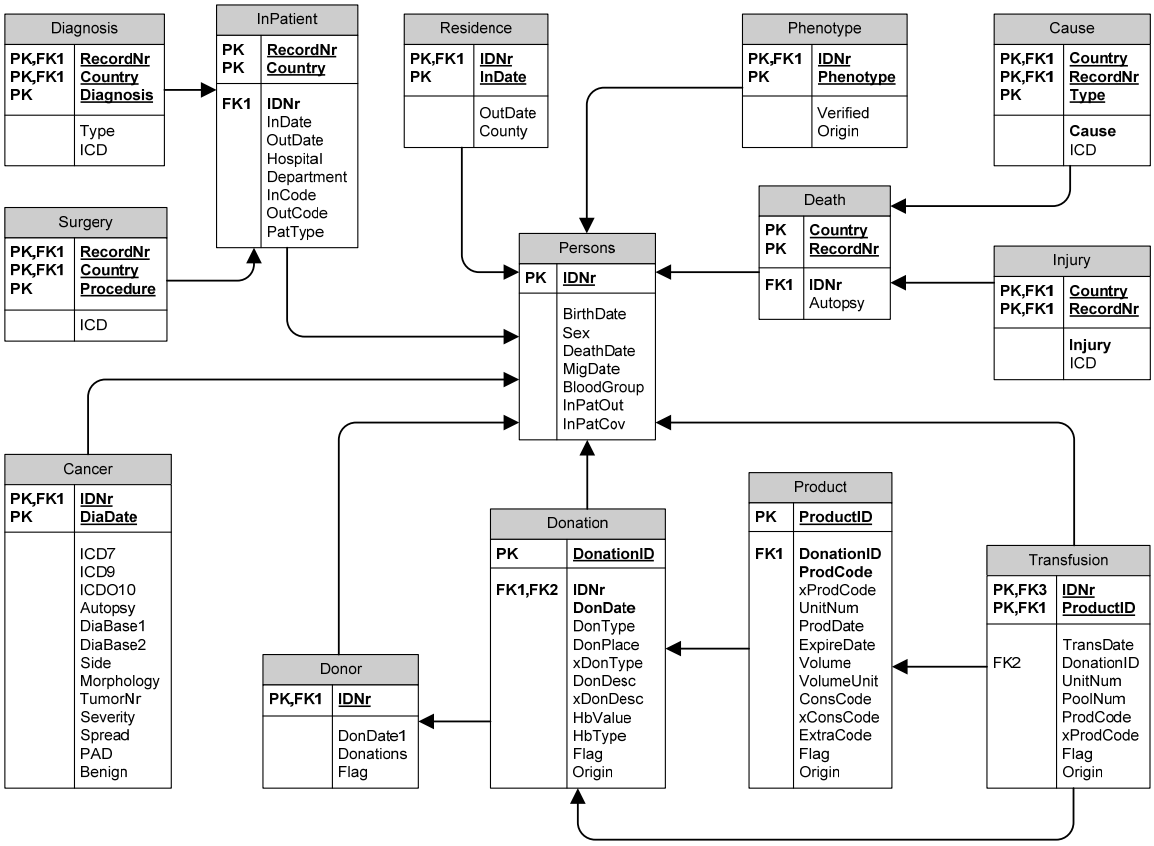
Proc SQL seminar 2006-04-28

Gustaf Edgren
gustaf.edgren@ki.se
Department of Medical Epidemiology and Biostatistics
Karolinska Institutet

Example database



SQL syntax

Selecting and subsetting data

```
proc sql;
   create table tablename as
   select [distinct]
      column1,
      column2,
      [*]
   from library.table
   where condition
   order by column1;
quit;
```

Selecting and modifying data

```
proc sql;
   create table tablename as
   select
      function(column1) as new1,
      column2 [+|-|*|/] column3 as new2
   from library.table;
quit;
```

Summary functions

```
proc sql;
   create table tablename as
   select
      column1,
      summaryfunction(column2) as new
   from library.table
   group by column_list
   having group_conditions;
quit;
```

Examples of commonly used summary functions are:

- mean( ) =selects the mean within the group as defined by group by statement
- count() =counts observations within each group as defined by group by statement
- min() =selects the minimum value of column within group
- max() = selects the maximum value of column within group

