Data management with MySQL

Msc Modelization

TE 2/11/17

Outline

- Data types and data values
- Introduction to relational algebra
- SQL queries: SELECT
- Joining multiple sources of data to query data relationships

Installing MySQL

login into system

- sudo apt-get install mysql-server

  (Remember to set root password)

- sudo apt-get install mysql-workbench

Using MySQL

```
login into system
open a terminal
Open mysql database session:
```

- `mysql -u root -p`
  `(enter your root password)`

```
mysql>
```

Also, use Mysql workbench tool, connect to mysql (localhost connection)

- `mysql-workbench`

Importing data into MySQL

Download experiments.sql file

```
cd Downloads
```

Open mysql database session:

- `mysql -u root -p`
  (enter your root password)

```
mysql>create database experiments;
Mysql>show databases;
Mysql>quit;
```

Now, import the dataset into the database

```
mysql –u root -p experiments < experiments.sql
```

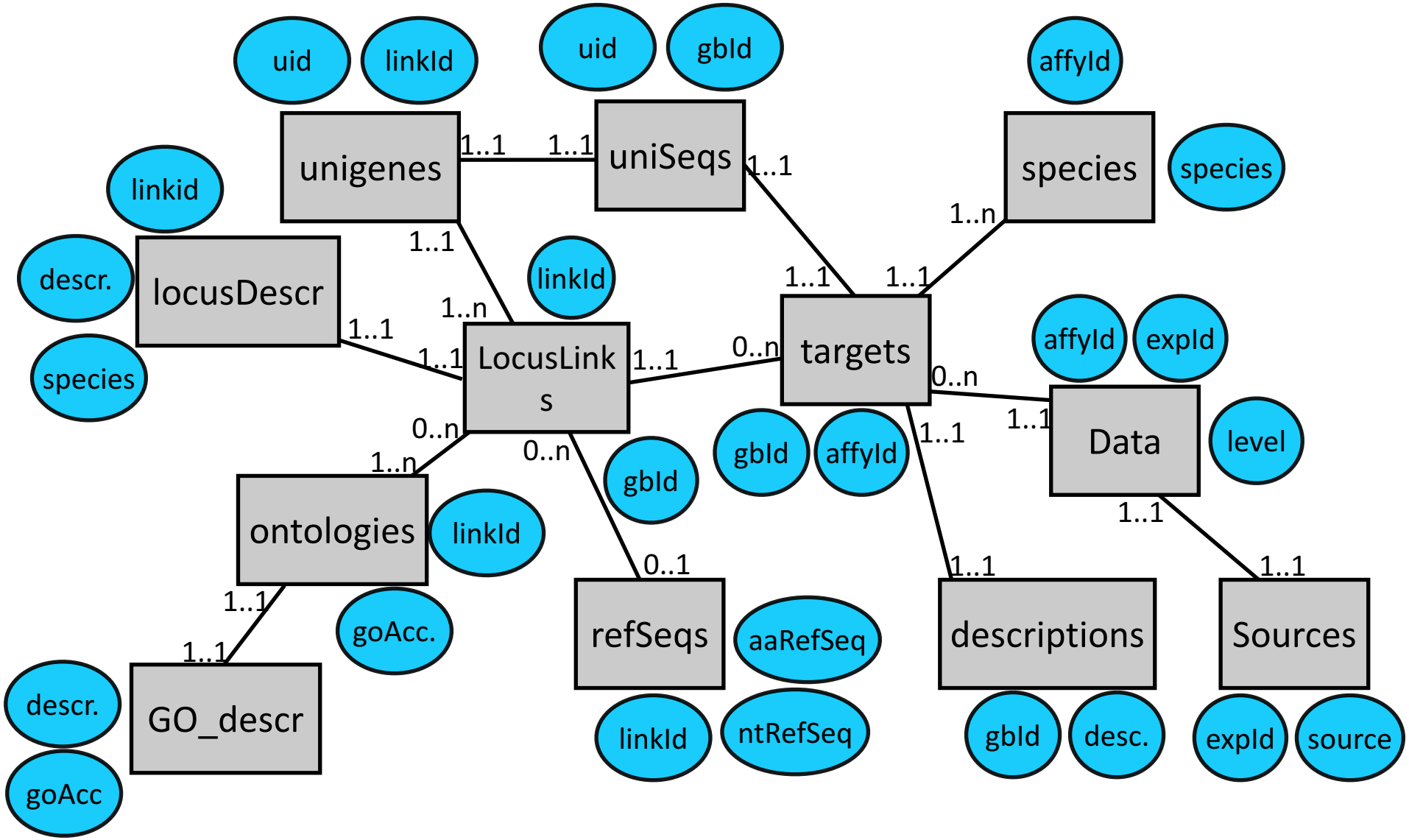
Importing data to MySQL

Open mysql database session:

