Tutorial:

Building up and querying databases

Objectives

•During this tutorial you will build a custom database on data retrieved from the *BioMart* portal.

•*BioMart* provides a data acquisition and mining platform, suited especially to biological datasets. The *BioMart* homepage links to several data clusters, like *Ensembl Bacteria* and *Ensembl Plant*, which are dedicated to genomic, proteomic and other datasets of the respective organism clades.

Step 1: Gathering data

•We will be focussing on the human genome and choose the ***BioMart*** **central portal** as our starting point (http://www.biomart.org/).

•Our first step will be to retrieve datasets for our self-made database. Let's assume we are especially interested in building a database on:

- 1. The genomic context of all *Ensembl* genes on Chromosome X (ChrX)
- 2. The Gene Ontology (GO) of all Ensembl genes on ChrX
- 3. The splice variants of all ChrX transcripts.

Step 2: Create / fill database

•The second step will take us to a database administration platform, where we will create a database to store, manage and query our retrieved data.

•Since we will use the *BioMart* retrieved data to build a database, we will need to come up with a suitable database scheme.

The ER-scheme

So, before going to the *BioMart* site, take a minute to think about what kind of information you want to store in the database

Hint: You will want to characterize your database entitities sufficiently but exclude unnecessary or redundand information. For example, to characterize a gene you absolutely need to store location information, but GC content is not an essential information you will need to store.

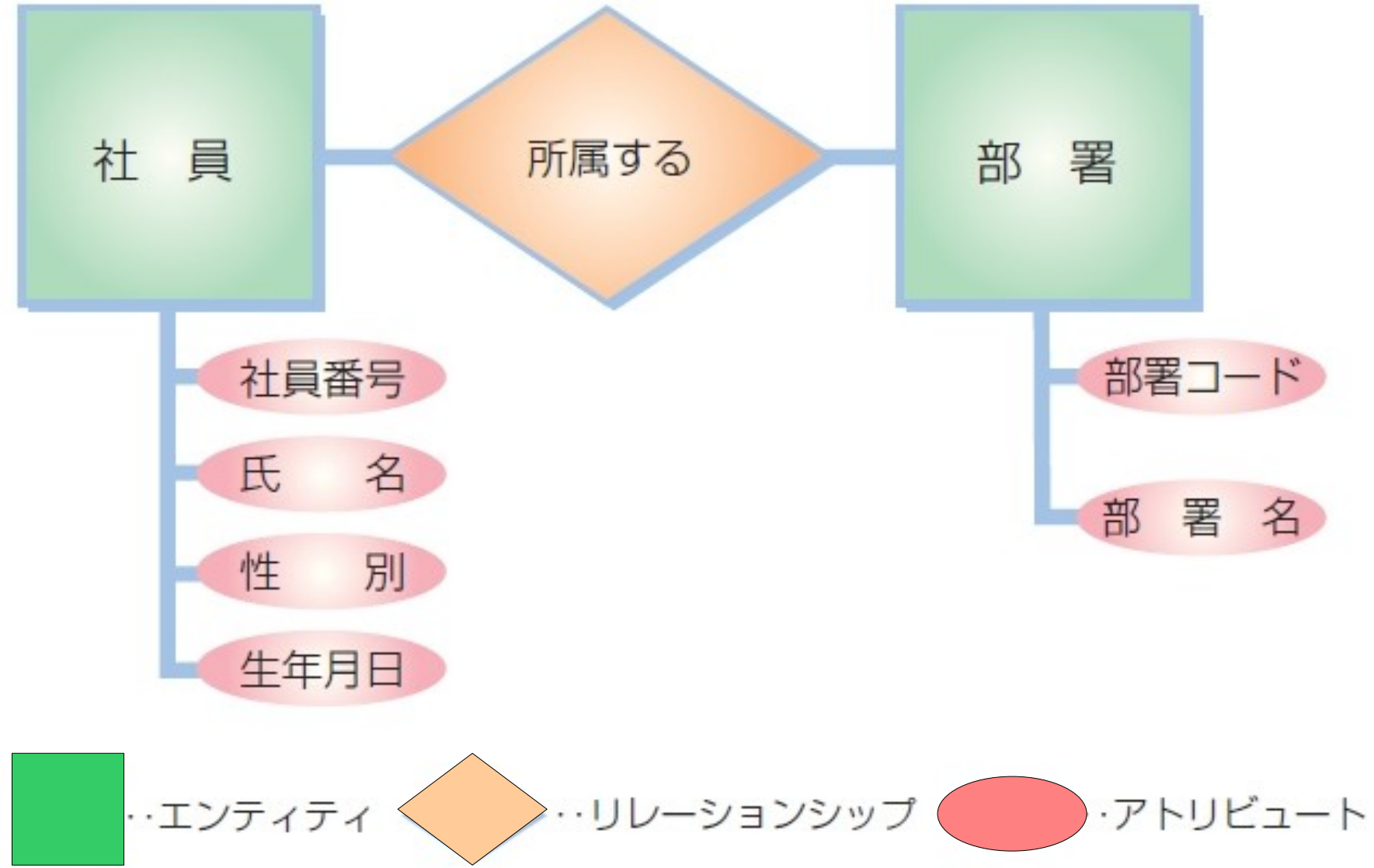
Next think about which entities will carry this information, and how you will be able to relate between these entities.

Hint: Without some kind of ID to characterize entries in your database tables, the entity will be insular, or at least it will be difficult to refer to it from other entities.

Draw on paper an ER (Entity Relation) scheme to visualize the database you are going to build.
This is somehow the main point of database building and for this reason crucial for this tutorial.

