D figure (including 2D line itself), than 2D figure is always displayed in plane, perpendicular to look from camera

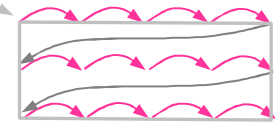
```
create table 3Dpoint (coord number array[3] dot,  
                    fig    number references 2Dfig(@id),  
                    note  varchar);  
create table 2Dfig  (id    number,  
                    line  number references 2Dline(@id),  
                    note  varchar);  
create table 2Dline (id    number,  
                    next  number references 2Dline(@id),  
                    coord number array[2] dot,  
                    note  varchar);
```

The same is for *text or blob* field, marked by 'incamera', in 3D point

```
create table 3Dpoint (coord number array[3] dot,  
                    note  varchar incamera);
```

3D triangle/quadrangle with 2D object

If 3D triangle (or quadrangle, i.e. left top angle of quadrangle) refers to 2D figure (including 2D line itself), than 2D figure is displayed in plane, perpendicular to look from camera (if referring field is marked by 'incamera'), or in plane of triangle/quadrangle (if is marked by 'infacet')



```
create table 3Dtriangle (vertex number array[3] references 3Dpoint(@id),
                        fig      number  infacet  references 2Dfig(@id) );
create table 3Dpoint   (id      number,
                        coord  number array[3] dot);
create table 2Dfig     (id      number,
                        line   number          references 2Dline(@id) );
create table 2Dline   (id      number,
                       next   number          references 2Dline(@id),
                       coord  number array[2] dot);
```

3D triangle/quadrangle with text

The same is for *text or blob* field, marked by
'incamera' or by 'infacet', in 3D triangle/quadrangle

```
create table 3Dtriangle (vertex number array[3] references 3Dpoint(@id),
                        note varchar infacet);
create table 3Dpoint (id number,
                    coord number array[3] dot);
```

Fielders, tablers

Fielders mark columns

```

create table fig (id num,
color num array[3] color,
offset num array[3] shift,
turn num array[3] euler,
dist num array[2] span);

create table fig4 (fg num references fig(@id),
range num array[2] span,
time time array[2] period,
side num references quadr(@id),
pic blob texture,
d num diffuse,
rf num reflect,
rd num radiate,
focus);

create table quadr (id num,
next num references quadr(@id),
vertex num array[4] references point(@id) );

create table fig3 (id num,
conc num references fig(@id),
range num array[2] span,
focus);

create table tri (fig num references fig3(@id),
vertex num array[3] references point(@id) );

create table point (id num,
coord num array[3] dot,
time time array[2] period);

```

-- where **user_fielder(@fld)** and ...

-- where **dot(@fld)** and ...

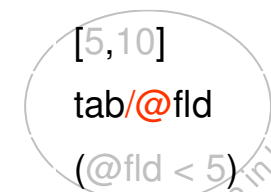
string fielder

fielder name	example of value
constraint	somename
varchar	(30)
default	5
array	[5,10]
refers	tab/@fld
check	(@fld < 5)

boolean fielder

fielder name	example of value
serial, unique, notnull,	true (if present)
notrift, notcross, sole,	false (if absent)
dot, shift, krylov, euler,	
rotation, span, period,	
color, texture,	
diffuse, reflect, radiate,	
infacet, incamera,	
user_fielder	

may be followed



phrase in brackets,
or only one word

Field constraint = domain = fielder

```
create fielder name as ... ;  
create fielder primary as unique notnull; -- fielder is a group of fielders
```

```
create fielder n as float10 diffuse;  
create fielder k as float4 n; -- error: fielder must not contain collisions  
alter table tab alter fld set|drop n; -- set|drop fielder
```

```
create table tab (fld int4 fieldername); -- elementary boolean fielder  
select * from tab where fieldername(@fld);
```

Tablers mark tables

```

create table fig (id num,
                 color num array[3] color,
                 offset num array[3] shift,
                 turn num array[3] euler,
                 dist num array[2] span);

create table fig4 (fg num references fig(@id),
                  range num array[2] span,
                  time time array[2] period,
                  side num references quadr(@id),
                  pic blob texture,
                  d num diffuse,
                  rf num reflect,
                  rd num radiate,
                  focus);

create table quadr (id num,
                   next num references quadr(@id),
                   vertex num array[4] references point(@id) );

create table fig3 (id num,
                  conc num references fig(@id),
                  range num array[2] span,
                  focus);

create table tri (fig num references fig3(@id),
                 vertex num array[3] references point(@id) );

create table point (id num,
                   coord num array[3] dot,
                   time time array[2] period);

```

-- where **focus(tab)** and ...
 -- where **user_tabler(tab)** and ...

string tabler

tabler name	example of value
constraint	somename
unique	(@fld1, @fld2)
foreign	(@fld1, @fld2)
refers	(tab[@fld1, @fld2])
check	(@fld+@fld2<10)

phrase in brackets,
or only one word

boolean tabler

tabler name	example of value
focus, user_tabler	true (if present) false (if absent)

Table constraint = tabler ≠ composed data type

```
create tabler name as ... ;  
create tabler f      as focus check(@fld1+@fld2<10); -- tabler is a group of tablers
```

```
create tabler n      as check(@fld1>7 and @fld2>9);  
create tabler k      as f n;                       -- error: tabler must not contain collisions  
alter table tab set|drop n;                         -- set|drop tabler
```

```
create table tab (fld1 int4, tablename); -- elementary boolean tabler  
select * from tab where tablename(tab);
```