Docente: **Matteo Re**

UNIVERSITÀ DEGLI STUDI DI MILANO

C.d.l. Informatica

Bioinformatica

A.A. 2013-2014 semestre I

p6 Interrogazione diretta di banche dati biologiche - SQL

Obiettivi

- **Interrogazione diretta di banche dati biologiche**
  - Accesso mediante Perl
  - Linguaggio SQL
- **Biologia computazionale**
  - Struttura (db schema) Ensembl database
  - API Ensembl
  - Estrazione di annotazioni
  - Estrazione di sequenze

Linee guida

- **Il livello di complessità di questa esercitazione è medio-alto**
  - Cercate di risolvere il problema dopo aver compreso gli schemi dai database presentati
  - I template script di questa esercitazione sono estremamente semplici … non fatevi ingennare da questa apparente semplicità **la difficoltà dell'esercizio risiede nella necessità di costruire le interrogazioni in linguaggio SQL** e di itegrarle in maniera opportuna negli script. Come sempre il codice che mi invierete DEVE essere commentato (in questo caso il commento riguarderà principalmente le query SQL).

- **Modalità di svolgimento dell'esercitazione:**
  - Nessun file da scaricare questa volta … lo script di base per effettuare le query SQL è molto contenuto ed è riportato in queste slide.
  - Lo stesso vale per gli esercizi sulle API Ensembl core (trovate molti più esempi risolti mediante le API che mediante SQL… Questo dipende dal fatto che la difficoltà intrinseca degli esercizi SQL sta nella necessità **di dover esplorare lo schema della banca dati Ensembl**).

Tipi di banche dati biologiche:

Collettori primari:

Sequenze sottomesse direttamente dai laboratori di ricerca alle banche dati Genbank, DDBJ ed EMBL. Qualità bassa, a volte contengono errori di annotazione.

Banche dati secondarie:

Le informazioni contenute in queste banche dati sono curate manualmente: qualità superiore. Spesso sono banche dati specializzate nel senso che contengono un solo tipo di informazione (seq. proteiche, seq. di trascritti, …).

Banche dati associate a progetti genomici:

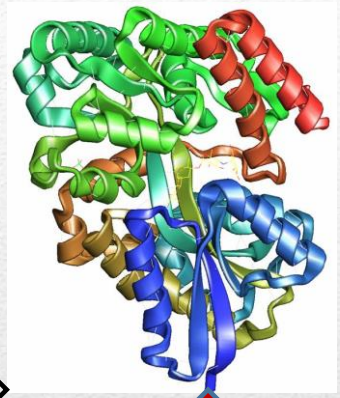
Le sequenze genomiche sono un tipo di dato molto particolare. Esse si prestano ad essere annotate a diversi livelli. A causa di questa caratteristica la loro annotazione richiede l'utilizzo di informazioni derivanti da un numero consistente di banche dati esterne. Come conseguenza le banche dati associate a progetti di annotazione genomica sono gli strumenti di elezione per **INTEGRARE** il contenuto di altre banche dati in modo da ottenerne una **rappresentazione unitaria**.

Tipi di dati biologici (solo alcuni)

- **Livello dei trascritti** misurati in particolari condizioni: esistono siti dedicati a collezioni di esperimenti microarray (es. NCBI Gene Expression Omnibus (NCBI GEO), http://www.ncbi.nlm.nih.gov/geo/)
- **Annotazione funzionale di proteine**: «funzionale» viene utilizzato come termine a «**basso**» livello, annotazione di una sequenza proteica **residuo per residuo**. Molti tipi di annotazione: siti di fosforilazione, presenza di ponti disolfuro, struttura secondaria della proteina, struttura 3D della proteina. Sito di riferimento è una banca dati che integra le informazioni di diverse banche dati: Uniprot (Universal Protein Resource, http://www.uniprot.org/) .
- **Annotazione funzionale di geni:** «funzionale» viene utilizzato come termine ad «**alto**» livello. Creazione di vocabolari controllati a partire da materiale reperibile in **LETTERATURA**. Team di curatori assegnano ogni gene ai termini dei vocabolari (**ontologie**). Sito di riferimento: Gene Ontology (http://www.geneontology.org/) .
- **Variabilità genetica:** Database dedicati a SNP (es. NCBI dbSNP) e a progetti su vasta scala (HapMap). Esistono inoltre databases dedicati a studi di associazione genome-wide (es GWAS central) http://www.gwascentral.org/index.