- `mysql -u root -p`
  (enter your root password)

```
mysql>show databases;
Mmysql>use experiments;
Mmysql>describe data;
```

Experiments database



MySQL data types: numeric

- Integer (INT)
  - Signed -2147483648 to 2147483647
  - Unsigned 18446744073709551615
- Real (FLOAT/DOUBLE)
  - Decimal values, 1.234, 1.47564839E+5
- dates (DATE/DATETIME)
  - ‘YYYY-MM-DD HH:MM:SS’
- TIMESTAMP
  - YYYYMMDDHHMMSS

MySQL datatypes: alphanumeric

- VARCHAR(string length)
  - Stores a string of characters smaller than 255

- TEXT
  - Stores a string smaller than 65535

Getting info on databases

- mysql>show databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| experiments        |
| genbank            |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

- mysql>show tables from experiments;

```
+-----------------------+
| Tables_in_experiments |
+-----------------------+
| Data                  |
| Descriptions          |
| GO_Descr              |
| LocusDescr            |
| LocusLinks            |
| Ontologies            |
| RefSeqs               |
| Sources               |
| Targets               |
| UniDescr              |
| UniSeqs               |
| Unigenes              |
+-----------------------+
```

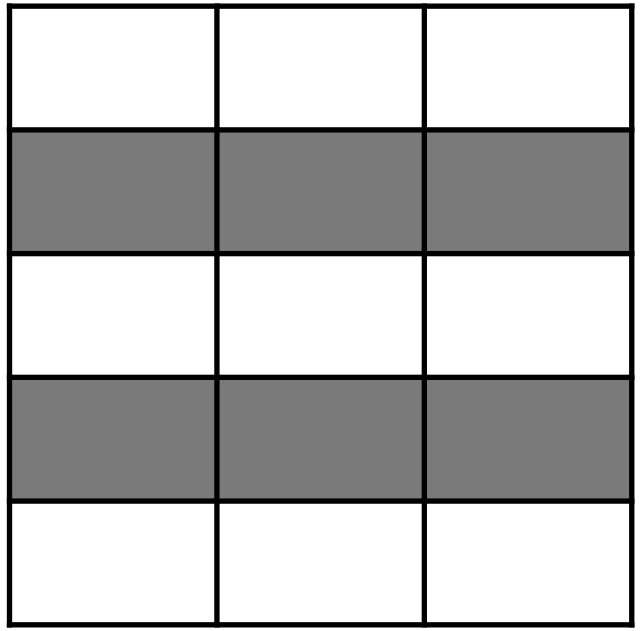
Getting info on tables

- mysql>use experiments;
- mysql>describe Data;

```
+--------+-------------+------+-----+---------+-------+
| Field    | Type      | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| affyId | varchar(30) | NO   | PRI | NULL    |       |
| exptId | varchar(10) | NO   | PRI | NULL    |       |
| level  | int(11)     | NO   |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

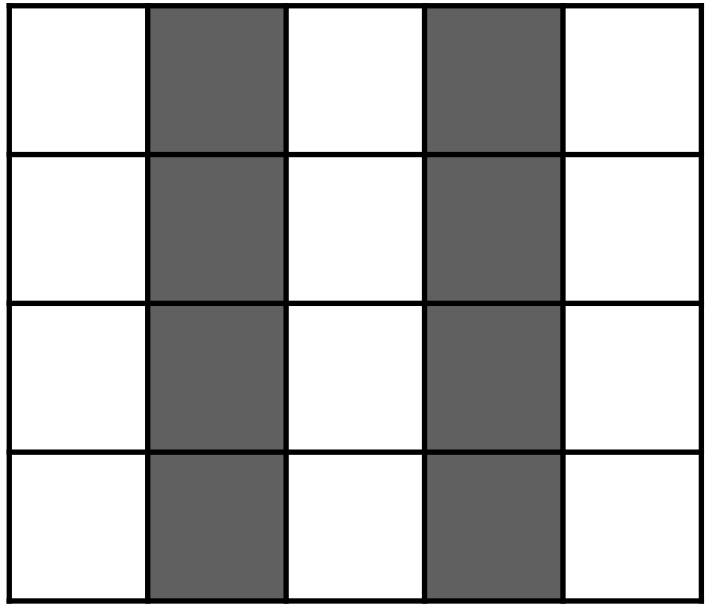
Relational algebra basic operations

- Restrict: eliminate tuples that do not fulfill a specific criteria



Relational algebra basic operations

- Project: eliminate attributes that are not required



Example: Restrict

- SELECT * from targets;

```
+----------+--------------+---------+
| gbId     | affyId       | species |
+----------+--------------+---------+
| AI846313 | 1234_at      | Mm      |
| M18228   | 1235_at      | Mm      |
| X70393   | 1236_at      | Hs      |
| L02870   | U95_32123_at | Mm      |
| S75295   | U95_40474_at | Hs      |
+----------+--------------+---------+
```

- SELECT * from targets where species=“Mm”;