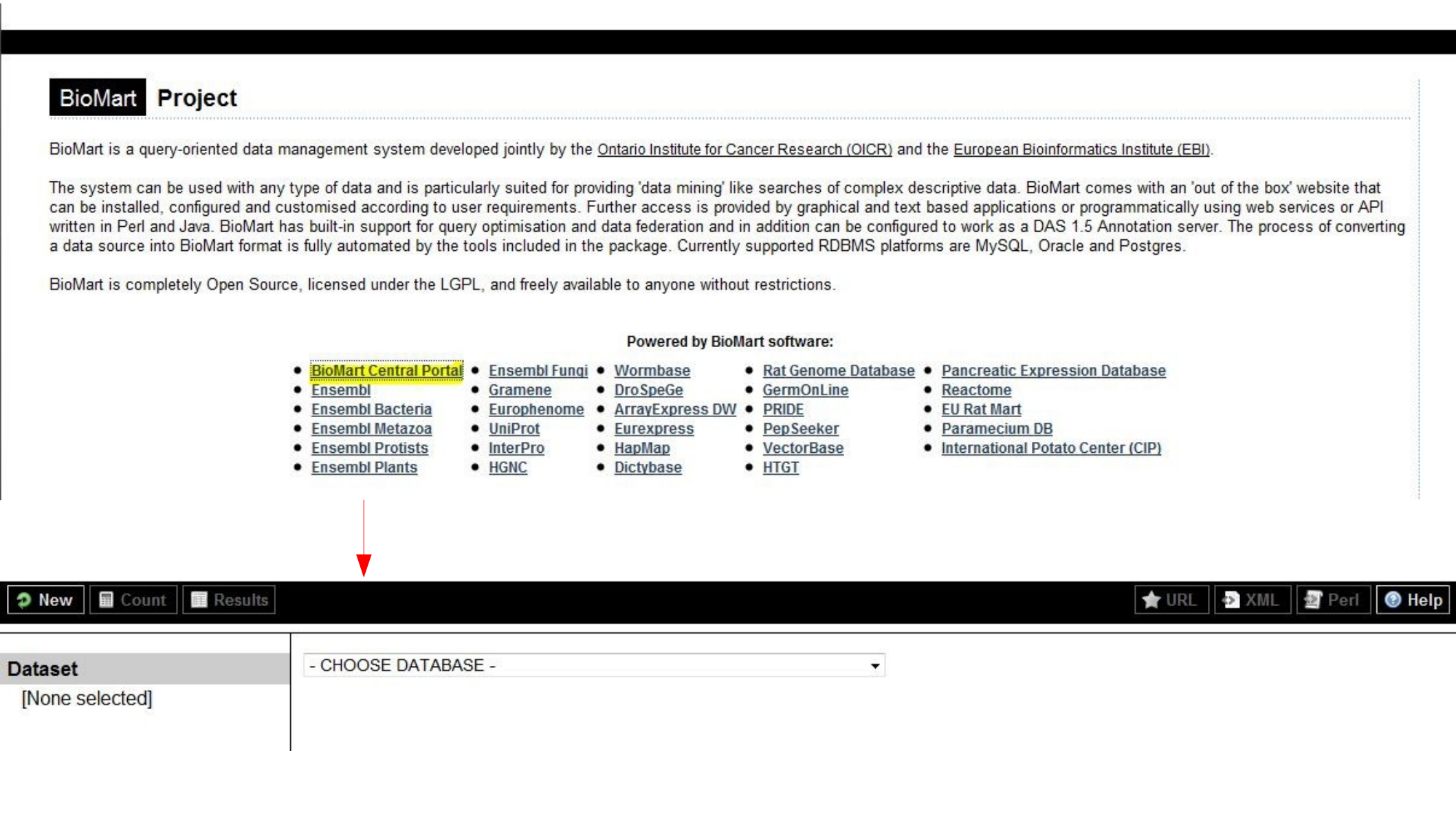
Realtional database scheme

- Think of the different relation-types between the entities in your scheme.
- How can these be modelled in your database, by using tables?
- Surf the www for help. (hint: *cardinality*)


社　員
所属する
部　署
社員番号
氏　　名
性　　別
生年月日
部署コード
部　署　名
··エンティティ
··リレーションシップ
·アトリビュート
Entity
Relation
Attribute

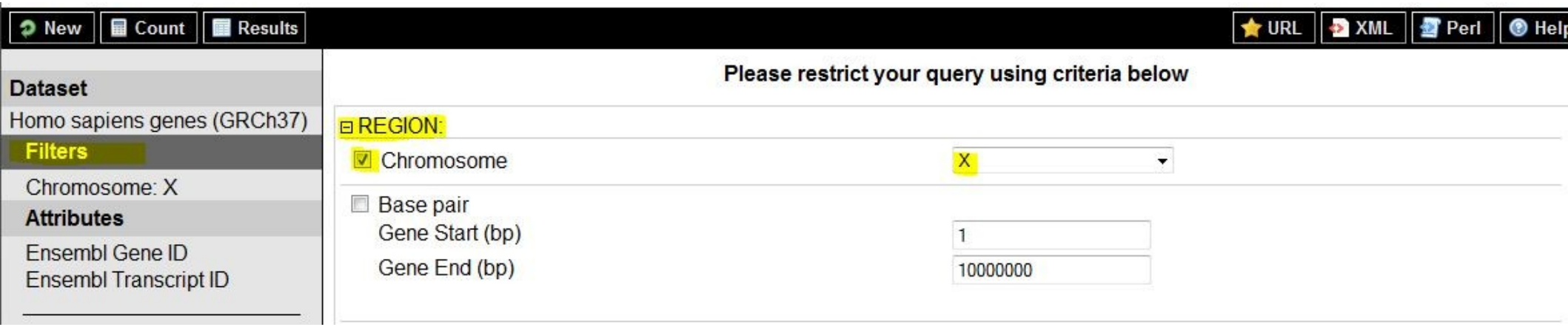
The data retrieval:

Browse the „BioMart Central Portal“



Choose the Ensembl 56 Dataset and the human genome.