E MOLTI ALTRI …

Esistono molti tipi di banche dati … perché quelle associate a progetti di annotazione genomica dovrebbero essere considerate più «importanti» di altre?



fuctional annotation

post tanslational editing

post transcriptional editing

folding

translation

transcription

Proteins DB

cross links

Transcripts DB

alignment

alignment

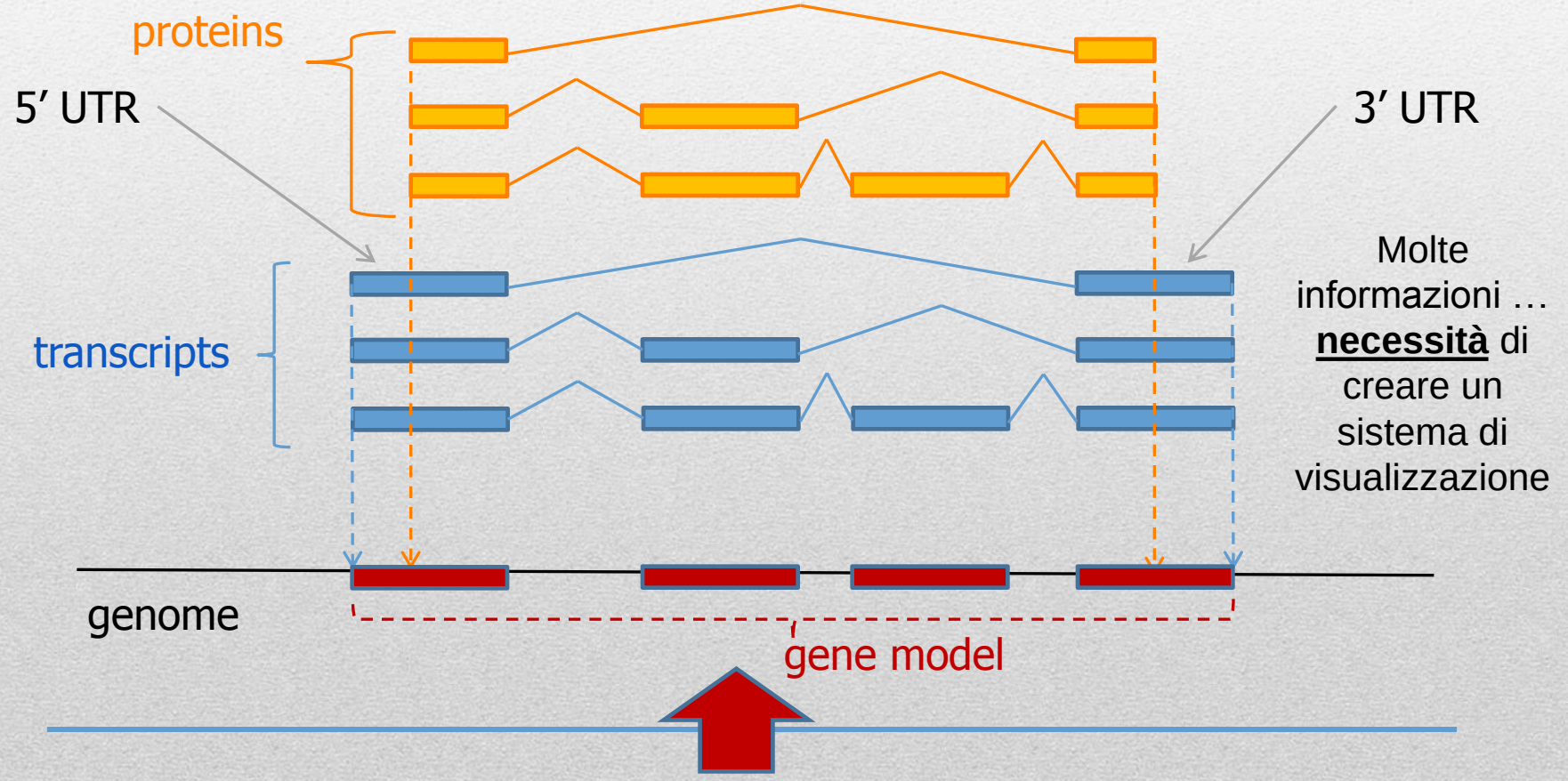
genome

regulatory region

regulatory motifs

Annotazione in coordinate genomiche

Se «proiettiamo» tutte le informazioni disponibili (seq. espresse, seq. proteiche, motivi regolatori ecc.) sul genoma rendiamo tali informazioni più semplici da consultare perché il genoma assume il ruolo di elemento di riferimento!