```
+----------+--------------+---------+
| gbId     | affyId       | species |
+----------+--------------+---------+
| AI846313 | 1234_at      | Mm      |
| M18228   | 1235_at      | Mm      |
| L02870   | U95_32123_at | Mm      |
+----------+--------------+---------+
```

Project example

- SELECT gbId from targets;

gbId	affyId	species
AI846313	1234_at	Mm
M18228	1235_at	Mm
X70393	1236_at	Hs
L02870	U95_32123_at	Mm
S75295	U95_40474_at	Hs

gbId
AI846313
M18228
X70393
L02870
S75295

Typical query structure: SELECT

SELECT whatever we need to get

FROM table or tables

WHERE conditions that must be TRUE

Simple query translation

```
SELECT gbId
FROM targets
WHERE species="Mm";
```

- Get all Gene Bank id's
- from table "targets"
- where genes belong to "Mus musculus" species

Simple query exercise

1. Get all data from LocusDescriptions
2. Get all data from LocusDescriptions where locus belongs to Human species (Hs)
3. Get all LocusLinks id's from table "LocusDescriptions" where locus belong to Human species

Using aggregate values on queries

- AVG: Average of attribute values
  - AVG(level)
- COUNT: Count of attribute values
  - COUNT(affyId)
- MAX: Maximum value of attributes
  - MAX(level)
- MIN: Minimum value of attributes
  - MIN(species)
- SUM: Total sum of attribute values
  - SUM(level)

Work!

1. Get all rows from table Data
2. Get level values from table Data
3. Get average, maximum, minimum expression level from Data

Use of DISTINCT in attributes

- SELECT count(affyId) from data

```
+----------------+
| count(affyId)  |
+----------------+
|             37 |
+----------------+
```

- SELECT count(**DISTINCT** affyId) from data

```
+----------------+
| count(affyId)  |
+----------------+
|             23 |
+----------------+
```

why less?

Using arithmetic expressions

- Get measured expression level and doubled value in the same query
- First: get all experimental data: SELECT * from data

```
+--------------+--------+-------+
| affyId       | exptId | level |
+--------------+--------+-------+
| U95-32123_at | 1      |   128 |
| U95-32123_at | 2      |   128 |
| U98-40474_at | 1      |    57 |
| U98-40474_at | 2      |    57 |
+--------------+--------+-------+
```

- Then, create your needed columns: SELECT level, **level*2 FROM data**

```
+-------+----------+
| level | level*2  |
+-------+----------+
|   128 |      256 |
|   128 |      256 |
|    57 |      114 |
|    57 |      114 |
+-------+----------+
```

Specifying conditions at where

- Expressions must be TRUE

```
mysql> select * from descriptions where description="Glucan";
+--------+-------------+
| gbId   | description |
+--------+-------------+
| S75295 | Glucan      |
+--------+-------------+
```

- Use comparisons of text or numeric values

```
mysql> select * from data where exptId != 1;
+---------------+--------+-------+
| affyId        | exptId | level |
+---------------+--------+-------+
| U95-32123_at  | 2      |   128 |
| U98-40474_at  | 2      |    57 |
+---------------+--------+-------+
```

not equal

Work: specify conditions

1. Get all expression data on Affymetrix Bio Probes: “AFFX-BioB-3_at”
2. Get all expression data on Affymetrix Bio Probes: “AFFX-BioB-3_at” AND “AFFX-BioB-5_at”
3. Get all expression data on Affymetrix Bio Probes: “AFFX-BioB-3_at” AND “AFFX-BioB-5_at” AND “AFFX-BioB-M_at”

Pattern matching with LIKE

```
mysql> select * from data where affyId LIKE "AFFX-BioB%";

+---------------+--------+-------+
| affyId        | exptId | level |
+---------------+--------+-------+
|AFFX-BioB-3_at| 3      |    97 |
|AFFX-BioB-5_at| 3      |    20 |
+---------------+--------+-------+
```

Sorting results with ORDER BY

```
mysql> SELECT * FROM refseqs
    -> where linkId like "105%"
    -> ORDER BY linkId DESC;
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
| 105910 | NM_134094 | NP_598855 |
| 105892 | NM_128276 | XP_128276 |
| 105887 | NM_127943 | XP_127943 |
| 105870 | XM_128254 | XP_128254 |
```

Sorting results with ORDER BY

```
mysql> SELECT *
    -> FROM refseqs
    -> WHERE linkId LIKE "105%"
    -> ORDER BY aaRefSeq ASC;
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
|   1057 | NR_001275 |           |
|   1053 | NM_001805 | NP_001796 |
|   1054 | NM_001806 | NP_001797 |
|   1056 | NM_001807 | NP_001798 |
|   1058 | NM_001809 | NP_001800 |
```

Work: ORDER BY

1. Get all values of table Data of experiment ‘3’ sorted by expression level

Asking for more WHERE conditions