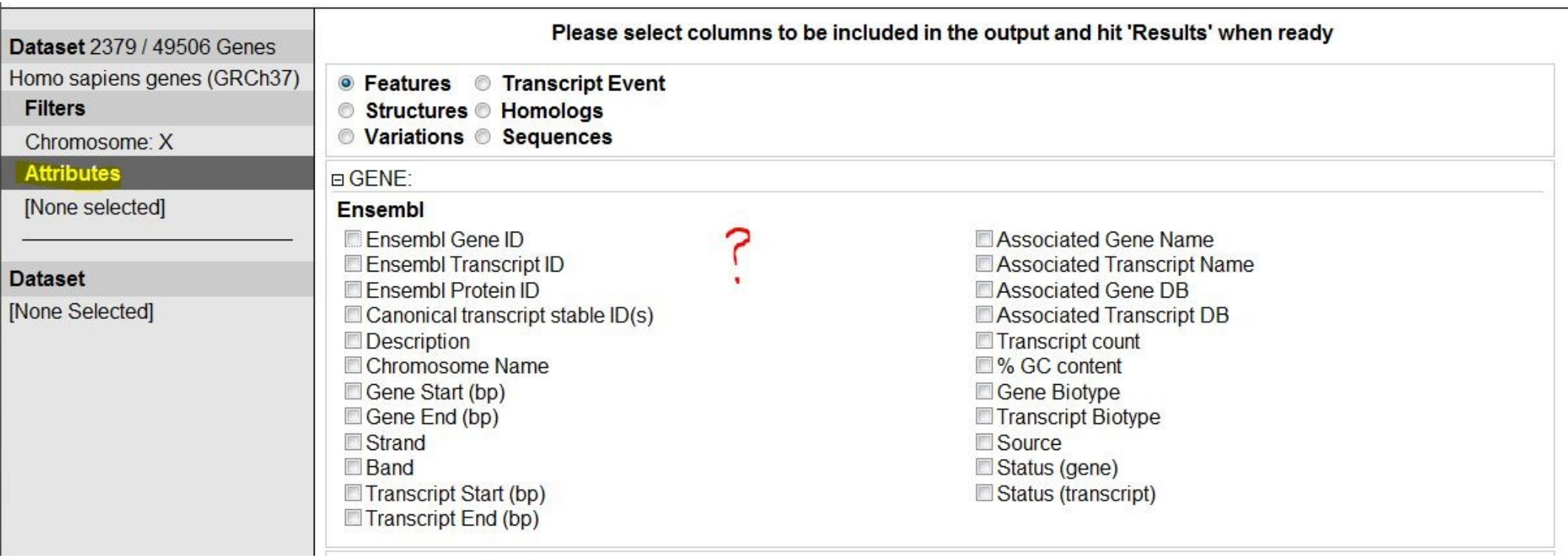
Restrict the region to the X-Chromosome



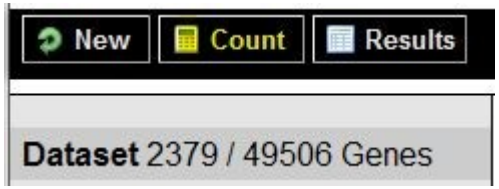
Next select the „GENE“ pop down menu in the „Attributes“ section, under the attribute subcategory „Features“.

Here you will find all downloadable information related to the genes themselves. Select those that you determine necessary to characterize the genes.

Check out the other pop down menus as well and find what information is presented there.



With the count tab at the top right you can get a rough estimate on how large your dataset will be.



The Results tab will retrieve the data according to the selected attributes and filters. It is important to mark the „Unique results only“ check box. Write down how many collumns your table has, whether it contains numbers or letters(+numbers) and how many letters there are in every collumn.

New | Count | Results | URL | XML | Perl | Help

Dataset 2379 / 49506 Genes
Homo sapiens genes (GRCh37)

Filters
Chromosome: X

Attributes
Ensembl Gene ID
Strand
Band
Gene Start (bp)
Gene End (bp)
Chromosome Name

Dataset
[None Selected]

Export all results to File TSV Unique results only Go

Email notification to

View 10 rows as HTML Unique results only

Ensembl Gene ID	Strand	Band	Gene Start (bp)	Gene End (bp)	Chromosome Name
ENSG00000232765	1	p11.1	55306576	55315219	X
ENSG00000196433	1	p22.33	1733894	1761974	X
ENSG00000204131	1	q13.1	71353499	71361212	X
ENSG00000126945	1	q22.1	100663191	100669121	X
ENSG00000241465	1	p11.23	49178558	49223943	X
ENSG00000239469	1	q22.3	107020963	107037689	X
ENSG00000170935	1	q22.3	107037451	107037912	X
ENSG00000240894	1	q22.2	102840506	102841753	X
ENSG00000182484	1	q28	155249967	155255375	X
ENSG00000189108	1	q22.3	103810996	105011822	X

Export the data as a .csv (comma separated value) file – in the compressed form, via the menu at the top.
You might want to rename the file after downloading it.