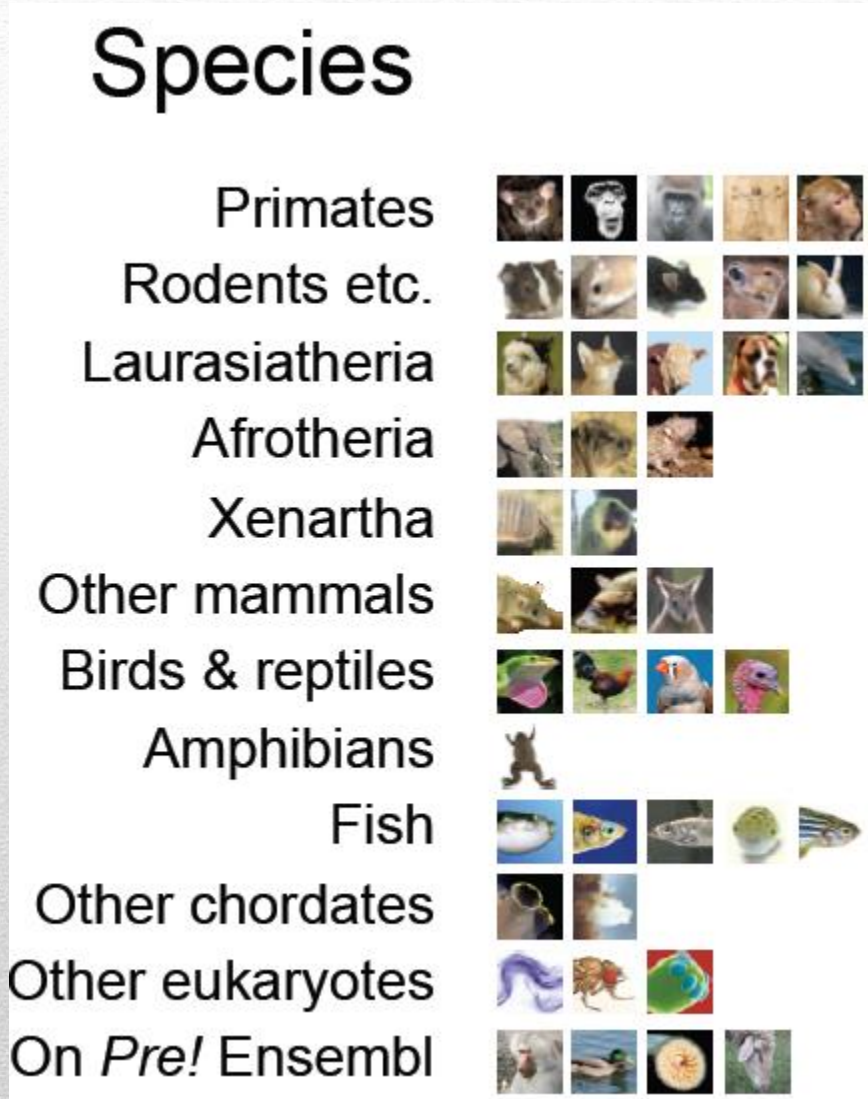


Annotazione in coordinate genomiche

BROWSERS GENOMICI

- **Ne esistono diversi:** Principalmente 3, NCBI map viewer (http://www.ncbi.nlm.nih.gov/projects/mapview/), Ensembl (http://www.ensembl.org/index.html) e UCSC genome browser (http://genome.ucsc.edu/) .
- **Presentano le stesse informazioni, ma in modo diverso**: tutti e tre permettono di trovare la posizione genomica di una sequenza (mediante allineamento o ricerca per parola chiave) e di visualizzare la regione genomica associata.