```
mysql> SELECT affyId,level
    -> FROM data
    -> WHERE level BETWEEN 80 AND 100
    -> LIMIT 5;
+---------------------------+--------+
| affyId                    | level  |
+---------------------------+--------+
| 31324_at                  |     91 |
| 31356_at                  |     91 |
| AFFX-BioB-3_at            |     97 |
| AFFX-HSAC07/X00351_M_at   |     86 |
| AFFX-HUMTFFR/M11507_at    |     90 |
+---------------------------+--------+
```

Asking for more WHERE conditions

```
mysql> SELECT *
    -> FROM uniseqs
    -> WHERE gbId NOT LIKE "NM_%"
    -> LIMIT 5;
+--------+---------+
| uId    | gbId    |
+--------+---------+
| Hs.2   | D90042  |
| Hs.11  | D90278  |
| Hs.11  | L00693  |
| Hs1640 | L02870  |
| Hs.21  | M16652  |
+--------+---------+
```

Querying expression levels with WHERE

```
mysql> SELECT * FROM data
    -> WHERE level BETWEEN 80 AND 100 OR level < 21
    -> LIMIT 5;
+----------------+--------+-------+
| affyId         | exptId | level |
+----------------+--------+-------+
| 31324_at       | 3      |    91 |
| 31356_at       | 3      |    91 |
| AFFX-BioB-3_at | 3      |    97 |
| AFFX-BioB-5_at | 3      |    20 |
| AFFX-BioB-M_at | 3      |    20 |
+----------------+--------+-------+
```

Querying expression levels with WHERE

```
mysql> SELECT affyId, level
    -> FROM data
    -> WHERE exptId != "hs-cer-1"
       AND level BETWEEN 250 AND 300
    -> LIMIT 5;
+----------------------------+--------+
| affyId                     | level  |
+----------------------------+--------+
| 31315_at                   |    250 |
| 31362_at                   |    260 |
| 31510_s_at                 |    257 |
| AFFX-HUMBAPDH/M33197_3_st  |    277 |
| AFFX-M27830_3_at           |    271 |
+----------------------------+--------+
```

Joining data from two tables

- Get the result from two matching tables and then look for tuples that match a specific condition

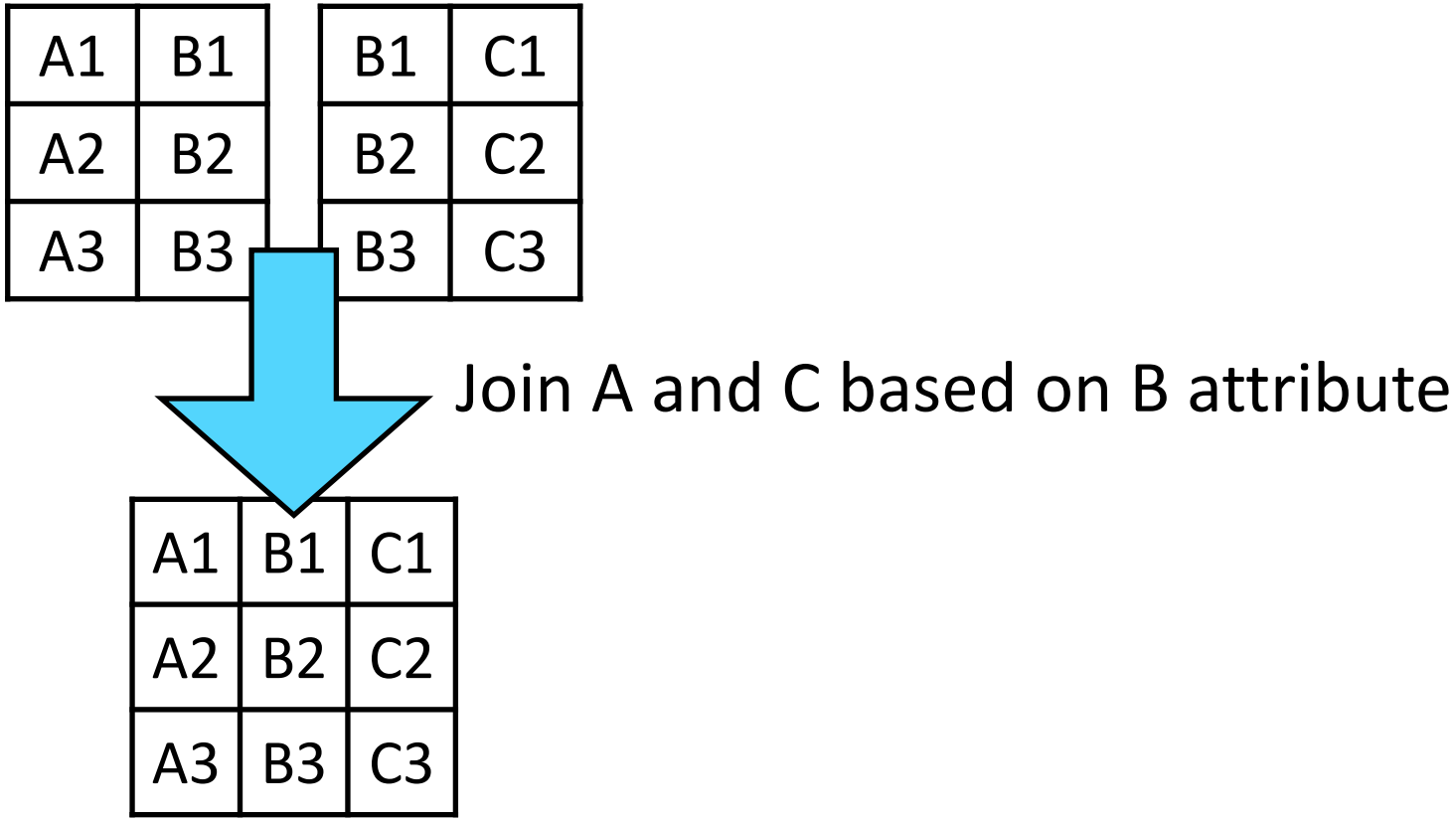


Table Joins

How to get gene ontology descriptions for linkID=4017?

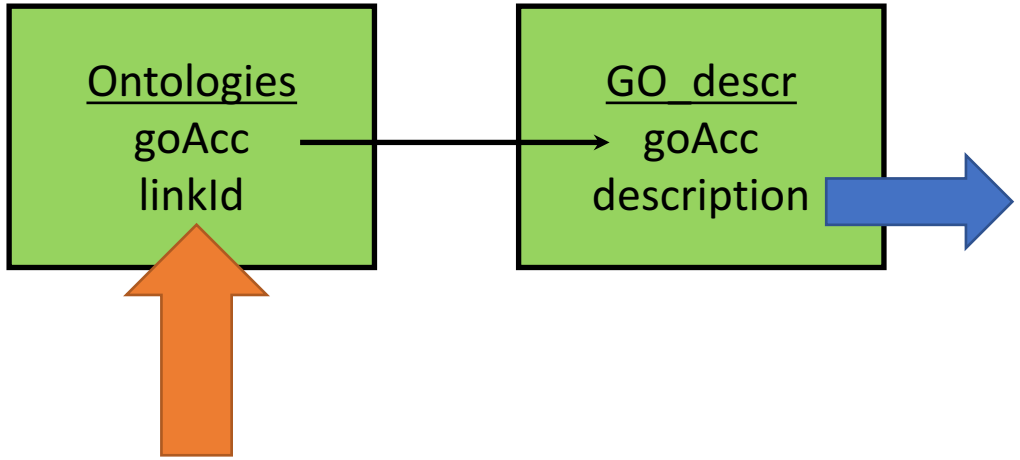
1. Get accession number for my link id=4017 -->ontology table

