582685 Project in Biodatabases

Leena Salmela

29.10.2012

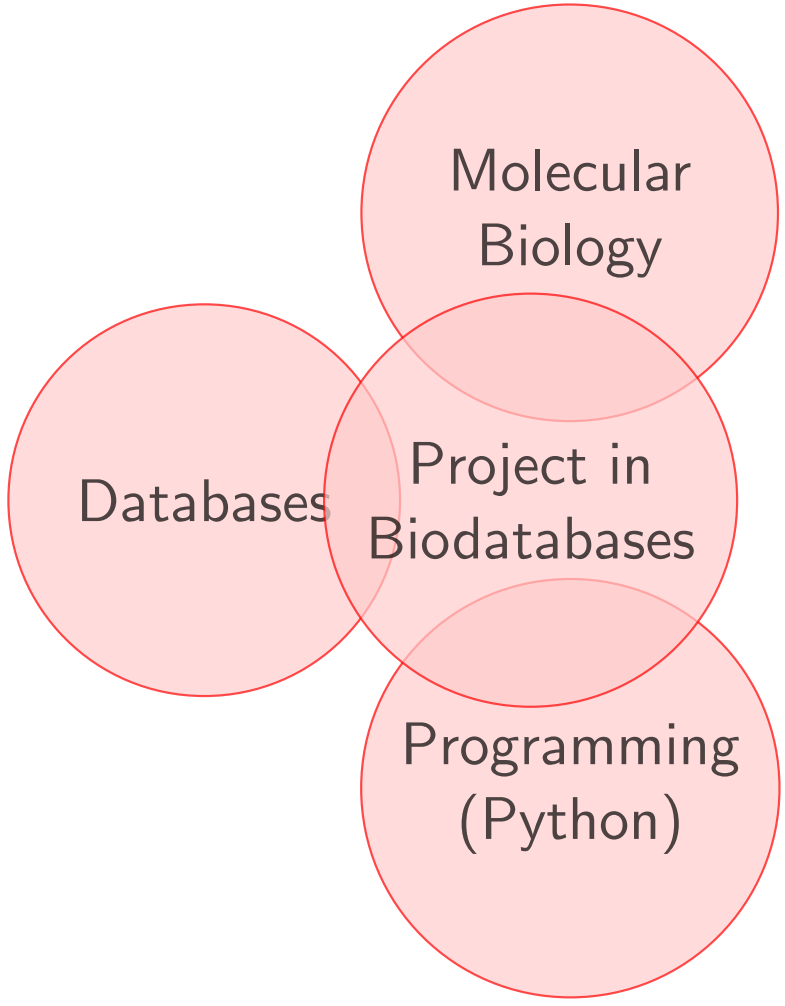
Partly based on slides by Ilari Scheinin / Elements of Bioinformatics 2010

Course Format

- Project work!
- Introductory session: 29.10.2012 (today!)
- Some material to be studied on your own
- Assignments to be completed individually
- Deadline for submitting your answers: 30.11.2012
- Guidance sessions: Mondays 12-14
  - If you have trouble with the assignments, you can come and ask for help.
- Grading: Passed/Failed

Course overview

- Goals:
  - Learn about Ensembl database
  - Programmatic access to databases
  - (Practise programming)
  - (Learn SQL)
- Prerequisites:
  - Basics of SQL
  - Basics of Python programming
  - Basics of molecular biology and bioinformatics (genome structure and alignments)
- Recommended courses taken before the project:
  - Algorithms for Bioinformatics
  - Molecular Biology Reading Group



Overview of the assignments

- Forming SELECT queries to get the needed information out of the database
- Querying the database for genes and computing alignments for the transcripts
- Querying the database for genes and computing motifs for the upstream regions of the genes

Material

- Study the slides by Ilari Scheinin:
  `http://www.cs.helsinki.fi/u/ischeini/eob/`
- SQL reference: `http://dev.mysql.com/doc/refman/5.1/en/`
  (you will only need a very small part of SQL)
- Ensembl Schema:
  `http://www.ensembl.org/info/docs/api/core/core_schema.html`

Databases

Why would you use a database? Why not just a file?