- **I dati contenuti nei browser genomici dipendono dal contenuto di altre banche dati:** necessità di aggiornare i dati molto spesso. Ensembl viene aggiornato mensilmente .
- **Produzione dei dati di annotazione genomica:** E' un processo cosatoso dal punto di vista delle risorse di calcolo (allineamento di intere banche dati di sequenze al genoma). I principali browser genomici contengono più di un genoma (in realtà contengono molti genomi). E' un processo basato su **pipeline di annotazione automatizzate**.

Ensembl

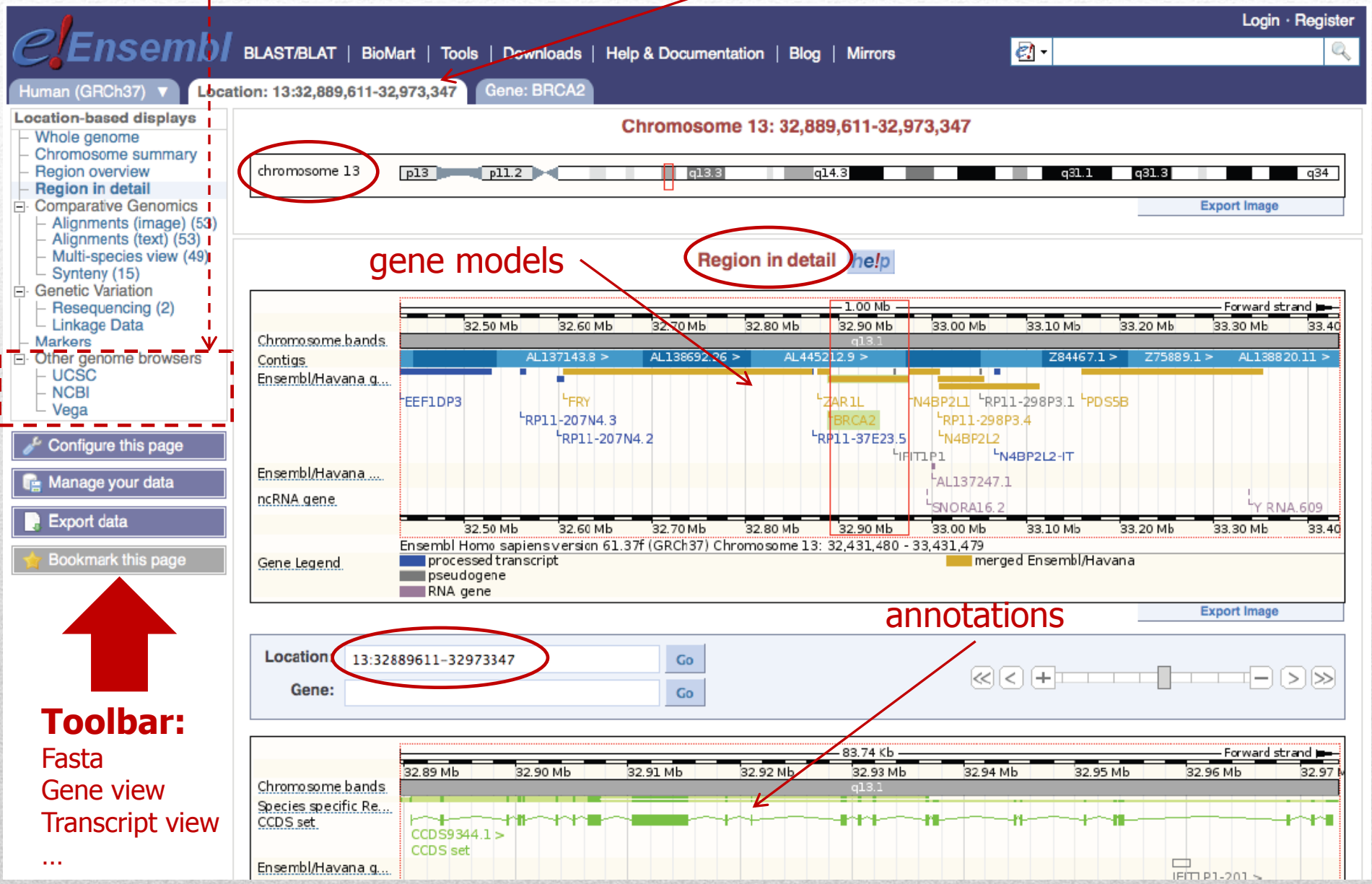


Caratteristica specifica di Ensembl :