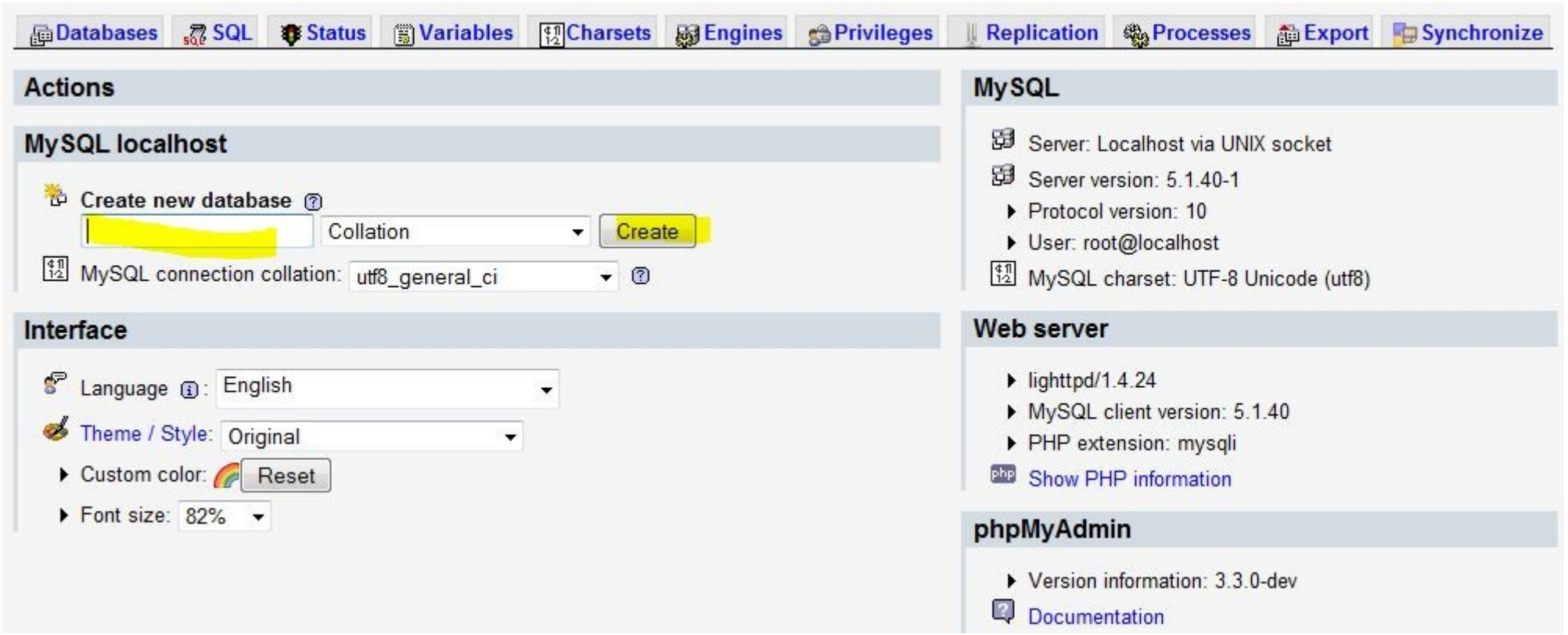
You can find GO data under the „External“ drop down menu under the Features Attributes subcategory. Transcript related data is available directly under *Features->Gene->Ensembl*, just like for the genes themselves.

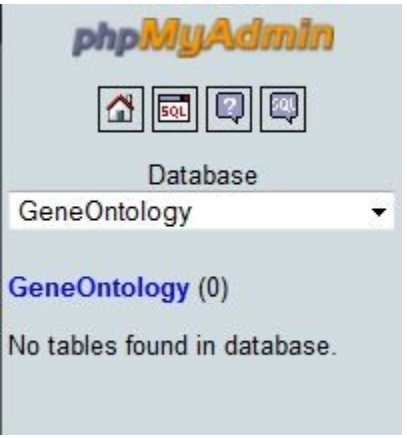
As you assemble your data collection, reconsider which attributes and entities are necessary. Feel free to debate this with others and the tutor – there are many ways to build a database and finding the optimal attribute set/arrangement is not allways easy. It also might depend on the focus of interest...

When you have downloaded all the data you want to integrate in your database you can use **phpmyadmin** to upload the data into the database server and build the database.

Go to http://skinner/phpmyadmin and log in as dbuser01 … dbuser30.
Please ask the tutor which username to take.

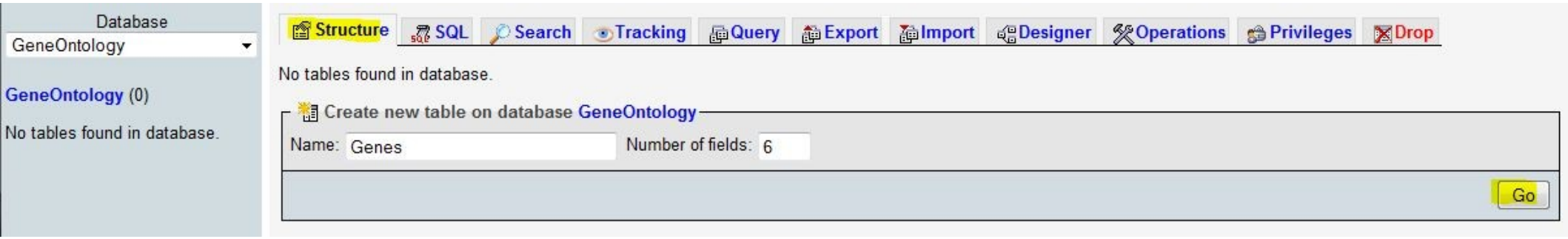
To create a new database enter the name of the database (no empty space or special characters) and click create.
You can change the language of the interface at the bottom left panel