```
mysql> select goAcc
    -> from ontologies
    -> where ontologies.linkId=4017;
```

2. Get gene ontology terms for found accesion numbers --> GO_descr table

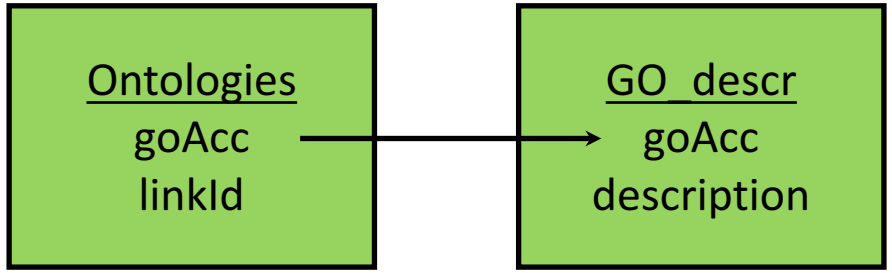
```
mysql> select description
    -> from GO_descr
    -> where GO_descr.goAcc= ?;
```

Get GO descriptions for linkIds



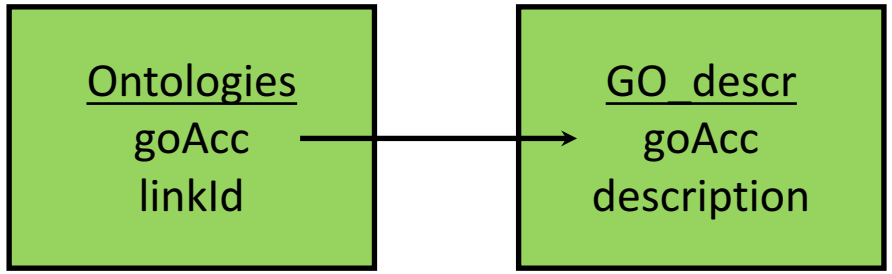
Joining ontologies and go_descr

