BITAH05 – Database technologies

Jasper Anckaert

Lecture 3 – MySQL and other languages, NoSQL, online databases, APIs & version control

Previously

- MySQL monitor → GUI
  - MySQL Workbench
- Data model
  - Structure of data
    - Conceptual
    - Logical
    - Physical
- Database model
  - Flat
  - Hierarchical
  - Network
  - Relational
  - Object-relational
  - Object-oriented

Previously

- Normalisation
  - UNF
  - 1NF
  - 2NF
  - 3NF
  - BCNF

- MySQL Workbench
  - Create model $\rightarrow$ Entity Relationship Diagram
  - Forward engineer
  - Reverse engineer
  - Import/Export data

MySQL and other languages

PHP

- MySQLi extension
  - Works only with MySQL
  - Procedural vs. Object-Oriented
  - `sudo yum install php-mysqli`
- PDO (PHP Data Objects)
  - Works with 12 different database systems

MySQL and other languages

PHP – MySQLi Procedural: Connect

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

MySQL and other languages

PHP – MySQLi Procedural: Query

```
<?php

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

- Other possibilities: INSERT, SELECT, DELETE, UPDATE

MySQL and other languages

PHP – MySQLi Procedural: INSERT

```
<?php

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " .
$last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);

?>
```

MySQL and other languages

PHP – MySQLi Procedural: INSERT (prepared)

```
<?php
// prepare and bind
$stmt = mysqli_stmt_init($conn);
mysqli_stmt_prepare($stmt, "INSERT INTO MyGuests (firstname, lastname,
email) VALUES (?, ?, ?)");
mysqli_stmt_bind_param ($stmt,"sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
mysqli_stmt_execute ($stmt);

echo "New record created successfully";

mysqli_stmt_close($stmt);

mysqli_close($conn);

?>
```

MySQL and other languages

PHP – MySQLi Procedural: SELECT

```
<?php
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

MySQL and other languages

PHP – MySQLi Object-Oriented: Connect, Query, Insert

```
<?php
$conn = new mysqli($servername, $username, $password);

$sql = "CREATE DATABASE myDB";

$conn->query($sql);


$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully. Last inserted ID is: " .
$last_id;
}
?>
```

MySQL and other languages

PHP – MySQLi Object-Oriented: Insert (prepared)

```
<?php
 // prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
echo "New records created successfully";

$stmt->close();
$conn->close();


?>
```

MySQL and other languages

PHP – MySQLi Object-Oriented: SELECT

```
<?php

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
}

$conn->close();

?>
```

MySQL and other languages

Python (more info in BIT04-Scripting)

- mysql.connector

```
import mysql.connector
cnx = mysql.connector.connect(user='username', database='myDD')
cursor = cnx.cursor()
query = ("SELECT id, firstname, lastname FROM MyGuests")
cursor.execute(query)
cursor.close()
cnx.close()
```

- MySQLdb

```
import MySQLdb
db = MySQLdb.connect(host="localhost",user=”username",passwd=”password",db=”myDB”)
cur = db.cursor()
cur.execute("SELECT id, firstname, lastname FROM MyGuests")
db.close()
```

NoSQL

Introduction

- Database Management Systems
- Non relational
- More flexible
- Big-data and real-time web applications
- Lack of
  - Atomicity
  - Consistency
  - Isolation
  - Durability

NoSQL

Introduction

- Different categories
  - Column store
  - Key-value store
  - Graph store
  - Multi-model
  - Document store

NoSQL

Introduction

- Column store
  - Tuples consist of 3 elements
    - Unique name
    - Value
    - Timestamp
  - Does not have to be in every row
  - Number of columns can change from row to row
  - E.g. Apache Cassandra

NoSQL

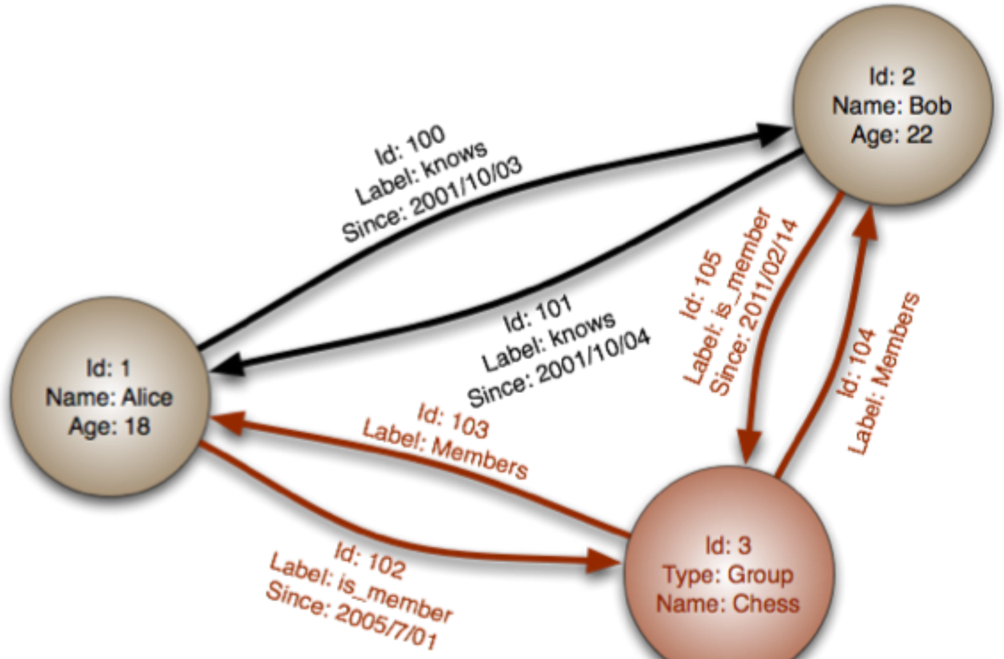
Introduction

- Key-value store
    - Data is stored in dictionary or hash
        - Collection of object or records
            - Different fields
    - Unique key per record
    - Single collection, different fields per record possible
    - Uses less memory
    - E.g. Oracle NoSQL Database

NoSQL

Introduction

- Graph store
  - Nodes, edges and properties
  - Simple and rapid data retrieval
  - SPARQL
  - Shortest path queries
  - E.g. OrientDB



NoSQL

Introduction

- Multi model
    - Multiple data models against a single, integrated backend
    - Document, graph, relational, and key-value models supported
    - E.g. Couchbase (relational, document)

NoSQL

Introduction

- Document store
    - Semi-structured data
    - Subclass of key-value store
        - Internal structure in the document
        - All information for given object stored in single instance
        - Every stored object can be different from every other
    - Web applications
    - Documents identified by unique key
    - Supports CRUD

NoSQL

Introduction

- Document store
  - No predefined data formats
    - Change in type and form has no effect on database and existing stored documents
  - No normalisation
  - E.g. MongoDB

NoSQL

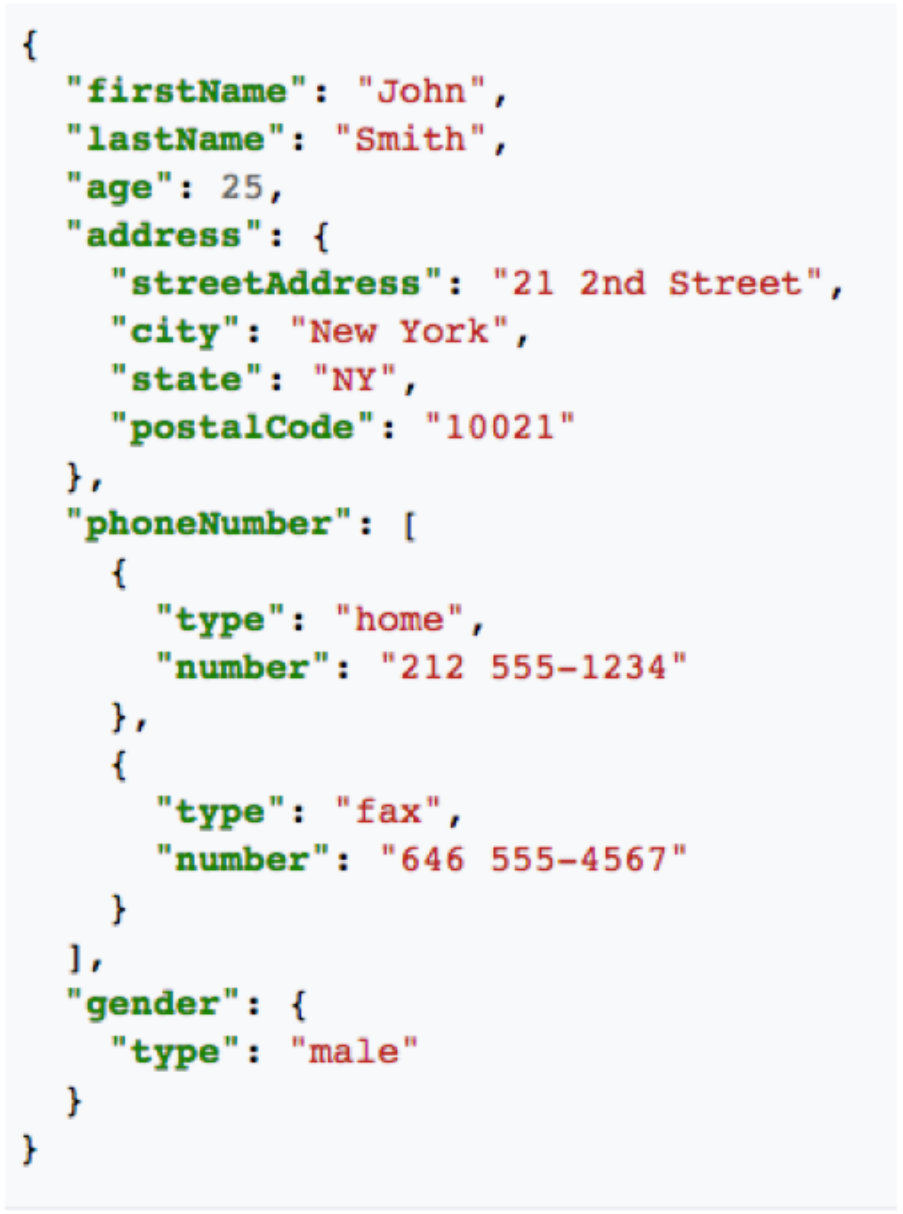
mongoDB

- Free and open-source
- JSON-like documents
- Ad hoc queries
  - Fields, ranges, regex
- Indexes
  - Primary and secondary
- Replication
- Load balancing
  - Sharding
- File storage
  - Grid File System (GridFS): store files larger than 16 MB
- Aggregation
  - MapReduce

NoSQL

mongoDB – JSON format

- JavaScript Object Notation
  - Data-interchange format
  - Easy to parse and generate
  - Objects and arrays
    - Attribute-value pairs



```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

NoSQL

mongoDB – install server

- Linux
  - https://docs.mongodb.com/getting-started/shell/tutorial/install-on-linux/
- Mac
  - https://docs.mongodb.com/getting-started/shell/tutorial/install-mongodb-on-os-x/
- Windows
  - https://docs.mongodb.com/getting-started/shell/tutorial/install-mongodb-on-windows/

NoSQL

mongoDB – start, stop, restart

- Linux/Mac

```
$ sudo service mongod start
$ sudo service mongod stop
$ sudo service mongod restart