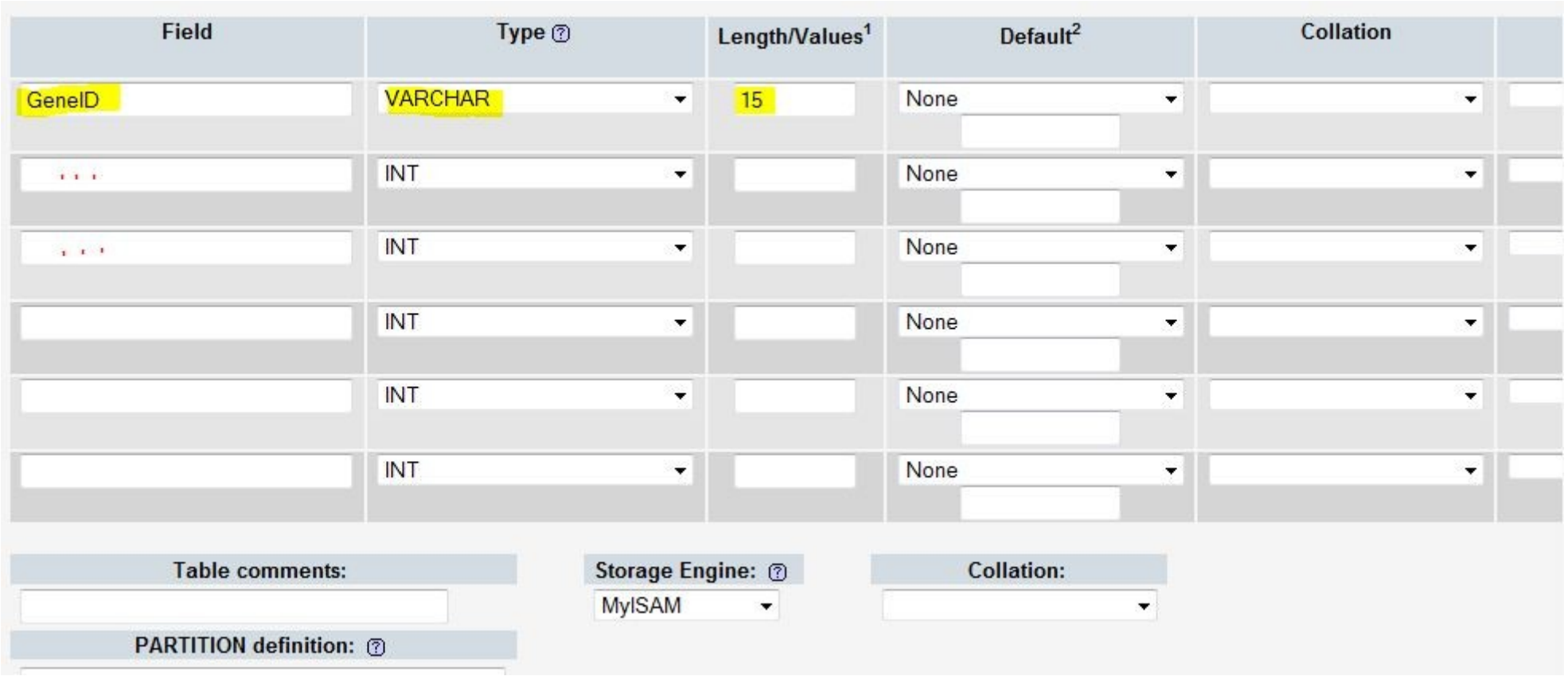
After creating the database you will find it at the top left. The number behind the db name indicates the number of tables. So far the database is empty

To fill it click on the database and the „Structure“ tab. Enter the name of the new table and the number of columns it is going to have. This needs to be exact or the data import will fail.