```
mysql> select go_descr.description,
              ontologies.linkId
    -> from go_descr, ontologies
    -> where ontologies.goAcc=go_descr.goAcc
       and ontologies.linkId=4017;
```



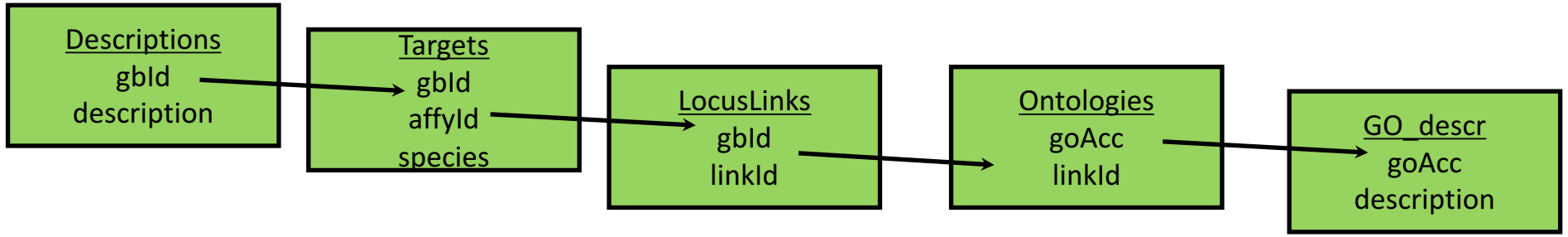
Query all linkIds for GO descriptions

```
mysql> select go_descr.description, ontologies.linkId
    -> from go_descr, ontologies
    -> where ontologies.goAcc=go_descr.goAcc;

+-----------------------------------------+--------+
| description                             | linkId |
+-----------------------------------------+--------+
| Serine Prot.                            |   1294 |
| Glucan Enz                              |   2632 |
| fructose-2, 6-biophosphatase 2-phosphatase |   4015 |
| regulation of mitosis                   |   4016 |
| protein kinase                          |   4017 |
| protein kinase                          |   4018 |
| extracellular space                     |   4019 |
+-----------------------------------------+--------+
```



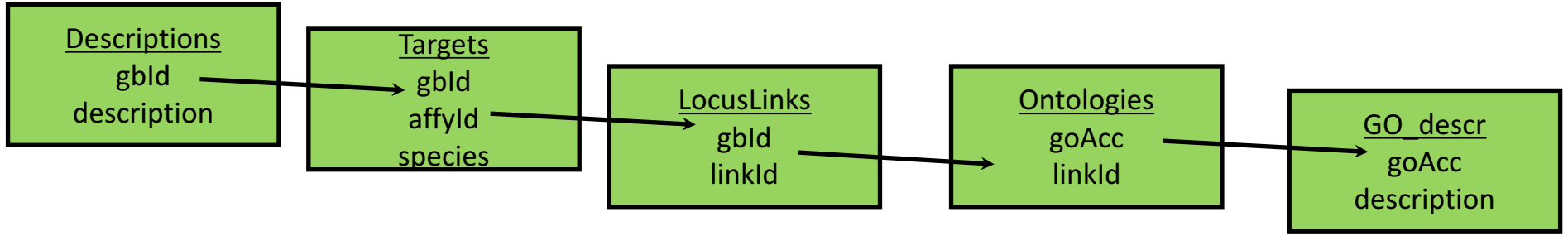
Work:

Translate this schema to a mysql query:



Join example

```
mysql> SELECT descriptions.description AS gene_description,
               go_descr.description AS GO_description
               FROM descriptions, go_descr, locuslinks, ontologies, targets
               WHERE descriptions.gbId=targets.gbId AND
               targets.gbId=locuslinks.gbId AND
               locuslinks.linkId=ontologies.linkId AND
               ontologies.goAcc=go_descr.goAcc
               LIMIT 5;

+---------------------------------------------+-----------------------------------------+
| gene_description                            | GO_description                          |
+---------------------------------------------+-----------------------------------------+
| HSLFBPS7 Human fructose-1, 6-biphosphatase  | fructose-2, 6-biophosphatase 2-phosphatase |
| HSU30872 Human mitosin mRNA                 | regulation of mitosis                   |
| HSU33052 Human lipid-activated protein kinase | protein kinase                          |
| HSU33053 Human lipid-activated protein kinase | protein kinase                          |
| Human clone lambda 5 semaphorin mRNA        | extracellular space                     |
+---------------------------------------------+-----------------------------------------+
```



Natural joins to get data from tables

- Take profit from tables relationships by traversing the database schema
- SELECT and JOIN refer to more than one table
- WHERE uses common attributes to define expressions that must be true

Work!

- Join locus descriptions and locus links using linkId
  - show: descriptions, locuslinks

GROUP BY

- GROUP BY is good for retrieving information about a group of data
- GROUP BY is useful when you have many similar things
- For example, if you have a number of elements of the same type, and you want to find out some statistical information like the minimum, maximum, or other top-level info

GROUP BY example

- Apply group by on data table
- We want information on the expression values of the experiments:
  - aggregate by experiment
  - get information on expression value

GROUP BY aggregations