$ mongod
```

- Windows

```
$ C:\> "C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe”
$ C:\> "net start MongoDB”
$ C:\> "net stopMongoDB”
```

NoSQL

mongoDB – mongo shell

- Once you have installed and starte MongoDB, connect to the mongo shell
  - Linux/Mac
    ```
    $ mongo
    ```
  - Windows
    ```
    $ C:\> ”mongo.exe”
    ```
- Many options, check the help
  ```
  > help
  ```

NoSQL

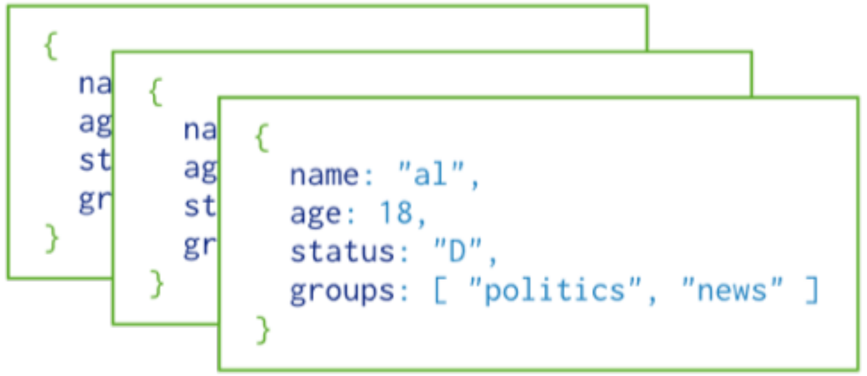
mongoDB – databases and collections

- BSON documents are stored in collections, collections in databases

- Select a database in the mongo shell

  ```
  > use myDB
  ```
- Create a database
  - DB does not have to exist yet

  ```
  > use myNewDB
  > db.myNewCollection1.insert({x:1})
  ```



Collection

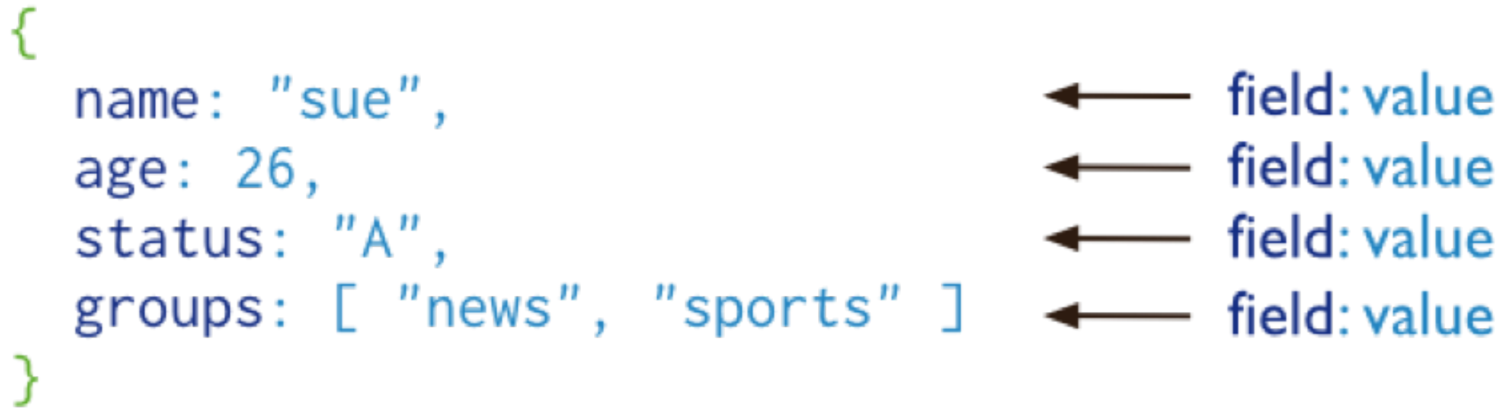
- Collections (~tables in relational databases)
  - Collection does not have to exist yet

  ```
  > db.myNewCollection2.insert( { x: 1 } )
  > db.myNewCollection3.createIndex( { y: 1 } )
  ```

NoSQL

mongoDB – documents

- Records = documents
    - Field and value pairs
    - Values of field may include other documents, arrays, and arrays of documents

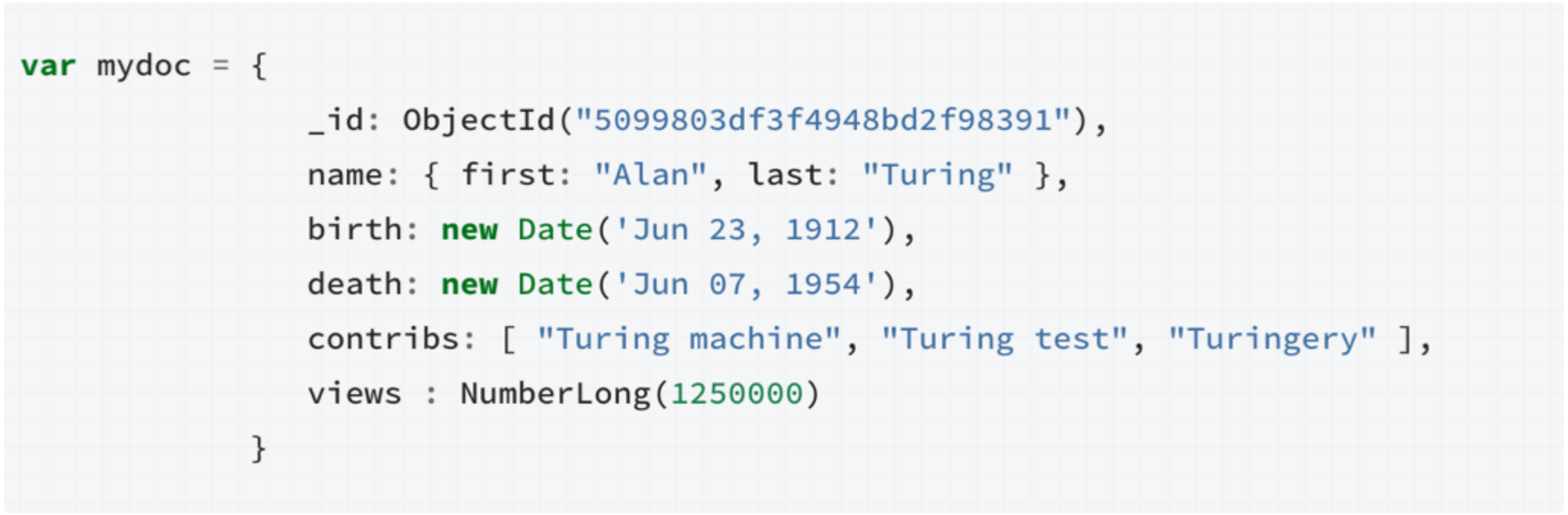


```
{
   name: "sue",
   age: 26,
   status: "A",
   groups: [ "news", "sports" ]
}
```

- Advantages of documents
    - Similar to programming languages
    - Less need for joins
    - Polymorphism

NoSQL

mongoDB – data types



```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

NoSQL

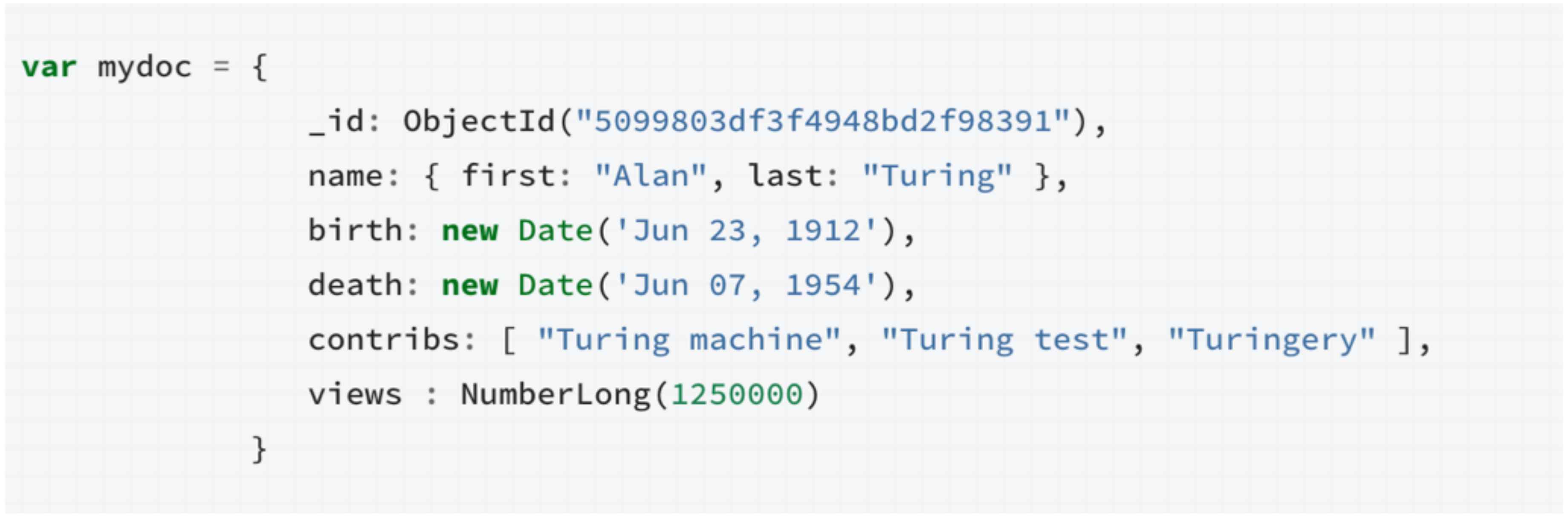
mongoDB – field names

- Strings with restrictions
  - The field name `_id` is reserved for use as a primary key; its value must be unique in the collection, is immutable, and may be of any type other than an array.
  - The field names cannot start with the dollar sign (`$`) character.
  - The field names cannot contain the dot (`.`) character.
  - The field names cannot contain the `null` character.

NoSQL

mongoDB – dot notation

- MongoDB uses the dot notation to access the elements of an array and to access the fields of an embedded document.
    - Arrays: use zero-based index
    - Embedded documents: use field name



```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
            }
```

NoSQL

mongoDB – _id field

- Primary key
- By default, MongoDB creates a unique index on the _id field during the creation of a collection
    - ObjectId
    - Always first field in document
    - If not, moved to beginning by DB
- May contain values of any BSON data type, other than an array.