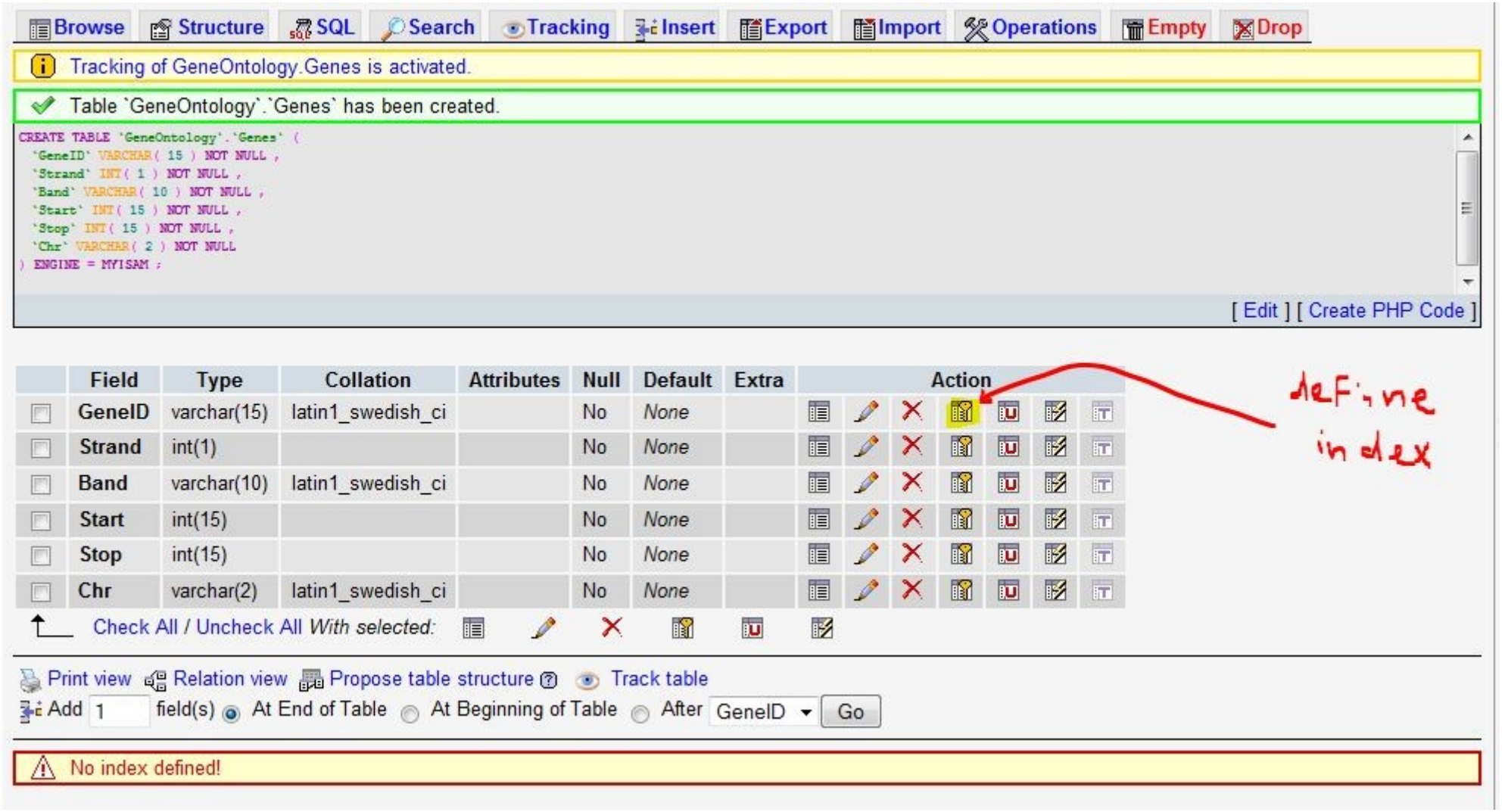


Phpmyadmin will ask you to specify the type of attribute in every column. If the collumn will contain letters only, or numbers as well as letters, choose „VARCHAR“, if it contains a number only, choose „INT“.

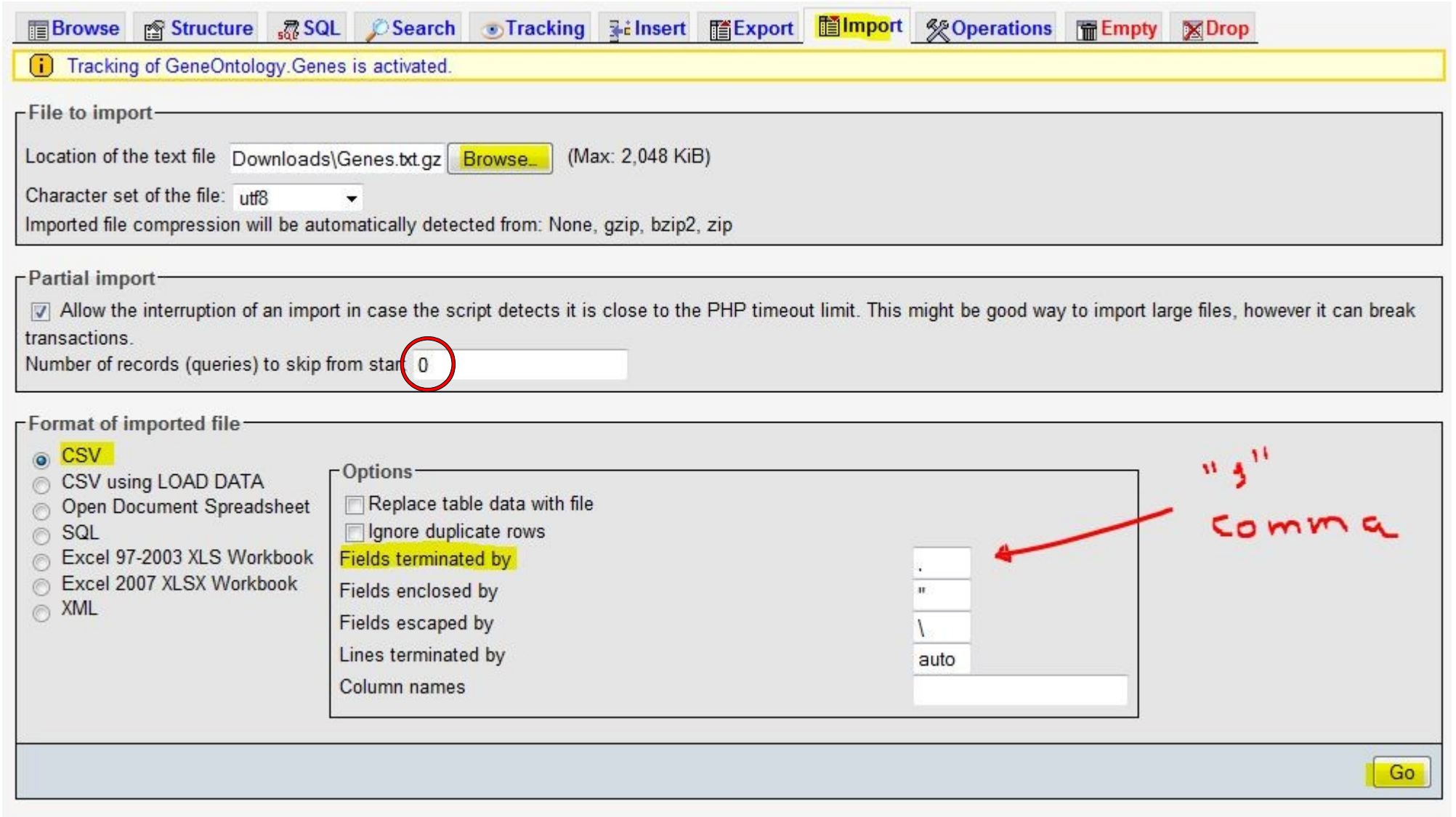
You need to enter a length that is at least as large as the maximum length of attributes in this column.