- Separation of content from the programs (and programming languages) that access it
- Simultaneous access from multiple sources
- Diverse search capabilities
- Access rights
- Malfunction recovery
- Complex dependences
- Huge amounts of data

Data modeling in databases

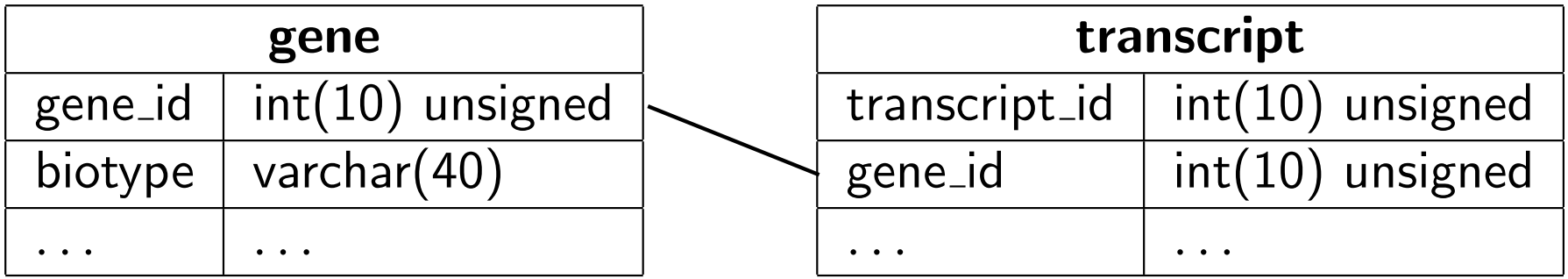
- Conceptual level (e.g. Entity-Relationship models)
  - Genes, transcripts, genes have names, transcripts are produced by genes, one gene can produce several transcripts...
- Structural level (e.g. relational model)
  - Other models exist also, object model...
- Physical level (e.g. B-trees)
  - How the data is actually stored

Relational databases

- Relational data model is based on set theory and the mathematical concept of relation
- A relation of k tuples is represented by a table with k attributes (columns)
- Operations of relation algebra
  - union, set difference, cartesian product
  - projection, selection
  - intersection, join
- SQL, Structured Query Language
- Database management system (DBMS)
  - authorization control, query optimizer, transaction manager, integrity control, command processor, buffer management, access methods
  - In our case MySQL

Database schema

- Describes the structure of the database
  - Tables
  - Table attributes
  - Relationships between tables



Structured Query Language (SQL)

- SQL is a language for accessing and manipulating a database
- For this project you will need only a part of SQL
- The SELECT query is the most important one for this project

SHOW

- SHOW TABLES
  - List all tables in a database

```
mysql> SHOW TABLES;
+-----------------------------------------+
| Tables_in_homo_sapiens_core_59_37d      |
+-----------------------------------------+
| alt_allele                              |
| analysis                                |
| analysis_description                    |
| assembly                                |
| assembly_exception                      |
...
```

SELECT

- Used for retrieving data from the database
- SELECT select_expr [, ...]
  [FROM table_references
  [WHERE where_condition]
  [GROUP BY col_name [, . . .]]
  [HAVING where_condition]
  [ORDER BY col_name [ASC|DESC], . . .]
  [LIMIT [offset,] row_count]]
- `http://dev.mysql.com/doc/refman/5.1/en/select.html`

SELECT example

```
mysql> SELECT gene_id,biotype FROM gene LIMIT 10;
+---------+---------+
| gene_id | biotype |
+---------+---------+
|  150024 | lincRNA |
|  150025 | lincRNA |
|  150026 | lincRNA |
|  150027 | lincRNA |
|  150028 | lincRNA |
|  150029 | lincRNA |
|  150030 | lincRNA |
|  150031 | lincRNA |
|  150032 | lincRNA |
|  150033 | lincRNA |
+---------+---------+
10 rows in set (0.00 sec)
```

JOIN

- Using multiple tables in a single query
- Multiple ways, here showing just one way

```
mysql> SELECT gene.gene_id, gene.biotype, transcript_id
    -> FROM gene JOIN transcript ON gene.gene_id=transcript.gene_id
    -> LIMIT 10;
+---------+----------------+---------------+
| gene_id | biotype        | transcript_id |
+---------+----------------+---------------+
|  135047 | protein_coding |        377576 |
|  135047 | protein_coding |        377577 |
|  135047 | protein_coding |        377578 |
|  135047 | protein_coding |        377575 |
|  135047 | protein_coding |        377571 |
|  135047 | protein_coding |        377572 |
|  135047 | protein_coding |        377573 |
|  135047 | protein_coding |        377574 |
|  135046 | protein_coding |        377565 |
|  135047 | protein_coding |        377566 |
+---------+----------------+---------------+
10 rows in set (0.00 sec)
```

How is the SELECT query processed?