NoSQL

mongoDB – BSON data types

- ObjectId
    - small, likely unique, fast to generate, and ordered
        - a 4-byte value representing the seconds since the Unix epoch,
        - a 3-byte machine identifier,
        - a 2-byte process id, and
        - a 3-byte counter, starting with a random value.
- String
- Timestamp
    - Internal MongoDB use
        - the first 32 bits are a time_t value (seconds since the Unix epoch)
        - the second 32 bits are an incrementing ordinal for operations within a given second.
- Date
    - number of milliseconds since the Unix epoch

NoSQL

mongoDB – mongo shell (continued)

- See all databases on the server
```
> show dbs
```
- List all methods you can use on your db object
```
> db.help()
```
- See all collections in a database
```
> show collections
```
- List all methods you can use on your collection object
```
> db.collection.help()
```
- List all cursor methods
```
> db.collection.find().help()
```

NoSQL

Query optimization

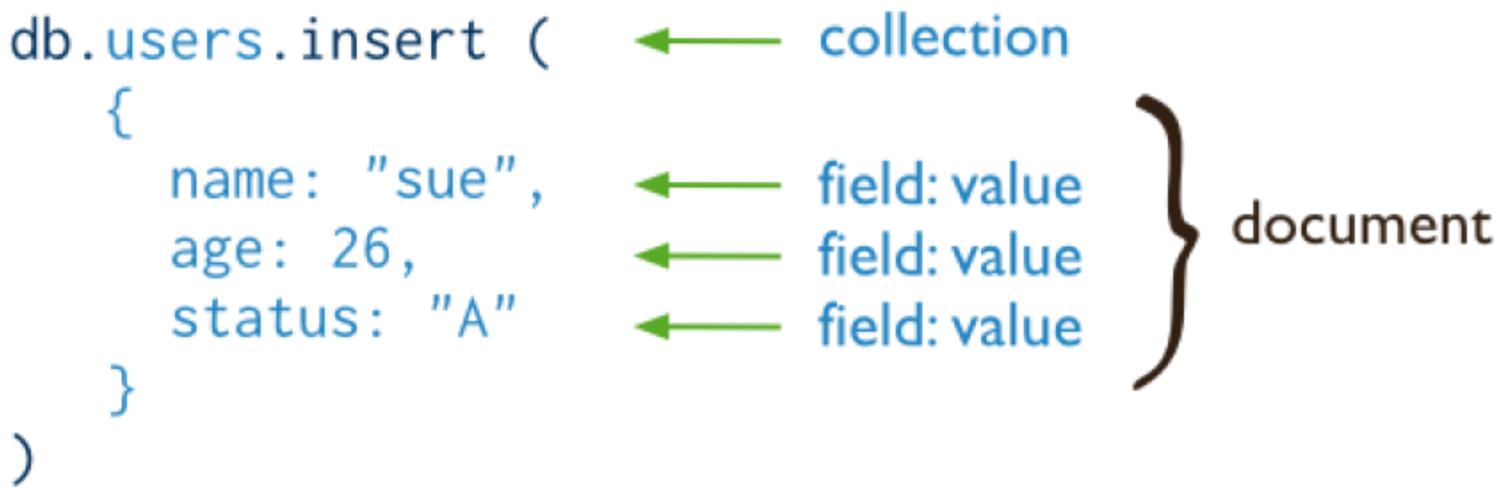
- Create index
- Query selectivity
    - Less selective queries match larger percentage of documents
        - Unable to use indexes (effectively)
- Covered query
    - All the fields in the query are part of an index
    - All the fields returned in the results are in the same index
- Limit number of query results
    - `limit()` method
    - Reduces network demand
- Return only necessary data
    - Use query projections

NoSQL

mongoDB – create operations

- Add new documents to a collection

```
> db.collection.insert()
> db.collection.insertOne()
> db.collection.insertMany()
```



```
db.users.insert (
    {
      name: "sue",
      age: 26,
      status: "A"
    }
)
```

NoSQL

mongoDB – create operations

- Collection created if not exists
- `db.collection.insertOne()`
  - Insert a single document into a collection
- `db.collection.insertMany()`
  - Insert multiple documents into a collection
- `db.collection.insert()`
  - Insert a single or multiple documents into a collection

NoSQL

Exercises

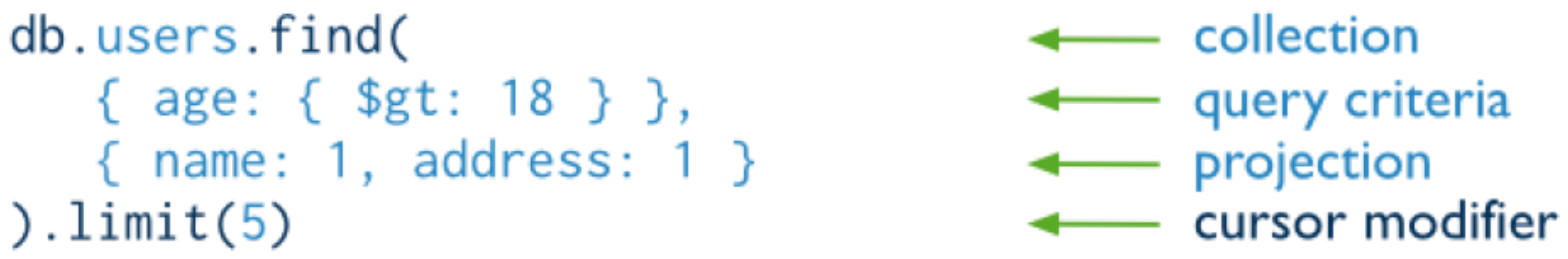
- Create a collection `users` and add yourself to it (name, age, sex)
- Add some more users using a different command

NoSQL

mongoDB – read operations

- Retrieve documents from a collection

```
> db.collection.find()
> db.collection.findOne()
```



```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

NoSQL

mongoDB – read operations

- `db.collection.find()` returns a cursor to the matching documents
- Optional fields are possible
    - Query filter
    - Query projection
- Cursor modifier if wanted

NoSQL

mongoDB – query filter

- Specify equality condition
```
> db.collection.find({ <field1>: <value1>, ... })
```
- Use query operators
```
> db.collection.find({ <field1>: { <operator1>: <value1> }, ...
})
```
- AND conditions
  - By default when specifying multiple conditions
- OR conditions
```
> db.collection.find({ $or: { < field1 >: <value1> }, { < field2
>: <value2> }})
```

NoSQL

mongoDB – query operators

- Comparison

`{ <field>: { <operator>: <value> } }`	
`$eq`	Values that are equal to a specified value
`$gt`	Values that are greater than a specified value
`$gte`	Values that are greater than or equal to a specific value
`$lt`	Values that are less than a specified value
`$lte`	Values that are less than or equal to a specific value
`$ne`	Values that are not equal to a specified value
`{ field: { <operator>: [<value1>, <value2>, ... <valueN> ] } }`	
`$in`	Any of the values specified in an array
`$nin`	None of the values specified in an array

NoSQL

mongoDB – query operators

- Logical

`{ <operator>: [ { <expression1> }, { <expression2> }, ... , { <expressionN> } ] }`	
`$or`	All documents that match the conditions of either clause
`$and`	All documents that match the conditions of both clauses
`$nor`	All documents that fail to match both clauses
{ field: { $not: { <operator-expression> } } }	
`$not`	All documents that do not match the query expression

- Element

`{ field: { $exists: <boolean> } }`	
`$exists`	All documents that have the specified field

NoSQL

mongoDB – query operators

- Array

`{ <field>: { $all: [ <value1> , <value2> ... ] } }`	
`$all`	Arrays that contain all elements specified in the query
`{ <field>: { $elemMatch: { <query1>, <query2>, ... } } }`	
`$elemMatch`	Documents where the element in the array field matches all the specified `$elemMatch` conditions
`{ field: { $size: 2 } }`	
`$size`	Documents if the array field is a specified size

NoSQL

mongoDB – query projection

```
{ field1: <value>, field2: <value> ... }
```

- Value can be any of the following
    - 1 or true
    - 0 or false
    - Expression using projection operators

NoSQL

mongoDB –projection operators

`db.collection.find( { <array>: <value> ... }, { "<array>.$": 1 } )`	
`$`	First element in an array that matches the query condition
`{ <field>: { $elemMatch: { <query1>, <query2>, ... } } }`	
`$elemMatch`	First element in an array that matches all the specified `$elemMatch` conditions

NoSQL

mongoDB – cursor modifier

- Methods that modify the way that the underlying query is executed
  - `sort()`
    - Returns results ordered according to a sort specification
  - `count()`
    - Returns the number of documents in the result set
  - `hasNext()`
    - Returns true if the cursor has documents and can be iterated
  - `next()`
    - Returns the next document in a cursor.
  - `limit()`
    - Constrains the size of a cursor's result set.
  - `skip()`
    - Returns a cursor that begins returning results only after passing or skipping a number of documents.
  - `size()`
    - Returns a count of the documents in the cursor after applying skip() and limit() methods.

NoSQL

Exercises

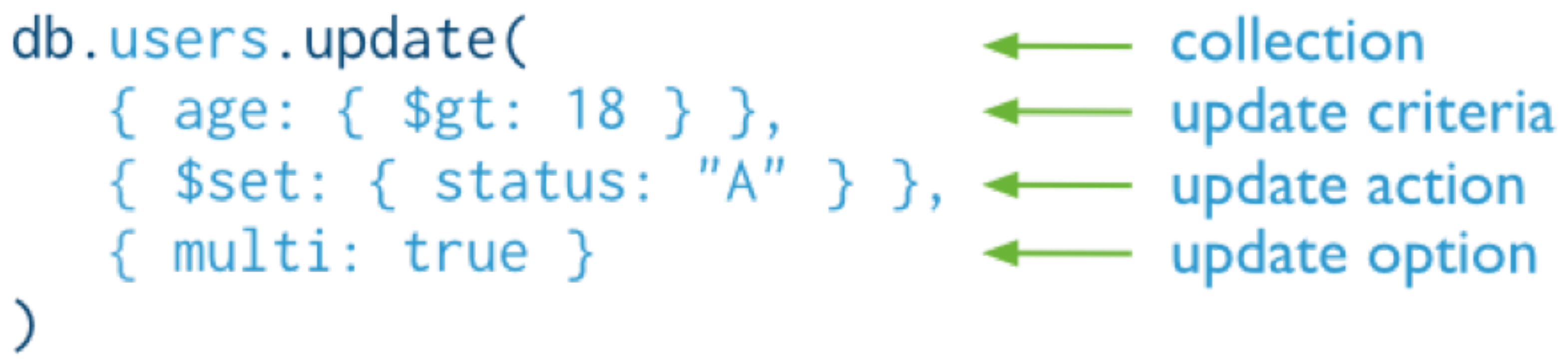
- Populate your collection using the contents of the file *mongo1.txt*
- Select all documents in the collection
- Select all documents where status field has the value A
- Select all documents in the collection where the status equals "A" and *either* age is less than than 30 or type equals 1
- Retrieve all documents from the users collection where status equals either "P"or "D"
- Retrieve all documents where the favorites field is an embedded document that contains only the fields artist equal to "Picasso" and food equal to "pizza", in that order
- Use the dot notation to match all documents where badges is an array that contains "black" as the first element
- Retrieve all documents where the finished array contains at least one element that is greater than ($gt) 15 and less than ($lt) 20

NoSQL

mongoDB – update operations

- Modify existing documents in a collection

```
> db.collection.update()
> db.collection.updateOne()
> db.collection.updateMany()
> db.collection.replaceOne()
```



```
db.users.update(
    { age: { $gt: 18 } },
    { $set: { status: "A" } },
    { multi: true }
)
```

NoSQL

mongoDB – update operators

`{ $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }`	
`$inc`	Increments the value of the field by the specified amount
`{ <operator>: { <field1>: <value1>, ... } }`	
`$set`	Sets the value of a field in a document.
`$addToSet`	Adds elements to an array only if they do not already exist in the set.
`$pop`	Removes the first or last item of an array.
`$pull`	Removes all array elements that match a specified query.
`$push`	Adds an item to an array.

NoSQL

Exercises

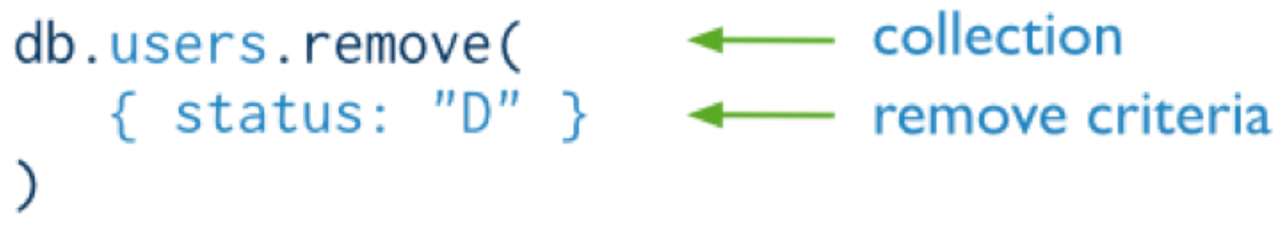
- Update all documents where Picasso is the favorite artist so that the value of the favorites.artist field is "Pisanello" and the value of the type field is 3
- Replace the *first* document that matches the filter name equals "abc" with the new document
  ```
  { name: "amy", age: 34, type: 2, status: "P", favorites: { "artist": "Dali",
  food: "donuts" } }
  ```
- Replace the first document that matches the filter name equals "xyz" with the new document (do not use the same method as before)
  ```
  { name: "mee", age: 25, type: 1, status: "A", favorites: { "artist": "Matisse",
  food: "mango" } }
  ```

NoSQL

mongoDB – delete operations

- Add new documents to a collection