contiene **modelli** di geni (altri browser utilizzano come entità fondamentale il trascritto o, comunque, la «sequenza allineata al genoma»)

external links

Ensembl: Genome view



Toolbar:

Fasta

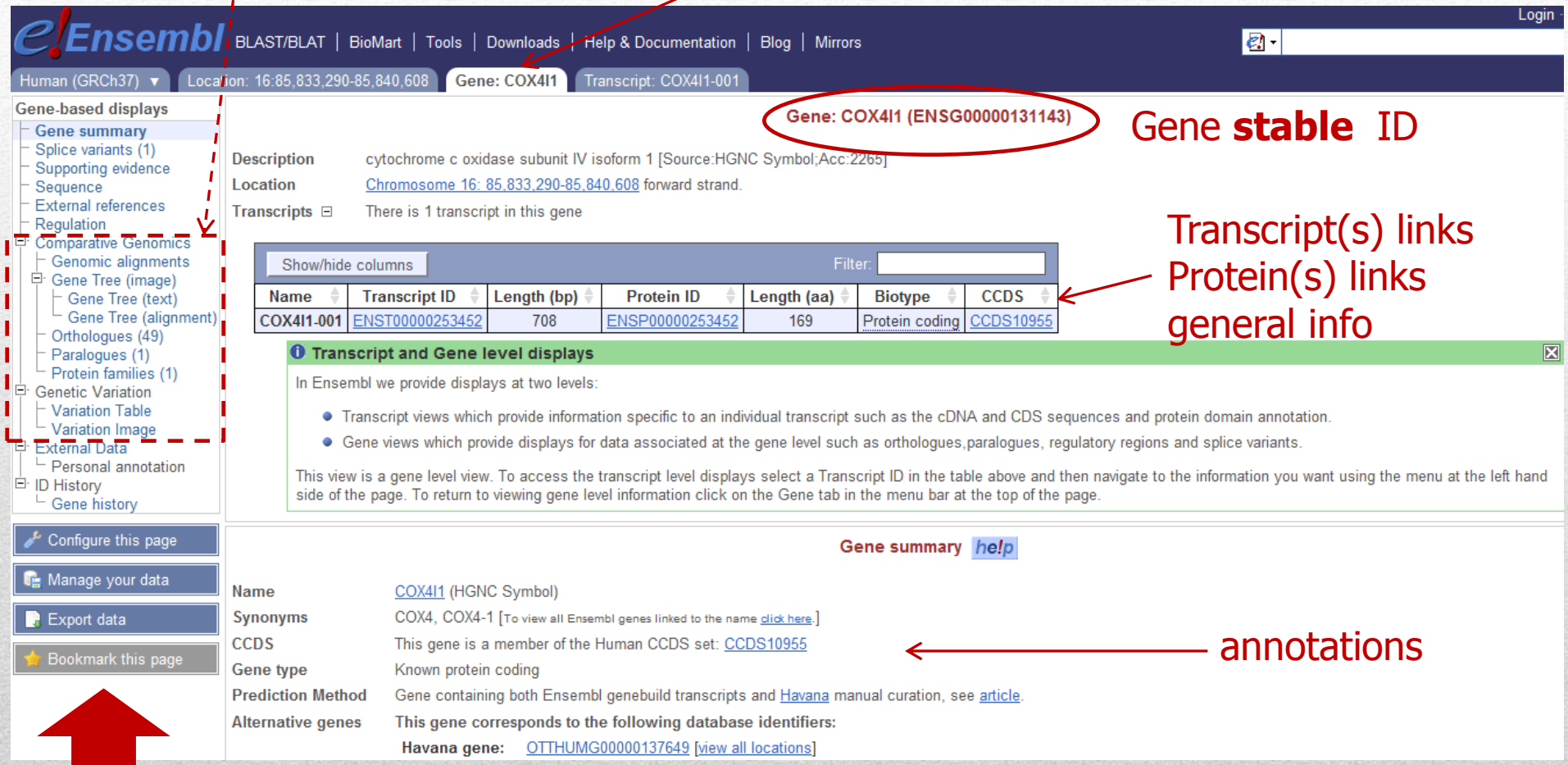
Gene view

Transcript view

...