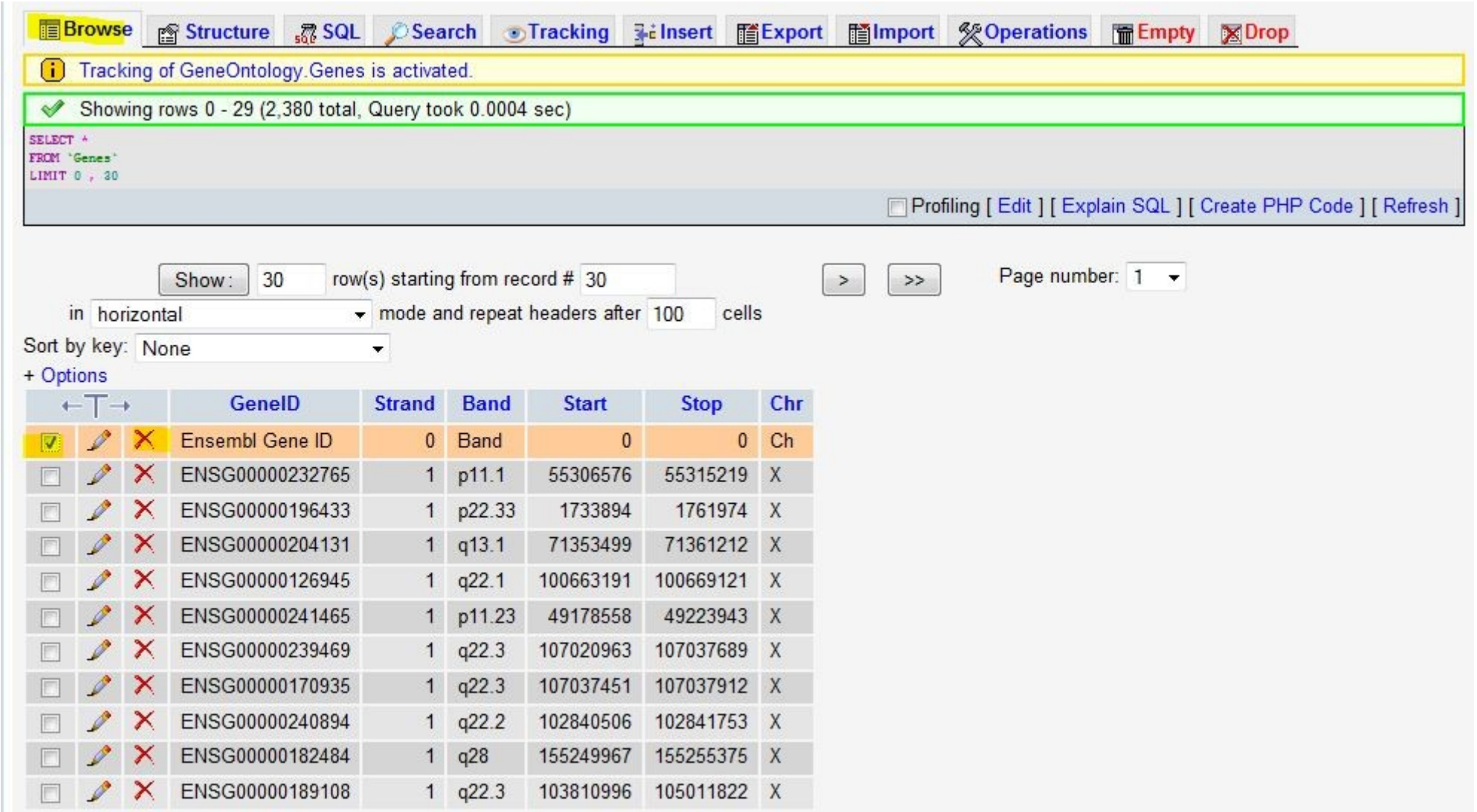
Think about whether you can declare one of the columns to be a primary key. It is advisable that every table has a primary key, if possible. The primary key column can only contain unique values, or the import will fail.