```
> db.collection.remove()
> db.collection.deleteOne()
> db.collection.deleteMany()
```



```
db.users.remove(
   { status: "D" }
)
```

NoSQL

Exercises

- Remove all documents from the users collection where the status field equals "P”
- Remove the first document from the users collection where the status field equals ”D” (2 ways)

NoSQL

MongoDB vs. SQL

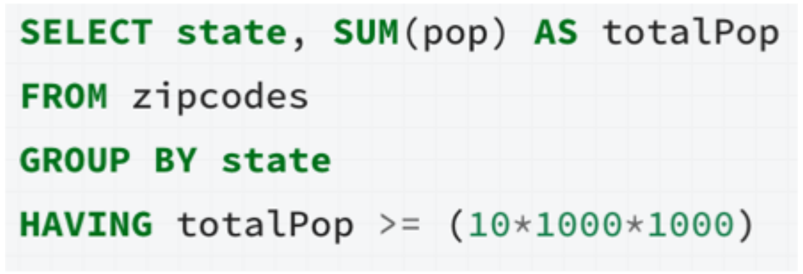
MongoDB	SQL
database	database
collection	table
document	row
field	column
index	index
Primary key	Primary key
Embedded documents	joins

NoSQL

mongoDB – aggregation

- `aggregate()` method
  - Return states with populations above 10 Million

```
> db.zipcodes.aggregate( [
   { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },
   { $match: { totalPop: { $gte: 10*1000*1000 } } } ] )
```



```
SELECT state, SUM(pop) AS totalPop
FROM zipcodes
GROUP BY state
HAVING totalPop >= (10*1000*1000)
```

- Many aggregation pipeline operators
  - `$sum`, `$avg`, `$group`, `$match`, `$limit`, `$skip`, `$sort`, `$concat`, …

NoSQL

Exercises

- Import the zipcodes selection from file zips.json
```
  $ mongoimport zips.json
```

- Return average city population by state
- Return largest and smallest cities by state

NoSQL

mongoDB - GUI

- MongoDB Compass
  - Visually explore your data
  - Interact with your data with full CRUD functionality
  - Run ad hoc queries
  - View detailed information about indexes
  - View visual explain plans to help optimize query performance
- Robo 3T
  - Full Power of a MongoDB Java Script environment. Robo 3T embeds a complete JavaScript engine (based on Mozilla SpiderMonkey).
  - Multiple Shells. Open as many shells as you need. Every tab in Robo 3T is a MongoDB shell, fully isolated from each other.
  - Multiple Results. Robo 3T executes your code in statement by statement way. That means that you will receive as many result as many statements you have.
  - Autocompletion. Robo 3T provides you with autocompletion for all objects (and thus functions) that are known by JavaScript runtime, including autocompletion for databases, collections and even your document objects.
  - Cross-platform, open source.

Databases in bioinformatics

Why?

- Make biological data available to scientists
    - Collect data in a single place
    - Published data may be difficult to find (time-consuming)
- Make biological data available in computer-readable form
    - Needed for analysis

Databases in bioinformatics

Types of databases

- Characterisation based on several properties
    - Type of data
    - Data entry and quality control
    - Primary or derived data
    - Technical design
    - Maintainer status
    - Availability

Databases in bioinformatics

Types of databases

- Type of data
    - Nucleotide sequences
    - Protein sequences
    - Gene expression data
    - Metabolic pathways
    - 3D structures

Databases in bioinformatics

Types of databases

- Data entry and quality control
    - Data deposited directly
    - Appointed curators add and update data
    - Treatment of erroneous data: removed, or marked
    - Type and degree of error checking

Databases in bioinformatics

Types of databases

- Primary or derived data
    - Primary databases: experimental results
    - Secondary databases: results of analysis of primary databases
    - Aggregate of many databases
        - Consolidation of data
        - Combination of data

Databases in bioinformatics

Types of databases

- Technical design
    - Flat files
    - Relational database
    - Object oriented database

Databases in bioinformatics

Types of databases

- Maintainer status
  - Large, public institution (EMBL, NCBI)
  - Quasi-academic institute (Swiss Institute of Bioinformatics, TIGR)
  - Academic group or scientist
  - Commercial company

Databases in bioinformatics

Types of databases

- Availability
    - Publicly available, no restrictions
    - Available, but with copyright
    - Accessible, but not downloadable
    - Academic, but not freely available
    - Commercial

Databases in bioinformatics

Identifiers and accession codes

- Identify an entry in two different ways
    - Identifier
        - String of letters and digits (understandable)
        - Can usually change
    - Accession code (or number)
        - Number that uniquely identifies an entry in its database
        - Stable

Databases in bioinformatics

Primary nucleotide sequence databases

- 3 main databases
    - EMBL (ENA), GenBank, DDBJ
    - Little error checking
    - Redundancy
    - Synchronized on a daily basis
    - No legal restrictions

Databases in bioinformatics

Primary nucleotide sequence databases

- European Nucleotide Archive
    - DNA and RNA sequences
    - 3 databases
        - Sequence Read Archive
        - Trace Archive
        - EMBL Nucleotide Sequence Database
    - Maintained by European Bioinformatics Institute
    - XML, HTML, FASTA, FASTQ
    - http://www.ebi.ac.uk/ena

Databases in bioinformatics

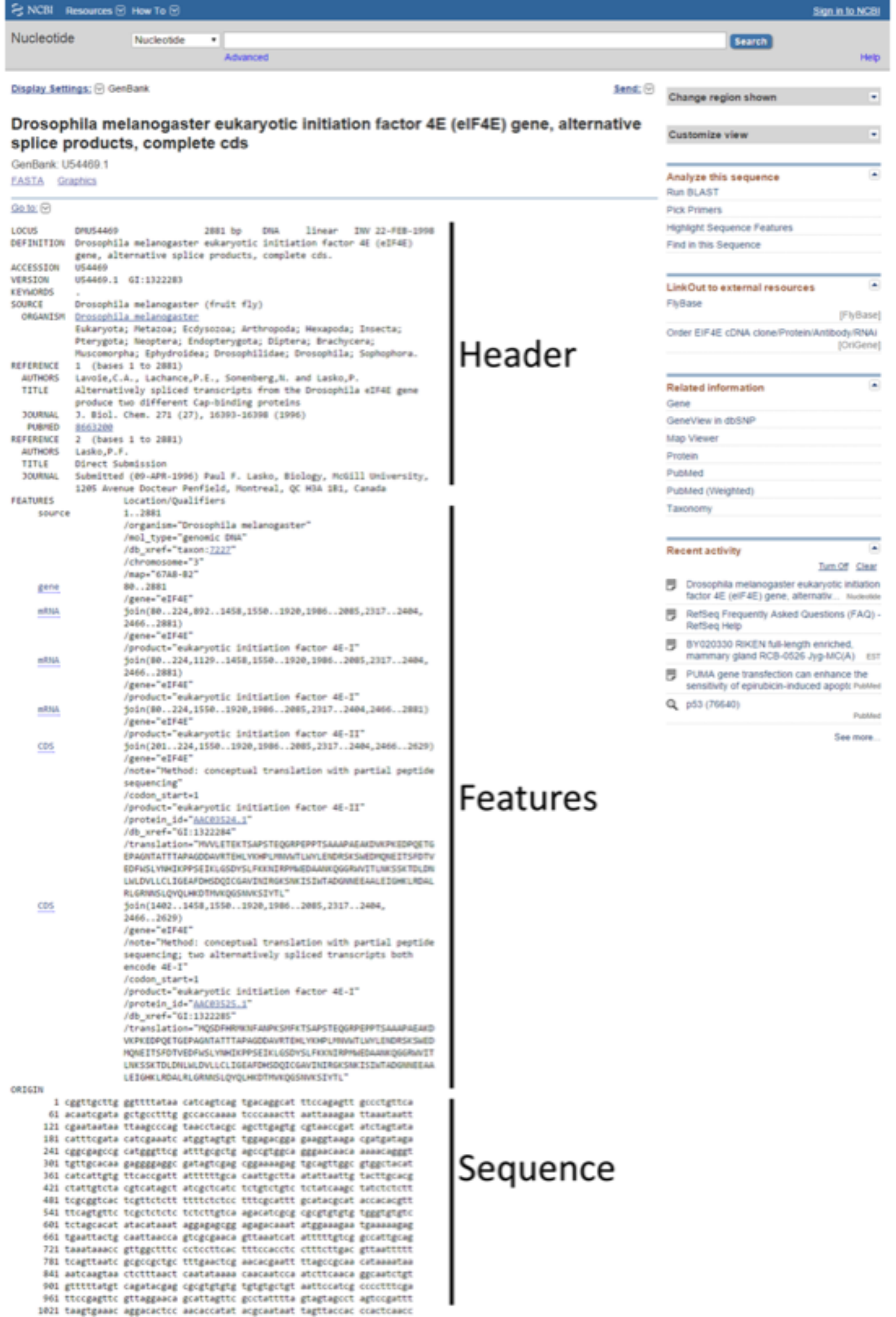
Primary nucleotide sequence databases

- GenBank
    - Open access
    - Publicly available nucleotide sequences and their protein translations
    - More than 100000 distinct organisms
    - Maintained by National Center for Biotechnology Information (NCBI)
    - Entries retrievable by NCBI GenBank webpage (or FTP)

Databases in bioinformatics

The GenBank format

- Flatfile with 3 main sections
  - Header
  - Features
  - Sequence



Drosophila melanogaster eukaryotic initiation factor 4E (eIF4E) gene, alternative splice products, complete cds

GenBank: U54469.1