external links

Ensembl: Gene view



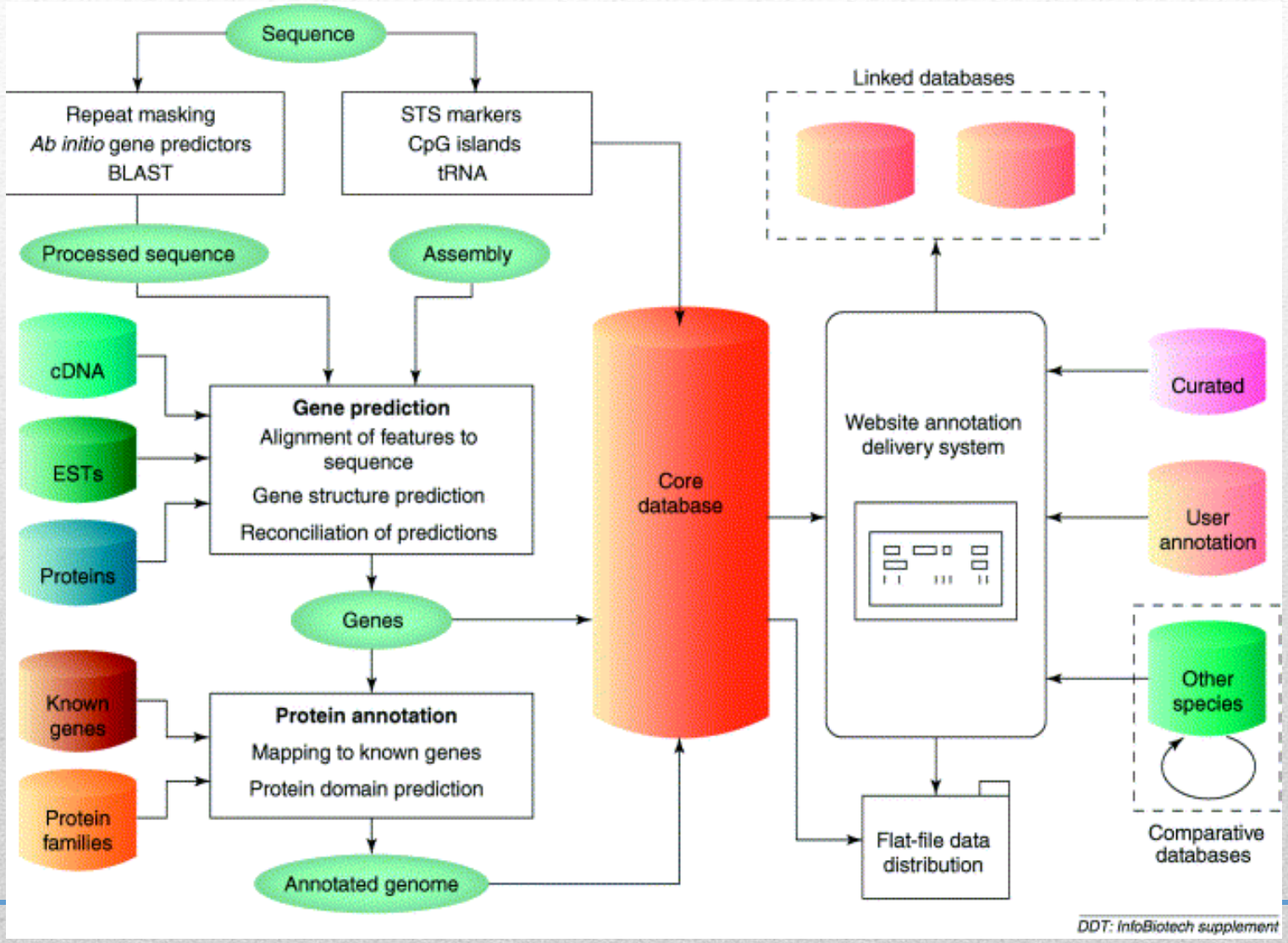
Gene **stable** ID

Transcript(s) links