- SELECT gene.gene_id, gene.biotype, transcript_id
  FROM gene JOIN transcript ON gene.gene_id=transcript.gene_id

gene	
gene_id	int(10) unsigned
biotype	varchar(40)
...	...

transcript	
transcript_id	int(10) unsigned
gene_id	int(10) unsigned
...	...

gene	
gene_id	**biotype**
135047	protein_coding
135046	protein_coding
150024	lincRNA

transcript	
transcript_id	**gene_id**
377576	135047
377577	135047
377565	135046

SELECT query: step 1

- Form a cartesian product of all the rows in the query
- FROM gene, transcript

gene.gene_id	gene.biotype	transcript.transcript_id	transcript.gene_id
135047	protein_coding	377576	135047
135047	protein_coding	377577	135047
135047	protein_coding	377575	135046
135046	protein_coding	377576	135047
135046	protein_coding	377577	135047
135046	protein_coding	377575	135046
150024	lincRNA	377576	135047
150024	lincRNA	377577	135047
150024	lincRNA	377575	135046

SELECT query: step 2

- Apply join condition
- JOIN ON gene.gene_id = transcript.gene_id

gene.gene_id	gene.biotype	transcript.transcript_id	transcript.gene_id
135047	protein_coding	377576	135047
135047	protein_coding	377577	135047
~~135047~~	~~protein_coding~~	~~377575~~	~~135046~~
~~135046~~	~~protein_coding~~	~~377576~~	~~135047~~
~~135046~~	~~protein_coding~~	~~377577~~	~~135047~~
135046	protein_coding	377575	135046
~~150024~~	~~lincRNA~~	~~377576~~	~~135047~~
~~150024~~	~~lincRNA~~	~~377577~~	~~135047~~
~~150024~~	~~lincRNA~~	~~377575~~	~~135046~~

SELECT query: step 3

- Select the columns
- SELECT gene.gene_id, gene.biotype, transcript_id

gene.gene_id	**gene.biotype**	**transcript.transcript_id**
135047	protein_coding	377576
135047	protein_coding	377577
135046	protein_coding	377575

SHOW PROCESSLIST

- SHOW [FULL] PROCESSLIST
  - shows all queries running under your username
- KILL id
  - can be used to kill a slow runaway query
- Everybody is using the same username, so don't kill each other's queries!

More about SQL

- For more about SQL see Ilari Scheinin's lecture slides! (Linked to course web page)

Ensembl database

- "The European" genome database
- 61 species
- New versions released $\sim$ every two months
- Current release is 69 (19.10.2012)
- Uses a MySQL relational database
  - `http://www.ensembl.org/info/docs/api/core/core_schema.html`

Ensembl mirror for the course

- users.cs.helsinki.fi
- MySQL mirror of the main human database
  - homo_sapiens_core_59_37d
- FASTA sequences for human transcriptome
  - /home/tkt_mbie/fasta/transcriptome/Homo_sapiens.GRCh37.59.cdna.all.fa
- FASTA sequences for human chromosomes
  - /home/tkt_mbie/fasta/genome/
  - One file per chromosome

Ensembl schema

- See Ilari Scheinin's slides on Ensembl schema (linked to course web page)
- ... and the Ensembl documentation

  `http://www.ensembl.org/info/docs/api/core/core_schema.html`

Python's DB API

- Database application programming interface
- Two levels:
  - top level provides an abstract interface, which is similar across all supported database engines $\Longrightarrow$ portability
  - lower level consists of drivers for specific engines $\Longrightarrow$ the one needed has to be installed, MySQLdb in out case

A DB-API script

- import the MySQLdb module
- open a connection to the MySQL server
- issue statements and retrieve results
- close the server connection

Example script

```
# fetchone.py

import MySQLdb

conn = MySQLdb.connect (host = "localhost",
                        user = "anonymous",
                        db = "homo_sapiens_core_59_37d",
                        unix_socket = "/home/tkt_mbie/mysql/socket")

cursor = conn.cursor ()

cursor.execute ("SELECT map_id, map_name FROM map")

while (1):
  row = cursor.fetchone ()
  if row == None:
    break
  print "%s\t%s" % (row[0], row[1])

cursor.close ()

conn.close ()
```