```
LOCUS       DMU54469                2881 bp    DNA     linear   INV 22-FEB-1998
DEFINITION  Drosophila melanogaster eukaryotic initiation factor 4E (eIF4E)
            gene, alternative splice products, complete cds.
ACCESSION   U54469
VERSION     U54469.1  GI:1322283
KEYWORDS    .
SOURCE      Drosophila melanogaster (fruit fly)
  ORGANISM  Drosophila melanogaster
            Eukaryota; Metazoa; Ecdysozoa; Arthropoda; Hexapoda; Insecta;
            Pterygota; Neoptera; Endopterygota; Diptera; Brachycera;
            Muscomorpha; Ephydroidea; Drosophilidae; Drosophila; Sophophora.
REFERENCE   1  (bases 1 to 2881)
  AUTHORS   Lavoie,C.A., Lachance,P.E., Sonenberg,N. and Lasko,P.
  TITLE     Alternatively spliced transcripts from the Drosophila eIF4E gene
            produce two different Cap-binding proteins
  JOURNAL   J. Biol. Chem. 271 (27), 16393-16398 (1996)
   PUBMED   8663200
REFERENCE   2  (bases 1 to 2881)
  AUTHORS   Lasko,P.F.
  TITLE     Direct Submission
  JOURNAL   Submitted (09-APR-1996) Paul F. Lasko, Biology, McGill University,
            1205 Avenue Docteur Penfield, Montreal, QC H3A 1B1, Canada
FEATURES             Location/Qualifiers
     source          1..2881
                     /organism="Drosophila melanogaster"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:7227"
                     /chromosome="3"
                     /map="67A8-B2"
     gene            80..2881
                     /gene="eIF4E"
     mRNA            join(80..224,892..1458,1550..1920,1986..2085,2317..2404,
                     2466..2881)
                     /gene="eIF4E"
                     /product="eukaryotic initiation factor 4E-I"
     mRNA            join(80..224,1129..1458,1550..1920,1986..2085,2317..2404,
                     2466..2881)
                     /gene="eIF4E"
                     /product="eukaryotic initiation factor 4E-I"
     mRNA            join(80..224,1550..1920,1986..2085,2317..2404,2466..2881)
                     /gene="eIF4E"
                     /product="eukaryotic initiation factor 4E-II"
     CDS             join(201..224,1550..1920,1986..2085,2317..2404,2466..2629)
                     /gene="eIF4E"
                     /note="Method: conceptual translation with partial peptide
                     sequencing"
                     /codon_start=1
                     /product="eukaryotic initiation factor 4E-II"
                     /protein_id="AAC03524.1"
                     /db_xref="GI:1322284"
                     /translation="MVVLETEKTSAPSTEQGRPEPPTSAAAPAEAKDVKPKEDPQETG
                     EPAGNTATTTAPAGDDAVRTEHLYKHPLMNVWTLWYLENDRSKSWEDMQNEITSFDTV
                     EDFWSLYNHIKPPSEIKLGSDYSLFKKNIRPMWEDAANKQGGRWVITLNKSSKTDLDN
                     LWLDVLLCLIGEAFDHSDQICGAVINIRGKSNKISIWTADGNNEEAALEIGHKLRDAL
                     RLGRNNSLQYQLHKDTMVKQGSNVKSIYTL"
     CDS             join(1402..1458,1550..1920,1986..2085,2317..2404,
                     2466..2629)
                     /gene="eIF4E"
                     /note="Method: conceptual translation with partial peptide
                     sequencing; two alternatively spliced transcripts both
                     encode 4E-I"
                     /codon_start=1
                     /product="eukaryotic initiation factor 4E-I"
                     /protein_id="AAC03525.1"
                     /db_xref="GI:1322285"
                     /translation="MQSDFHRMKNFANPKSMFKTSAPSTEQGRPEPPTSAAAPAEAKD
                     VKPKEDPQETGEPAGNTATTTAPAGDDAVRTEHLYKHPLMNVWTLWYLENDRSKSWED
                     MQNEITSFDTVEDFWSLYNHIKPPSEIKLGSDYSLFKKNIRPMWEDAANKQGGRWVIT
                     LNKSSKTDLDNLWLDVLLCLIGEAFDHSDQICGAVINIRGKSNKISIWTADGNNEEAA
                     LEIGHKLRDALRLGRNNSLQYQLHKDTMVKQGSNVKSIYTL"
ORIGIN
        1 cggttgcttg ggttttataa catcagtcag tgacaggcat ttccagagtt gccctgttca
       61 acaatcgata gctgcctttg gccaccaaaa tcccaaactt aattaaagaa ttaaataatt
      121 cgaataataa ttaagcccag taacctacgc agcttgagtg cgtaaccgat atctagtata
      181 catttcgata catcgaaatc atggtagtgt tggagacgga gaaggtaaga cgatgataga
      241 cggcgagccg catgggttcg atttgcgctg agccgtggca gggaacaaca aaaacagggt
      301 tgttgcacaa gaggggaggc gatagtcgag cggaaaagag tgcagttggc gtggctacat
      361 catcattgtg ttcaccgatt atttttgca caattgctta atattaattg tacttgcacg
      421 ctattgtcta cgtcatagct atcgctcatc tctgtctgtc tctatcaagc tatctctctt
      481 tcgcggtcac tcgttctctt ttttctctcc tttcgcattt gcatacgcat accacacgtt
      541 ttcagtgttc tcgctctctc tctcttgtca agacatcgcg cgcgtgtgtg tgggtgtgtc
      601 tctagcacat atacataaat aggagagcgg agagacaaat atggaaagaa tgaaaaagag
      661 tgaattactg caattaacca gtcgcgaaca gttaaatcat atttttgtcg gccattgcag
      721 taaataaacc gttggctttc cctccttcac tttccacctc ctttcttgac gttaattttt
      781 tcagttaatc gcgccgctgc tttgaactcg aacacgaatt ttagccgcaa cataaaataa
      841 aatcaagtaa ctctttaact caatataaaa caacaatcca atcttcaaca ggcaatctgt
      901 gtttttatgt cagatacgag cgcgtgtgtg tgtgtgctgt aattccatcg cccctttcga
      961 ttccgagttc gttaggaaca gcattagttc gcctatttta gtagtagcct agtccgattt
     1021 taagtgaaac aggacactcc aacaccatat acgcaataat tagttaccac ccactcaacc
```

Analyze this sequence
Run BLAST
Pick Primers
Highlight Sequence Features
Find in this Sequence
LinkOut to external resources
FlyBase
[FlyBase]
Order EIF4E cDNA clone/Protein/Antibody/RNAi
[OriGene]
Related information
Gene
GeneView in dbSNP
Map Viewer
Protein
PubMed
PubMed (Weighted)
Taxonomy
Recent activity
Turn Off
Clear
Drosophila melanogaster eukaryotic initiation factor 4E (eIF4E) gene, alternativ... Nucleotide
RefSeq Frequently Asked Questions (FAQ) - RefSeq Help
BY020330 RIKEN full-length enriched, mammary gland RCB-0526 Jyg-MC(A) EST
PUMA gene transfection can enhance the sensitivity of epirubicin-induced apopt PubMed
p53 (76640) PubMed
See more...
Header
Features
Sequence

Databases in bioinformatics

Primary nucleotide sequence databases

- DNA Data Bank of Japan
    - DNA sequences
    - Only nucleotide sequence data bank in Asia
    - http://www.ddbj.nig.ac.jp/

Databases in bioinformatics

Secondary nucleotide sequence databases

- RefSeq
    - DNA, RNA and their protein products
    - Annotated and curated
    - Single record for each natural biological molecule
- OMIM
    - Catalog of human genes and genetic disorders and traits
    - Based on selection and review of published peer-reviewed literature
- HapMap
    - Haplotype map of the human genome
    - Genetic variants affecting health, disease and responses to drugs and environmental factors

Databases in bioinformatics

Other nucleic acid databases

- Gene expression databases
    - Mostly microarray data
    - a.o. Gene Expression Omnibus, Expression Atlas, …
- Gene ontology
    - Relationships between concepts within a domain
- Genome databases
    - Annotated and analyzed genome sequences
    - a.o. Ensembl (Genomes), Flybase, Wormbase, …
- Phenotype databases
    - a.o. PhenCode
- RNA databases
    - a.o. miRBase, LNCipedia, …

Databases in bioinformatics

Sequencing databases

- Datasets from sequencing experiments
  - Sequence Read Archive
    - Hosted by NCBI
    - Raw data in BAM-format
    - Experimental metadata available
  - European Genome-phenome Archive
    - Hosted by EMBL-EBI
    - Data not publicly available

Databases in bioinformatics

Protein databases

- Protein sequence
  - Derived from translation of nucleotide sequences
    - secondary databases: NCBI Protein and trEMBL
  - Computational analysis, manual review and annotation
    - SwissProt
- Protein structure
  - a.o. Protein Data Bank, NCBI Structure

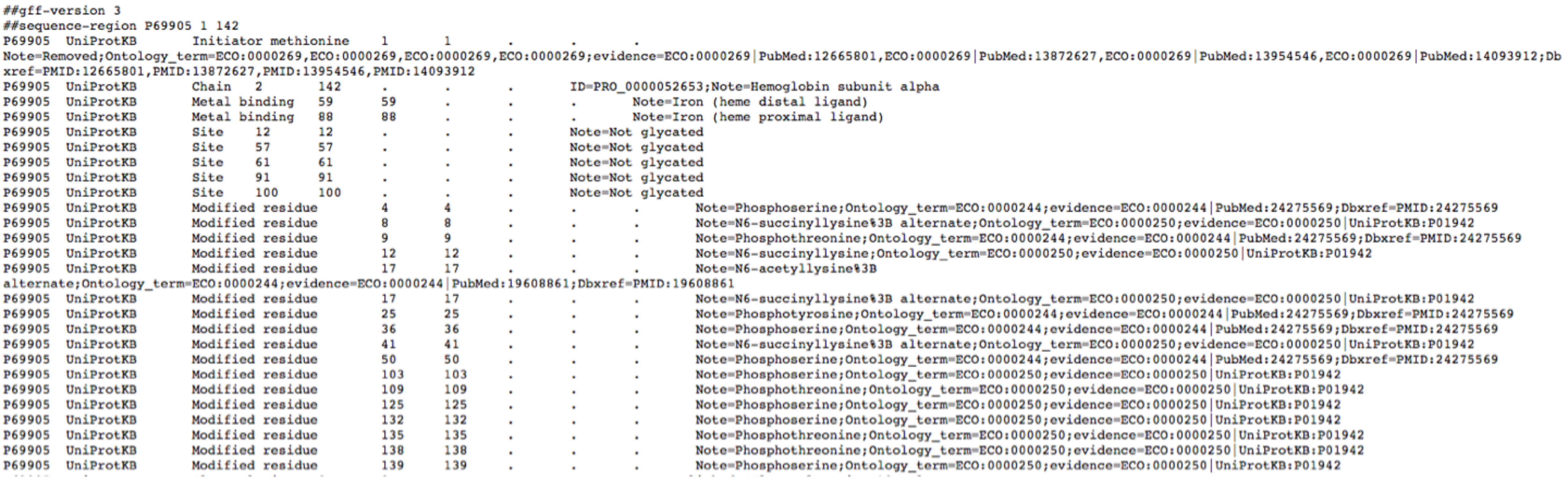
Databases in bioinformatics

The General Feature Format (GFF)

- Features of a particular gene, DNA and protein sequence
  - Tab-delimited
  - One line per feature, all but the final field in each feature line must contain a value; "empty" columns should be denoted with a '.’
    - **seqname** - name of the chromosome or scaffold; chromosome names can be given with or without the 'chr' prefix. **Important note**: the seqname must be one used within Ensembl, i.e. a standard chromosome name or an Ensembl identifier such as a scaffold ID, without any additional content such as species or assembly. See the example GFF output below.
    - **source** - name of the program that generated this feature, or the data source (database or project name)
    - **feature** - feature type name, e.g. Gene, Variation, Similarity
    - **start** - Start position of the feature, with sequence numbering starting at 1.
    - **end** - End position of the feature, with sequence numbering starting at 1.
    - **score** - A floating point value.
    - **strand** - defined as + (forward) or - (reverse).
    - **frame** - One of '0', '1' or '2'. '0' indicates that the first base of the feature is the first base of a codon, '1' that the second base is the first base of a codon, and so on..
    - **attribute** - A semicolon-separated list of tag-value pairs, providing additional information about each feature.

Databases in bioinformatics

The General Feature Format (GFF)