Protein(s) links
general info

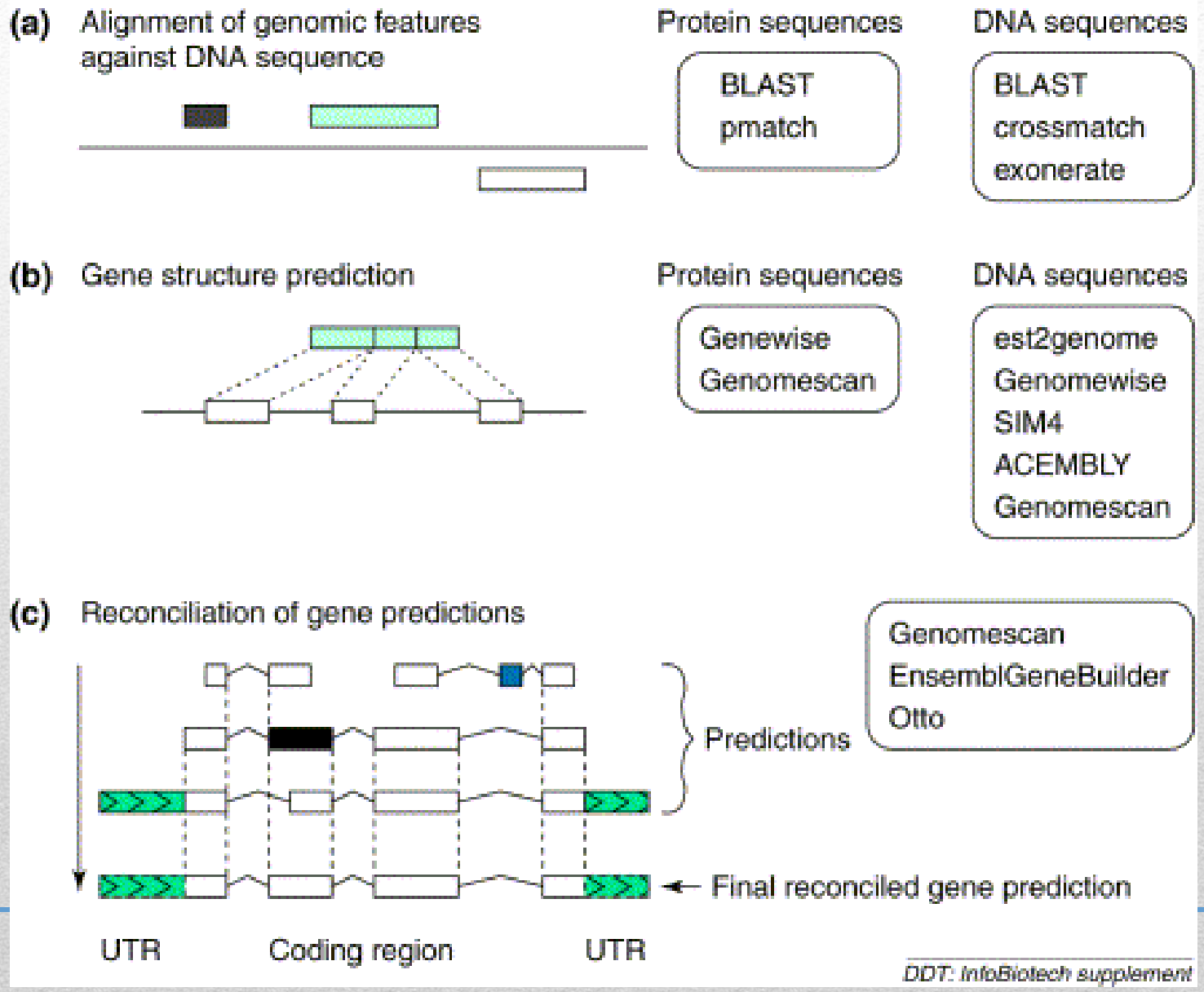
annotations

Toolbar:
Export seq.
Export annot.
...