Unfortunately no median function exists, so sometimes proc summary is a must…

Joining/merging tables

```
proc sql;
   create table tablename as
   select
      alias1.column1 as new1,
      alias2.column1 as new2
   from library.table1 as alias1
      [inner|outer|left|right] join
        library.table2 as alias2
   on join_clause;
quit;
```

Common types of joins:

- inner join =pick observations from both tables only where both tables satisfy the join clause
- outer join =pick observations from whatever table satisfies the join clause
- left join =always pick observations from the first table and from the second table whenever it satisfies the join clause
- right join =reverse of left join

Using the pass-through facility for querying databases

```
proc sql;
   connect to DBMS-name (connection statements);
   select
      column_list
   from connection to DBMS-name
      (
      DBMS-query
      )
   disconnect from DBMS-name;
quit;
```

The connection statement must consist of:

- DBMS-name = e.g. oracle or access (at MEB, typically oracle)
- path = the address of the database (at MEB, typically store.meb.ki.se)
- user = user name for database
- password = password for database
- Plus a number of other options. See SAS help for reference

The DBMS query must follow the specifications of the database you are working with. Thus you can nolonger use native SAS functions, but there is almost always a DBMS-equivalent to a SAS-function. For a full specification of oracle functions see SQL reference at: http://baldur.meb.ki.se/oracle10g/

Using the pass-through facility for executing commands

```
proc sql;
   connect to DBMS-name (connection statements);
   execute
      (DBMS-commands)
      by DBMS-name;
   disconnect from DBMS-name;
quit;
```

Can be used for executing commands on a server, such as creating indices or removing tables. Should, of course, be used with caution. Perhaps best left to your DBA…

SQL examples

Example 1

```
proc sql;
   create table women as
   select
      *
   from cblood.persons
   where sex=2
   order by birthdate;
quit;
```

Example 2

```
proc sql;
   create table patients as
   select distinct
      idnr
   from cblood.transfusion;
quit;
```

Example 3

```
proc sql;
create table dead as
   select
      idnr,
      (deathdate-birthdate)/365.24 as age
   from cblood.persons
   where not deathdate is null;
quit;
```

Example 4

```
proc sql;
   create table cancers as
   select
      idnr,
      count(*) as cancers
   from cblood.cancer
   group by idnr;
quit;
```

Example 5

```
proc sql;
   create table unlucky_few as
   select
      idnr,
      count(*) as cancers
   from cblood.cancer
   group by idnr
   having count(*) > 10;
quit;
```

Example 6

```
proc sql;
   create table unlucky_donors as
   select
      a.idnr
   from cblood.donor as a
      inner join cblood.cancer as b
   on a.idnr=b.idnr
   group by a.idnr
   having count(*) > 10;
quit;
```

Example 7

```
proc sql;
connect to oracle (user=? path=? password=?);
select
  *
from connection to oracle
  (
  select
     extract(year from dondate) as year,
     count(*) as count
   from cblood2.donation
   group by extract(year from dondate)
   );
disconnect from oracle;
quit;
```

Example 8

```
proc sql;
connect to oracle (user=? path=? password=?);
select
   *
from connection to oracle
   (
   select
      a.country,
      b.sex,
      trunc(months_between(a.transdate,b.birthdate)/12) as age,
      count(*) as count
   from cblood2.transfusion a inner join cblood2.persons b
   on a.idnr=b.idnr and b.birthdate <= a.transdate
   where extract(year from a.transdate) between 1968 and 2002
      and instr(coalesce(b.flag,' '),'ID')=0
   group by
      a.country,
      b.sex,
      trunc(months_between(a.transdate,b.birthdate)/12)
   );
quit;
```

Example 9 – selecting controls in a nested case-control study

```
proc sql;
create table potential_controls as
   select
      a.casenr label='',
      a.exit as indexdate,
      b.recipient label='',
      ranuni(2) as randomnr
   from cases a left join acr3 b
   on a.entry-180 le b.entry le a.entry+180
      and a.country=b.country
      and a.county=b.county
      and a.bloodgroup=b.bloodgroup
      and a.age=b.age
      and b.entry < a.exit < b.exit
   where a.county ne .
      and b.county ne .
   order by a.casenr, calculated randomnr;
quit;

data selected_controls;
   set potential_controls;
   by casenr;
   if first.casenr then n=0;
   n+1;
   if n le 3;
   if recipient='' then delete;
run;
```