```
##gff-version 3
##sequence-region P69905 1 142
P69905  UniProtKB       Initiator methionine    1       1       .       .       .
Note=Removed;Ontology_term=ECO:0000269,ECO:0000269,ECO:0000269,ECO:0000269;evidence=ECO:0000269|PubMed:12665801,ECO:0000269|PubMed:13872627,ECO:0000269|PubMed:13954546,ECO:0000269|PubMed:14093912;Db
xref=PMID:12665801,PMID:13872627,PMID:13954546,PMID:14093912
P69905  UniProtKB       Chain   2       142     .       .       .       ID=PRO_0000052653;Note=Hemoglobin subunit alpha
P69905  UniProtKB       Metal binding   59      59      .       .       .       Note=Iron (heme distal ligand)
P69905  UniProtKB       Metal binding   88      88      .       .       .       Note=Iron (heme proximal ligand)
P69905  UniProtKB       Site    12      12      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    57      57      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    61      61      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    91      91      .       .       .       Note=Not glycated
P69905  UniProtKB       Site    100     100     .       .       .       Note=Not glycated
P69905  UniProtKB       Modified residue        4       4       .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        8       8       .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        9       9       .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        12      12      .       .       .       Note=N6-succinyllysine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        17      17      .       .       .       Note=N6-acetyllysine%3B
alternate;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:19608861;Dbxref=PMID:19608861
P69905  UniProtKB       Modified residue        17      17      .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        25      25      .       .       .       Note=Phosphotyrosine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        36      36      .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        41      41      .       .       .       Note=N6-succinyllysine%3B alternate;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        50      50      .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000244;evidence=ECO:0000244|PubMed:24275569;Dbxref=PMID:24275569
P69905  UniProtKB       Modified residue        103     103     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        109     109     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        125     125     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        132     132     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        135     135     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        138     138     .       .       .       Note=Phosphothreonine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