```
SELECT exptId, MIN(level)
FROM   data
GROUP BY exptId;
```

exptId	MIN(level)
1	57
2	57
3	8
4	51
5	8
6	4
7	20
8	40
9	20

GROUP BY rules

- The column that you GROUP BY must also be in your SELECT statement.
- Remember to group by the column you want information about and not the one you are applying the aggregate function on.
- In our above example we wanted information on the level and the aggregate function was applied to the experiment ID attribute.

GROUP BY work!

- Find maximum expression value from data table for each experiment

Counting aggregated values

- The COUNT function is an aggregate function that simply counts all the items that are in a group
- One use of COUNT might be to find out how many items of each type there are in the table

How many probes for each experiment?

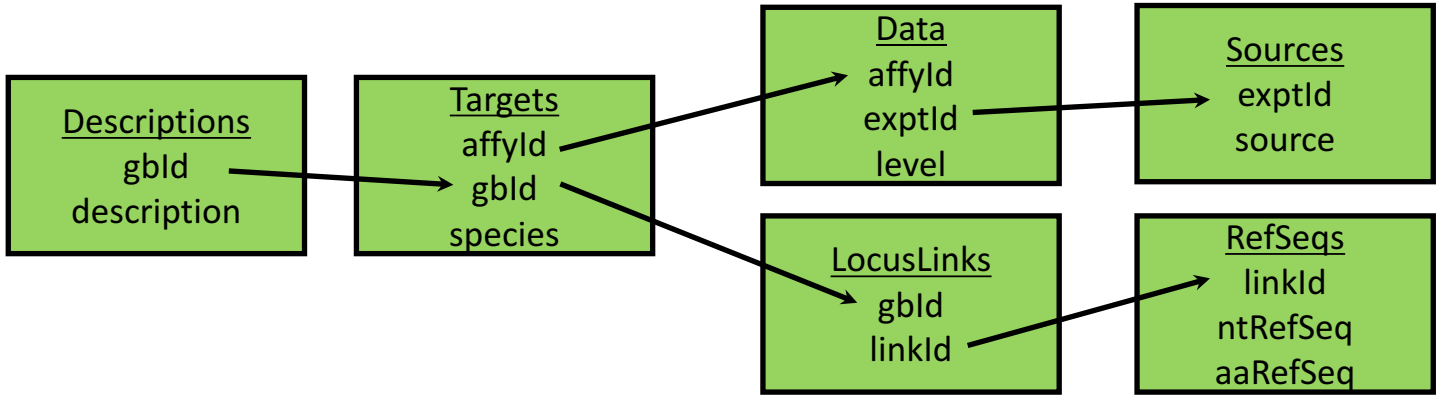
```
SELECT exptId, COUNT(affyId)
FROM   data
GROUP BY exptId;