Now the table is ready to be filled with our data. Click on the „import“ tab and upload the first table data. You need to specify the format as *csv* and change the „Fields terminated by“ field to comma (“,“) - since this is what out *csv* is delimited by

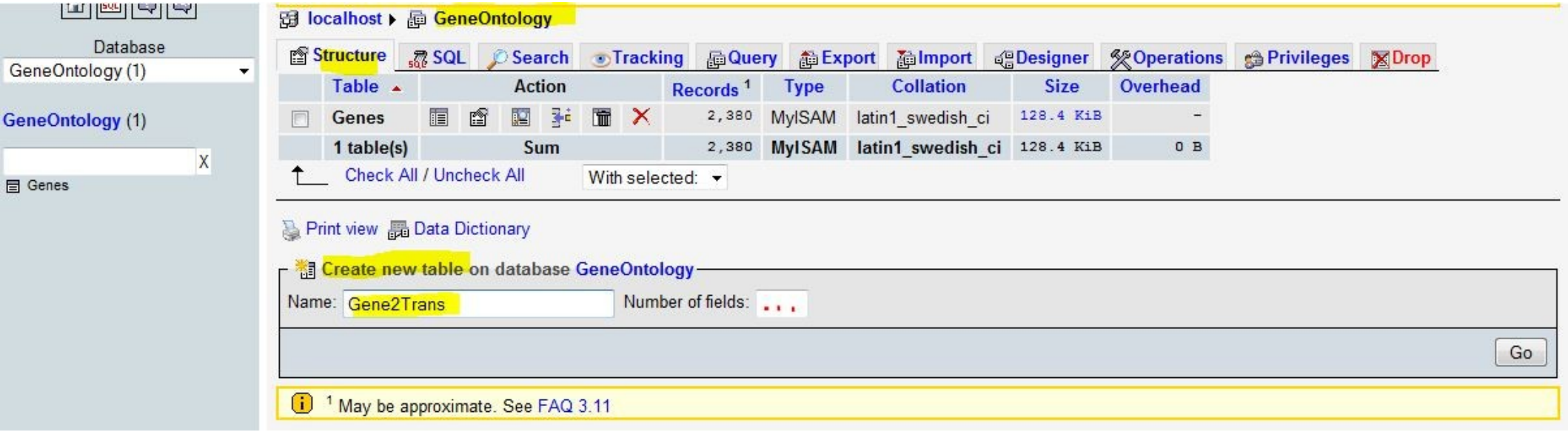


Under „Browse“ you can check whether the import is successfull.
You may need to delete the first row if it does not contain actual attributes but the column names of BioMart. Alternatively u can skip the first line during import (see previous slide/ red circle-> change from '0' to '1').

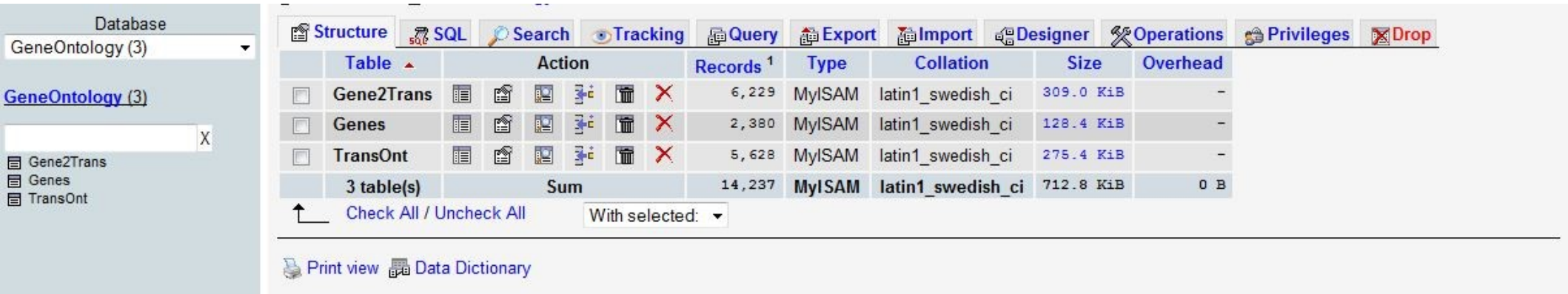


GeneID	Strand	Band	Start	Stop	Chr
Ensembl Gene ID	0	Band	0	0	Ch
ENSG00000232765	1	p11.1	55306576	55315219	X
ENSG00000196433	1	p22.33	1733894	1761974	X
ENSG00000204131	1	q13.1	71353499	71361212	X
ENSG00000126945	1	q22.1	100663191	100669121	X
ENSG00000241465	1	p11.23	49178558	49223943	X
ENSG00000239469	1	q22.3	107020963	107037689	X
ENSG00000170935	1	q22.3	107037451	107037912	X
ENSG00000240894	1	q22.2	102840506	102841753	X
ENSG00000182484	1	q28	155249967	155255375	X
ENSG00000189108	1	q22.3	103810996	105011822	X

Now click on your database (at the top) and add the next table, following the same steps as with the first.



In the end you should have three tables imported



Last, click on the „SQL“ tab and try out some queries as showed below (next slide).

What do these queries achieve?
At least try to understand, please.
Discuss with others or the tutor.

```
SELECT * FROM gene WHERE gene_id = "ENSG00000232765";
SELECT * FROM gene LIMIT 3;
SELECT count(*) FROM gene;
SELECT gene.name FROM gene LIMIT 5;

SELECT count(*) FROM gene, transcript WHERE gene.gene_id =
transcript.gene_id;
SELECT count(*) FROM gene JOIN transcript ON gene.gene_id
=transcript.gene_id;
SELECT transcript.gene_id FROM ...;
SELECT transcript.gene_id, gene.band FROM ...;
SELECT transcript.gene_id, gene.band FROM ... ... WHERE gene.band
="q28";

SELECT gene.name, gene.gene_id, go.go_id, go_term FROM gene, gene2go, go
WHERE gene.gene_id = gene2go.gene_id AND gene2go.go_id = go.go_id;

SELECT g.name,go.go_term FROM gene g JOIN gene2go g2go ON g.gene_id =
g2go.gene_id JOIN go go ON go.go_id = g2go.go_id WHERE go.go_term LIKE
"%immune response%";

SELECT AVG((transcript.trans_end - transcript.trans_start)) FROM
transcript;
```