P69905  UniProtKB       Modified residue        139     139     .       .       .       Note=Phosphoserine;Ontology_term=ECO:0000250;evidence=ECO:0000250|UniProtKB:P01942
```

Databases in bioinformatics

Genome browsers

- Graphical interface for genomic data
    - UCSC genome browser
        - Search by gene name
        - Search by location (chrN:startposition-stoppostion)
    - Ensembl genome browser
        - Annotated genes aligned to a reference genome
        - Export data in multiple format (FASTA, GFF, EMBL, …)

Databases in bioinformatics

Exercices

- What information about the rabies virus sequence can you obtain from its annotations in the NCBI Sequence Database? Give the accession number, definition, organism and PubMed ID of the record.
- How many nucleotide sequences are there from the bacterium Chlamydia trachomatis?
- How many nucleotide sequences are there from the bacterium Chlamydia trachomatis in the RefSeq part of the NCBI Sequence Database?
- How many nucleotide sequences were submitted to NCBI by Matthew Berriman?
- How many nucleotide sequences from the nematode worms are there in the RefSeq Database?

Databases in bioinformatics

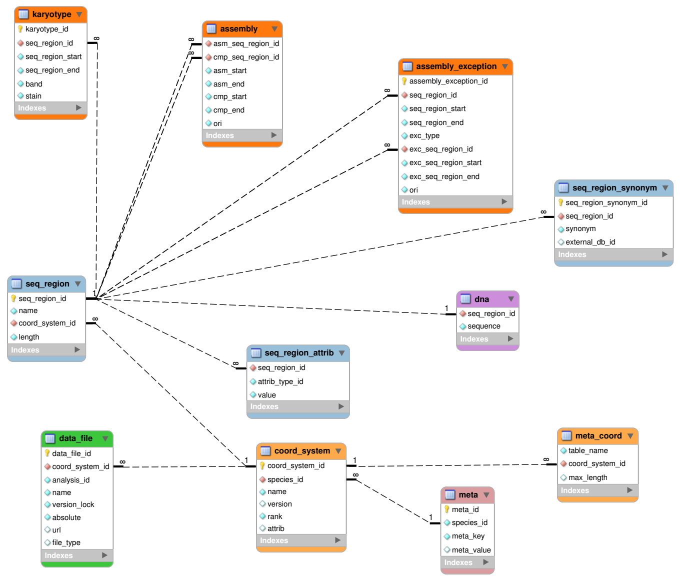
Query MySQL databases directly

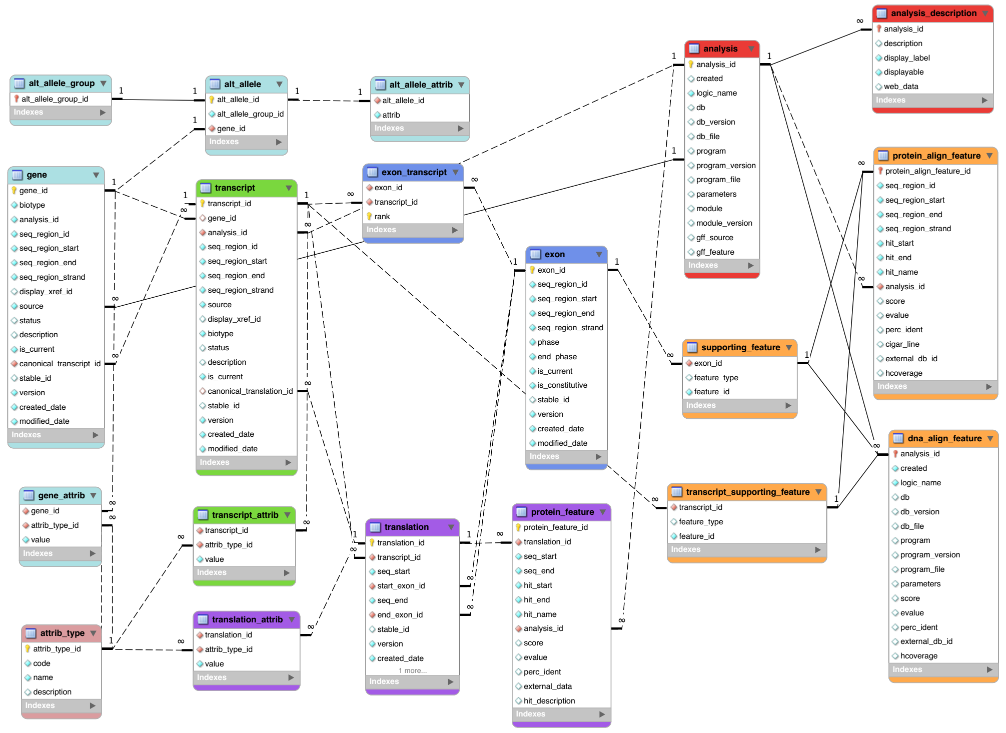
- UCSC
  - Hostname genome-mysql.cse.ucsc.edu
  - User genome
  - Password
- Gene Ontology
  - Hostname mysql-amigo.ebi.ac.uk
  - User go_select
  - Password amigo
  - Database go_latest
  - Port 4085
- Ensembl
  - Hostname ensembldb.ensembl.org
  - User anonymous
  - Password

Databases in bioinformatics

Ensembl database

- Complex database schemas
- Not suited to retrieve sequences





Databases in bioinformatics

API

- Uniform method of access to data
- Reusable in different systems
- Reliable
- Insulates developers to underlying database changes

Databases in bioinformatics

Ensembl API

- Perl API
- Installation instructions on Ensembl website
- Different versions based on Ensembl release
- Use Registry to find Ensembl database and connect to them

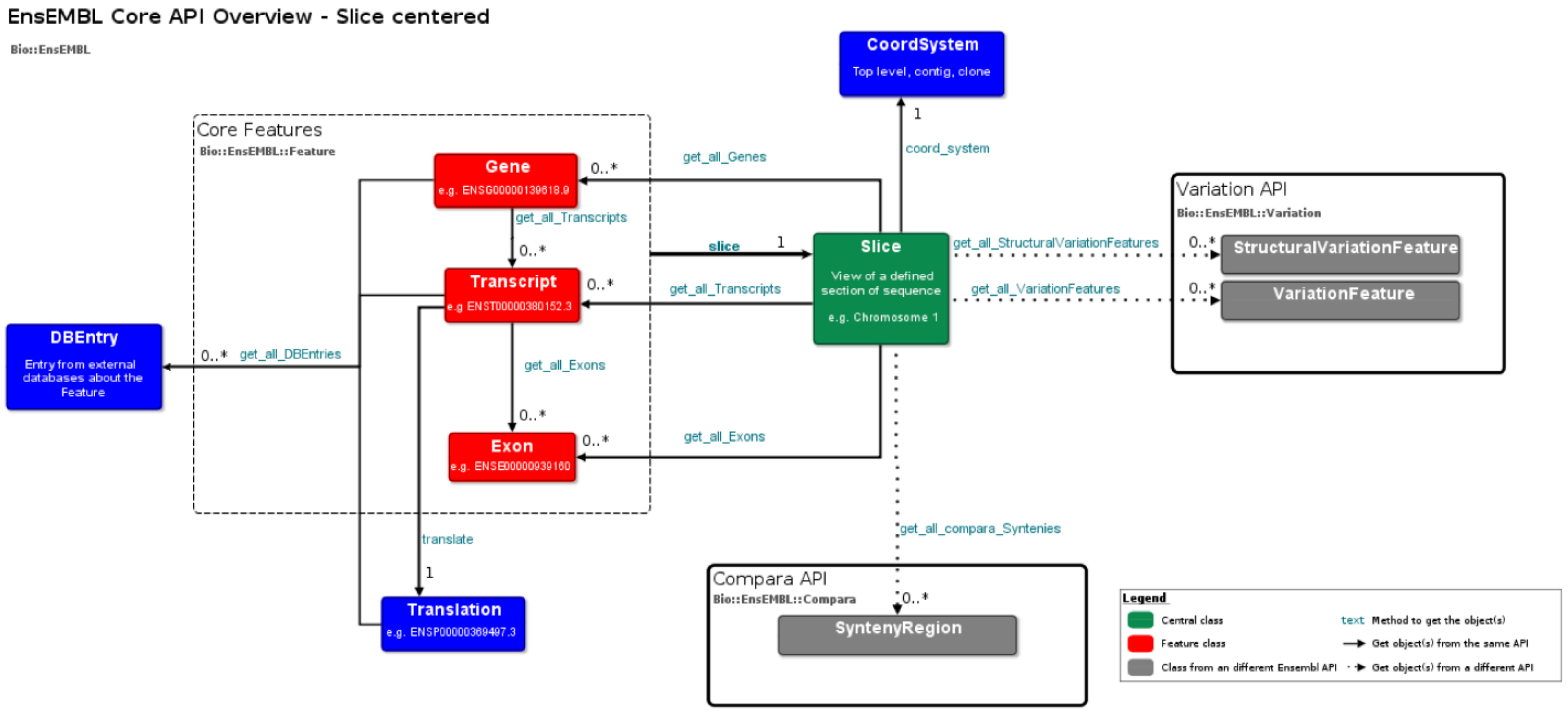
```
Bio::EnsEMBL::Registry->load_registry_from_db(
    -host => 'ensembldb.ensembl.org',
    -user => 'anonymous',
    -verbose => '1'
);
```

Databases in bioinformatics

Ensembl API

- Several databases
    - Core (genes, transcripts, translations, assembly, sequence)
    - Compara (SNVs, CNVs, somatic variations, phenotypes)
    - Variation (gene trees, homologies, multiple and pairwise genomic alignments)
    - Regulation (regulation, motifs, array probes)

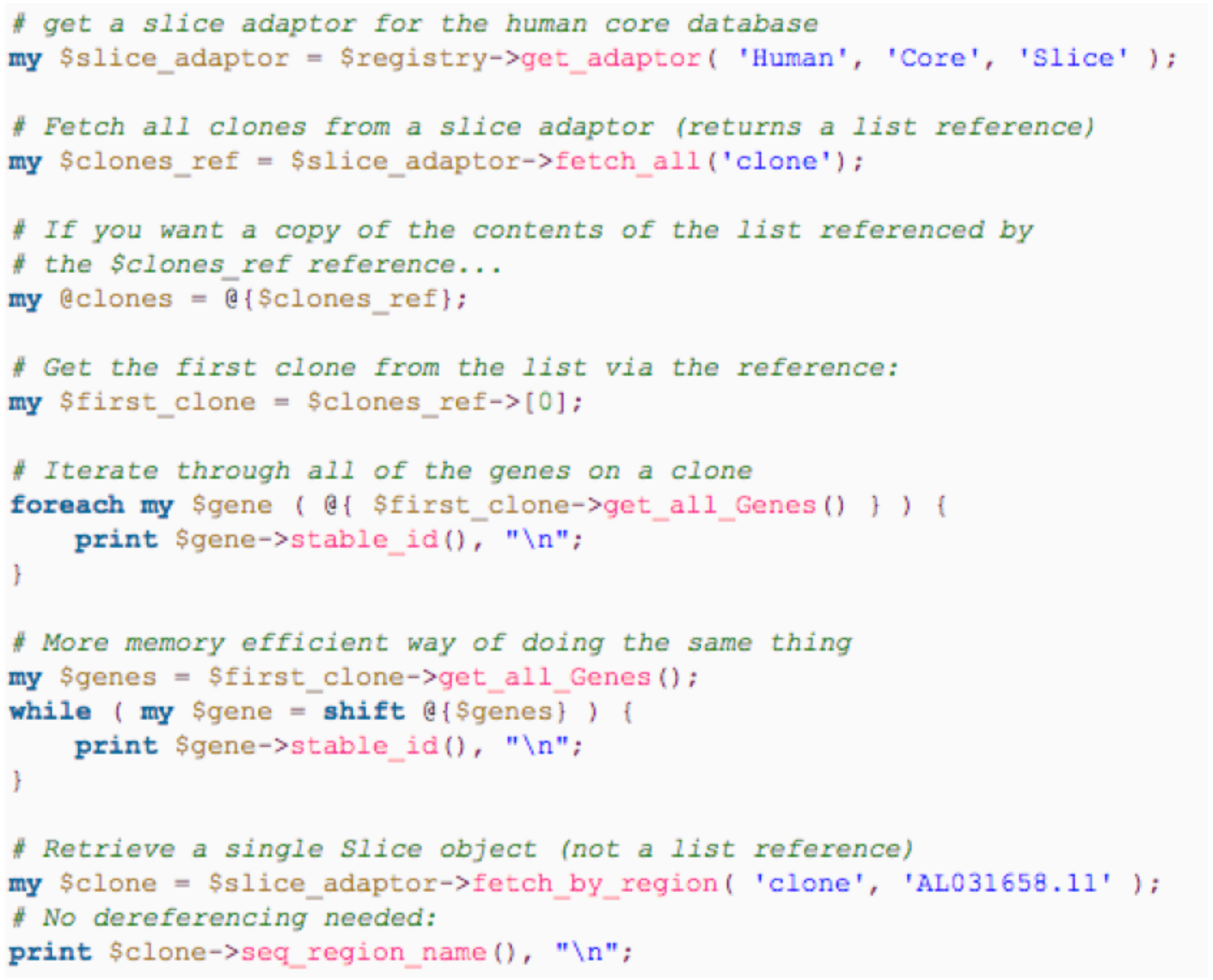
Databases in bioinformatics



Databases in bioinformatics

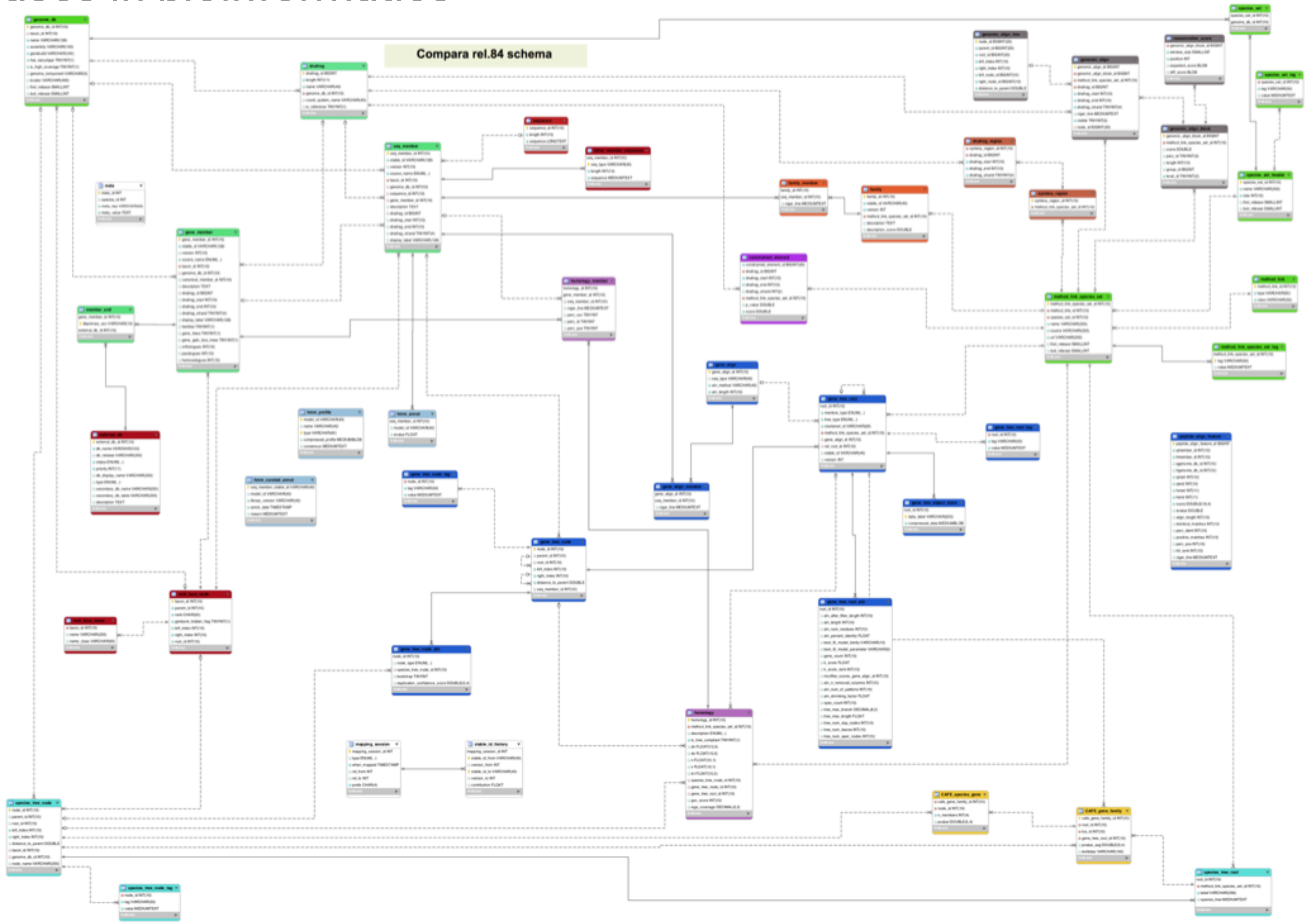
Ensembl API

- Core database
    - Annotation information for each organism in Ensembl
    - Species specific databases