+--------+---------------+
| exptId | COUNT(affyId) |
+--------+---------------+
| 1      |             3 |
| 2      |             3 |
| 3      |            16 |
| 4      |             6 |
| 5      |             4 |
| 6      |             3 |
| 7      |             1 |
| 8      |             1 |
| 9      |             1 |
+--------+---------------+
```

Count Work!

- Count the number of elements of table **targets** for each individual species

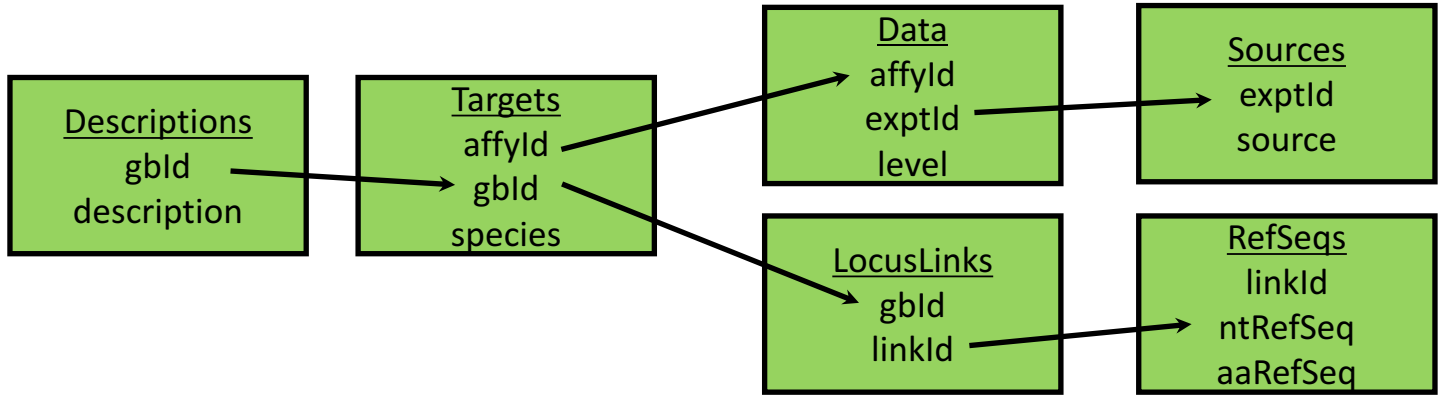
```
+----------+----------------+
| species  | COUNT( )       |
+----------+----------------+
| Hs       |             17 |
| Mm       |              3 |
+----------+----------------+
```

Mega table join



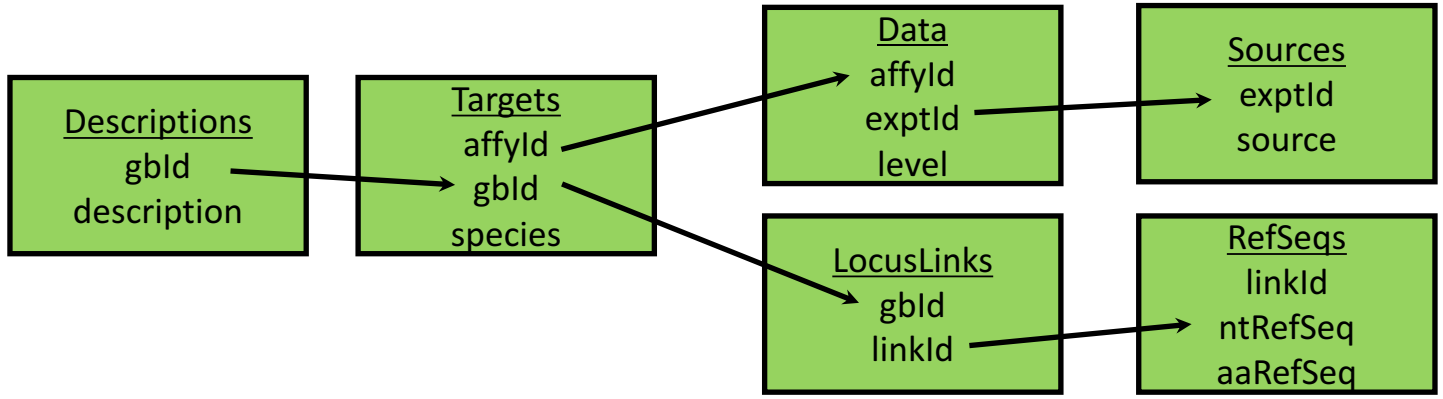
Mega table join

```
SELECT descriptions.description, sources.source, refseqs.ntrefseq
FROM   descriptions, sources, refseqs, targets, locuslinks, data
WHERE  descriptions.gbId=targets.gbId and
       Targets.gbid = locuslinks.gbId
       locuslinks.linkId=refseqs.linkId and
       targets.affyId=data.affyId and
       data.exptId=sources.exptId
```



Mega table join

```
mysql> select descriptions.description, sources.source, refseqs.ntrefseq
          FROM descriptions, sources, refseqs, targets, locuslinks, data
          WHERE descriptions.gbId=targets.gbId and
          Targets.gbid = locuslinks.gbId
          locuslinks.linkId=refseqs.linkId and
          targets.affyId=data.affyId and
          data.exptId=sources.exptId
+-------------------------------------------------------------+-------------+-----------+
| description                                                 | source      | ntrefseq  |
+-------------------------------------------------------------+-------------+-----------+
| Homo sapiens immunoglobulin lambda locus DNA, clone 288A10 | Human Liver | NG_000002 |
| Human rearranged immunoglobulin lambda light chain mRNA    | Human Liver | NG_000002 |
| Homo sapiens immunoglobulin lambda locus DNA, clone 31F3   | Human Liver | NG_000002 |
| Homo sapiens immunoglobulin lambda locus DNA, clone 288A10 | Human Liver | NG_000002 |
| Human rearranged immunoglobulin lambda light chain mRNA    | Human Liver | NG_000002 |
+-------------------------------------------------------------+-------------+-----------+
```



Self join

- Explore data relationships within a table

```
SELECT data1.affyId, data1.exptId AS exptId1, data2.exptId AS exptId2,
       data1.level AS level1, data2.level AS level2
FROM   data data1, data data2
WHERE  data1.affyId=data2.affyId AND
       data1.level>=data2.level*2;
```

affyId	exptId1	exptId2	level1	level2
31325_at	1	4	191	51
31325_at	1	5	191	71
31325_at	1	6	191	31
31325_at	2	6	101	31
31325_at	5	6	71	31