```
# get a slice adaptor for the human core database
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );

# Fetch all clones from a slice adaptor (returns a list reference)
my $clones_ref = $slice_adaptor->fetch_all('clone');

# If you want a copy of the contents of the list referenced by
# the $clones_ref reference...
my @clones = @{$clones_ref};

# Get the first clone from the list via the reference:
my $first_clone = $clones_ref->[0];

# Iterate through all of the genes on a clone
foreach my $gene ( @{ $first_clone->get_all_Genes() } ) {
    print $gene->stable_id(), "\n";
}

# More memory efficient way of doing the same thing
my $genes = $first_clone->get_all_Genes();
while ( my $gene = shift @{$genes} ) {
    print $gene->stable_id(), "\n";
}

# Retrieve a single Slice object (not a list reference)
my $clone = $slice_adaptor->fetch_by_region( 'clone', 'AL031658.11' );
# No dereferencing needed:
print $clone->seq_region_name(), "\n";
```

Databases in bioinformatics

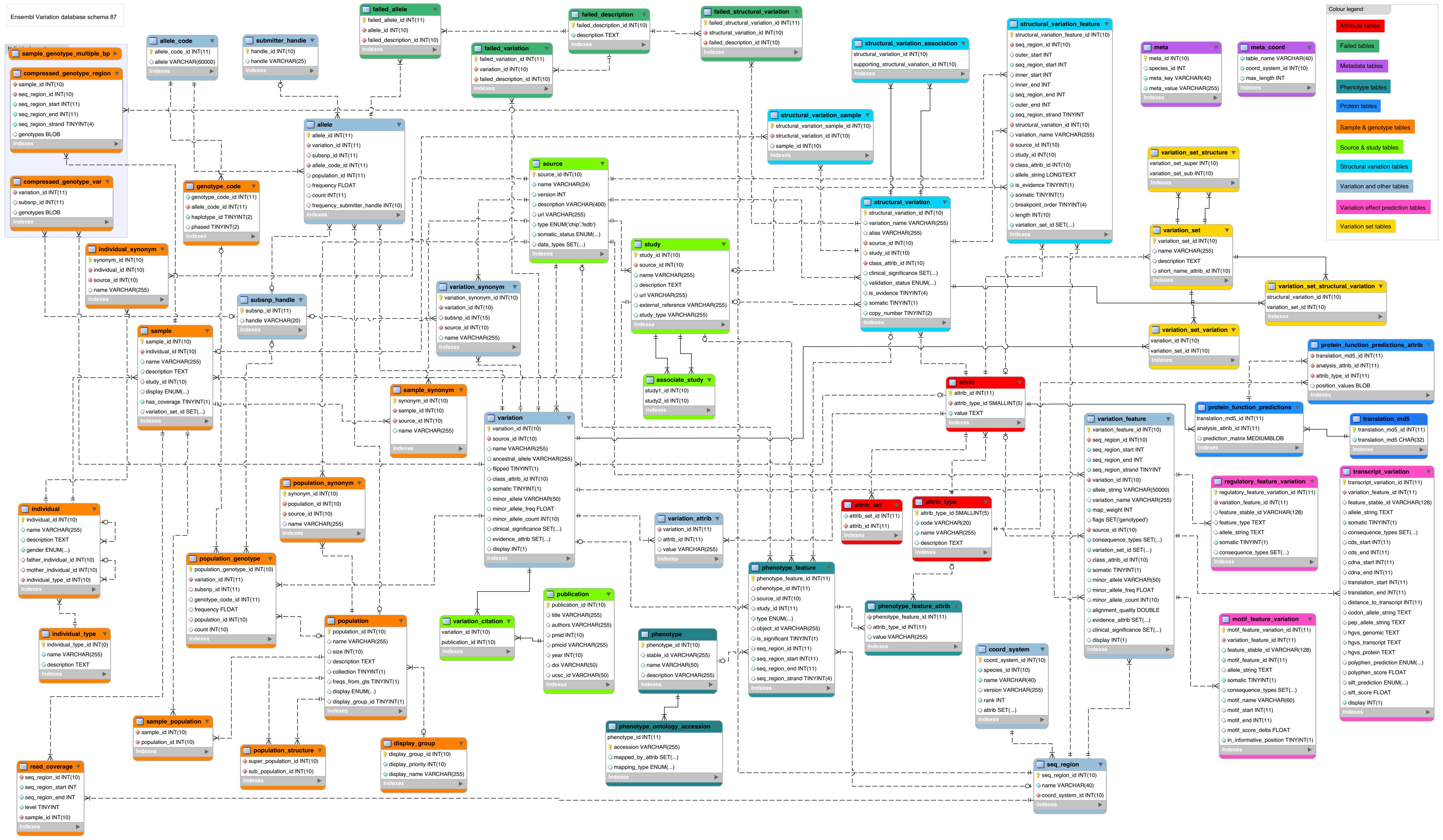


Databases in bioinformatics

Ensembl API

- Compara database
    - Cross-species database
    - Genome-wide species comparisons
        - DNA-sequence level
            - Whole genome alignments
            - Synteny regions
            - Conservation scores / constrained elements
        - Gene level
            - Phylogenetic trees
            - Homology predictions

Databases in bioinformatics

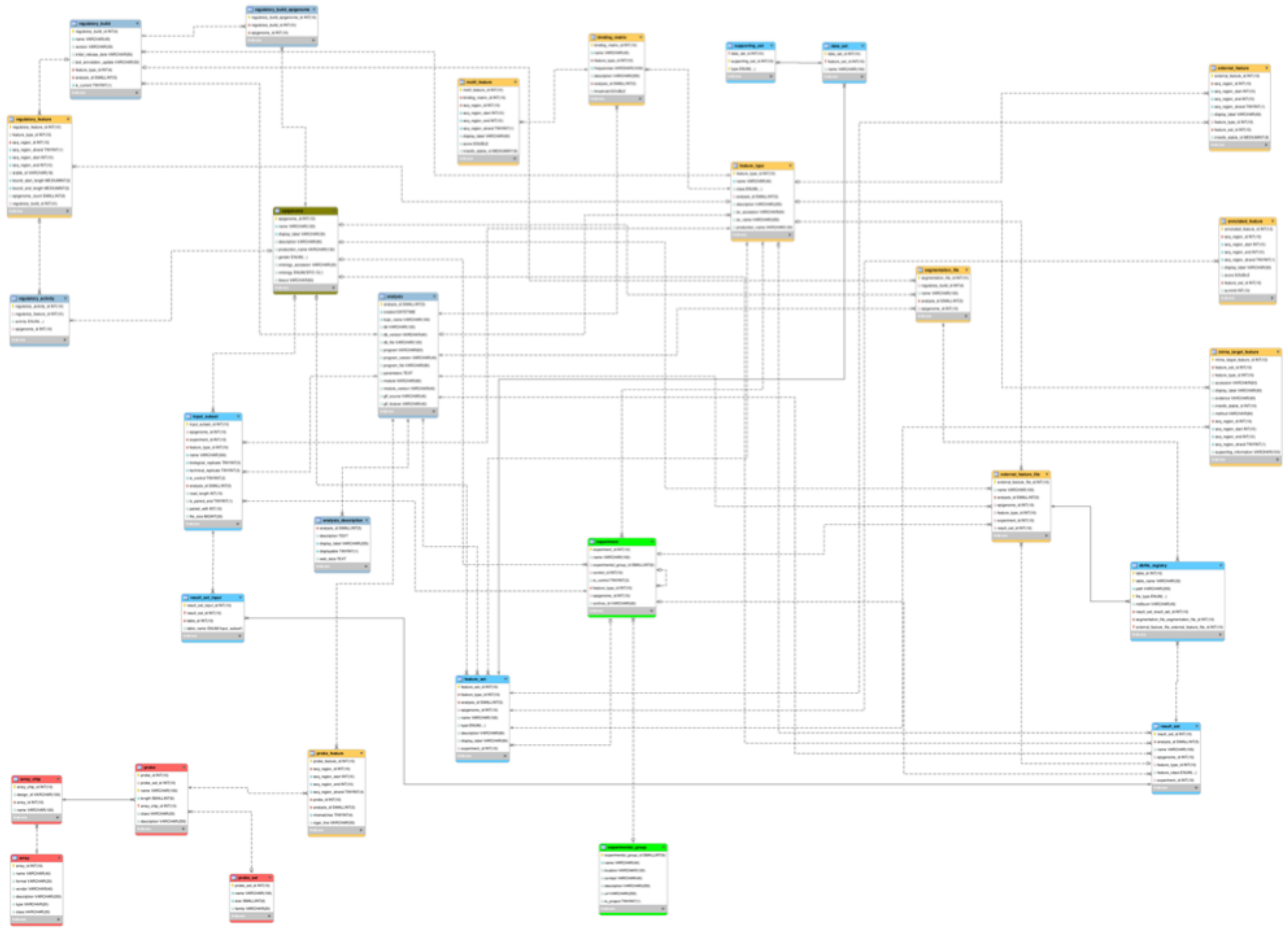


Databases in bioinformatics

Ensembl API

- Variation database
    - Areas of the genome that differ between individual genomes
    - Associated disease and phenothpe information
    - Different types of variants
        - Sequence variants
            - SNP (Single Nucleotide Polymorphism)
            - Insertion (one or more nucleotides)
            - Deletion (one or more nucleotides)
            - Indel (insertion and deletion, affecting 2 or more nucleotides)
            - Substitution (no change in length)
        - Structural variants
            - CNV (Copy Number Variation)
            - Inversion
            - Translocation

Databases in bioinformatics



Databases in bioinformatics

Ensembl API

- Regulation database
    - Gene expression and its regulation in human and mouse
    - Focus on transcriptional and post-transcriptional mechanisms

Databases in bioinformatics

REST API

- Representational state transfer / RESTful
    - Base URL
    - Internet media type
    - Standard HTTP methods
        - OPTIONS
        - GET: list or retrieve
        - PUT: replace or create
        - POST: create new entry
        - DELETE: remove

Databases in bioinformatics

Ensembl REST API

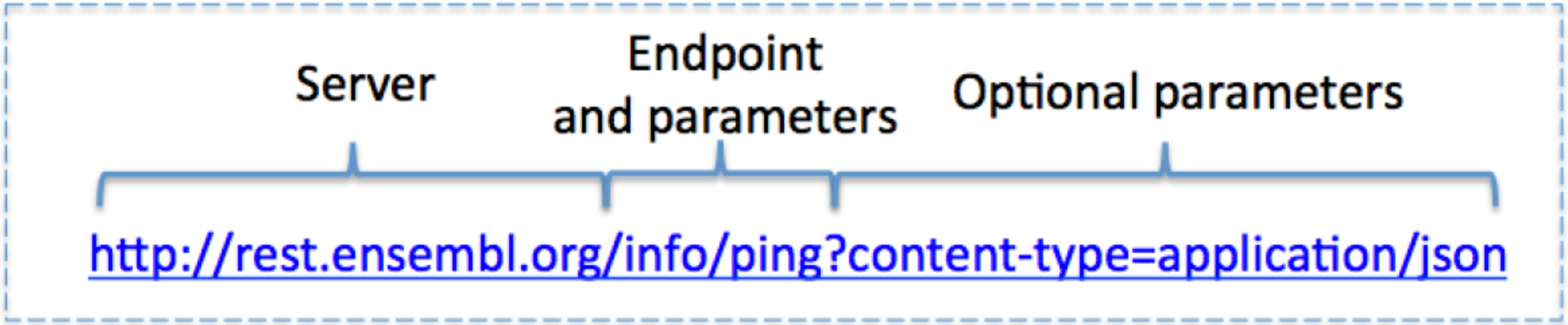
- http://rest.ensembl.org
- Language agnostic bindings to Ensembl data

- Able to create REST client in
  - JAVA
  - Perl
  - Python
  - Ruby

Databases in bioinformatics

Ensembl REST API

- URL structure
  - 0 or more required parameters
  - 0 or more optional parameters
  - In a standard URL required parameters are flagged with a : e.g. :species.
  - Optional parameters should go into the request body if performing a POST or as key value pairs after the ? if performing a GET.


Server
Endpoint and parameters
Optional parameters
http://rest.ensembl.org/info/ping?content-type=application/json

Databases in bioinformatics

Ensembl REST API

- Parameters
    - Specify what is required and type of returned data from REST API

    - id
    - region
    - species
    - symbol
    - external_db
    - object_type
    - callback

Databases in bioinformatics

Ensembl REST API

- Output formats
    - JSON, FASTA, BED, XML, …
    - Depends on client and operation
        - GET
            - `Content-type` HTTP header
            - `Content-type` HTTP parameter
            - `Accept` HTTP header
            - File extension
        - POST
            - `Accept` HTTP header

Databases in bioinformatics

Ensembl REST API – Endpoints

- Archive

`GET archive/id/:id`	Uses the given identifier to return the archived sequence
`POST archive/id`	Retrieve the archived sequence for a set of identifiers

- Comparative genomics

`GET genetree/id/:id`	Retrieves a gene tree for a gene tree stable identifier
`GET genetree/member/id/:id`	Retrieves the gene tree that contains the gene / transcript / translation stable identifier
`GET genetree/member/symbol/:species/:symbol`	Retrieves the gene tree that contains the gene identified by a symbol
`GET alignment/region/:species/:region`	Retrieves genomic alignments as separate blocks based on a region and species
`GET homology/id/:id`	Retrieves homology information (orthologs) by Ensembl gene id
`GET homology/symbol/:species/:symbol`	Retrieves homology information (orthologs) by symbol

Databases in bioinformatics

Ensembl REST API – Endpoints

- Variation

`GET variation/:species/:id`	Uses a variant identifier (e.g. rsID) to return the variation features including optional genotype, phenotype and population data
`POST variation/:species`	Uses a list of variant identifiers (e.g. rsID) to return the variation features including optional genotype, phenotype and population data

- Sequence

`GET sequence/id/:id`	Request multiple types of sequence by stable identifier. Supports feature masking and expand options.
`POST sequence/id`	Request multiple types of sequence by a stable identifier list.
`GET sequence/region/:species/:region`	Returns the genomic sequence of the specified region of the given species. Supports feature masking and expand options.
`POST sequence/region/:species`	Request multiple types of sequence by a list of regions.

Databases in bioinformatics

Exercices

- Return the archived sequence with Ensembl id *ENSG00000141510*
- Return the archived sequence for both *ENSG00000012048* and *ENSG00000136997*
- Return a condensed XML-list of all orthologues in Mus musculus for *ENSG00000159763*
    - Do the same for BRCA2

  HINTS: *type=orthologues*
  target_taxon=<taxon_id>
  format=condensed
- Retrieve the genomic FASTA sequence for *ENST00000288602.10*
- Get a sequence from 100 nucleotides located on human chromosome 2 starting at position 100000
- Show the taxonomy information of the mouse
- Find the species and the database for *ENSMUSG00000059552*
- Return the length of following chromosomes in human and mouse
    - 2
    - 7
    - X
    - Which are the longest?

Version control

GIT – Track and store revisions/versions of files

- Help
  ```
  $ git help [<git_command>]
  ```
- Configuration
  ```
  $ git config
  ```

More info: [Git-it]

Version control

GIT – Track and store revisions/versions of files

- Initialize
```
$ git init
```
- Show status
```
$ git status
```
- Track files
```
$ git add <filename>
```
- Commit changes
```
$ git commit [-m "<commit_message>"]
$ git reset
```
- Show logs
```
$ git log
```
- Checkout a commit
```
$ git checkout <checksum>
```
- Show differences between revisions
```
$ git diff [<checksum1> [<checksum2>]]
```

Version control

GIT – Track and store revisions/versions of files

- Branching
  ```
  $ git branch
  $ git branch <‘new_branch’>
  ```
- Merging
  ```
  $ git merge <new_branch>
  ```
  - Merge conflicts: same file modified on 2 seperate branches
- Delete branch
  ```
  $ git branch -d <new_branch>
  ```
- Remotes
  - GitHub (public repositories)
- Clone repository
  ```
  $ git clone <repository_name> <local_dir>
  ```
- Update repository
  ```
  $ git pull <remote_name> <branch_name>
  ```
- Submit changes
  ```
  $ git push <remote_name> <branch_name>
  ```

Version control

Exercises

- Create a directory *`db_git`* and copy some of the course files to this new directory
- Create a git repository in this directory
  - Make sure your user_name and user_email are set correctly (HINT: `git config`)
- Commit all `.sql` files and all other files with two different commit messages
  - Check your commit history
- Add a README file to your repository
- Create a second branch in your repository
  - Change to this new branch
  - Add and delete some files
  - Add some lines to your README file
  - Show the differences between your 2 branches
- Include the changes from your new branch into your original branch
- Delete your second branch

Version control

Exercises

- Go to https://github.com/ and create a new repository *`db_github`*
- Add the contents of your existing *`db_git`* repository to your newly created remote one.
- Check the results