Automatizzazione del processo di annotazione di una sequenza genomica

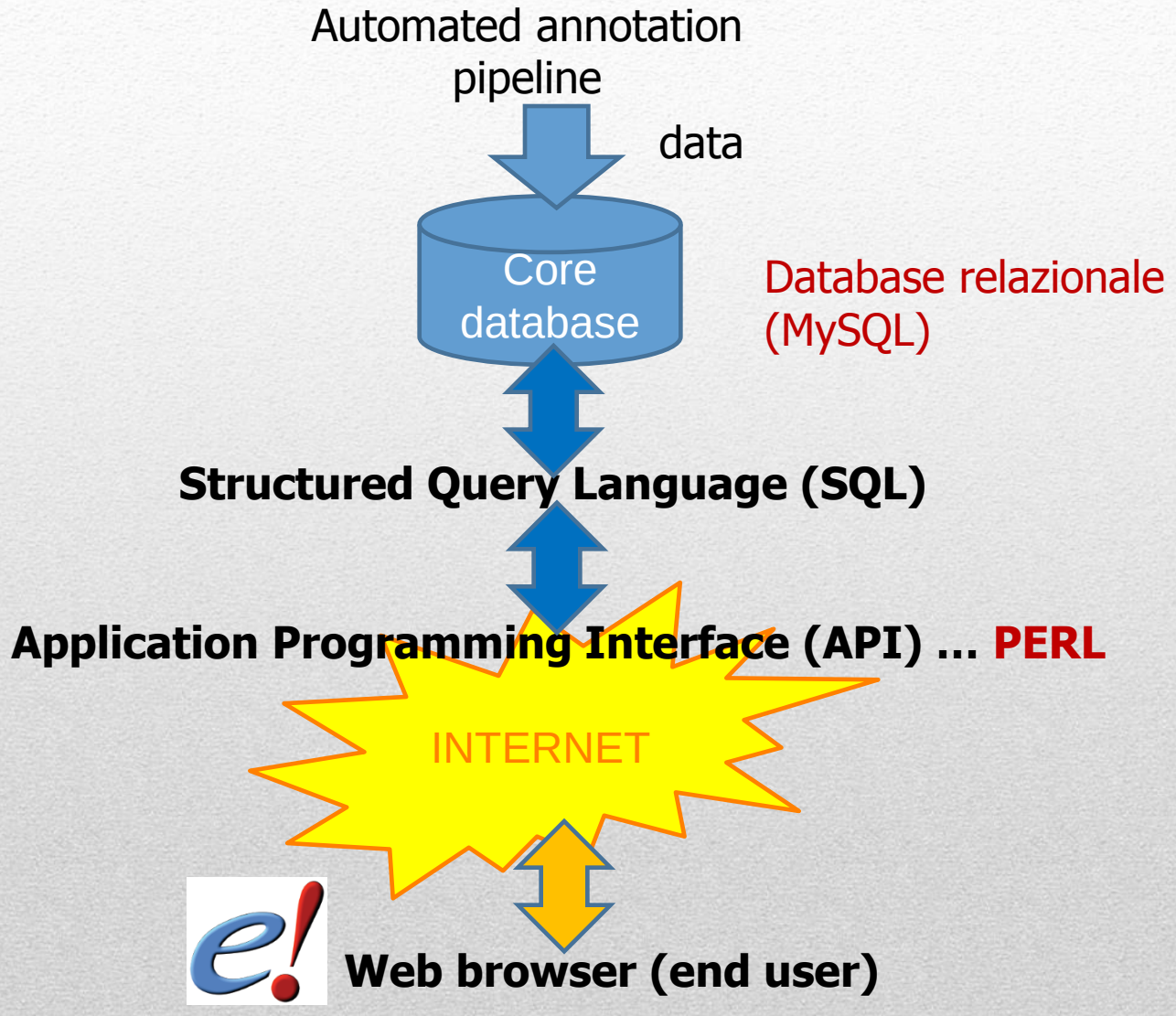


Creazione di «modelli» di geni

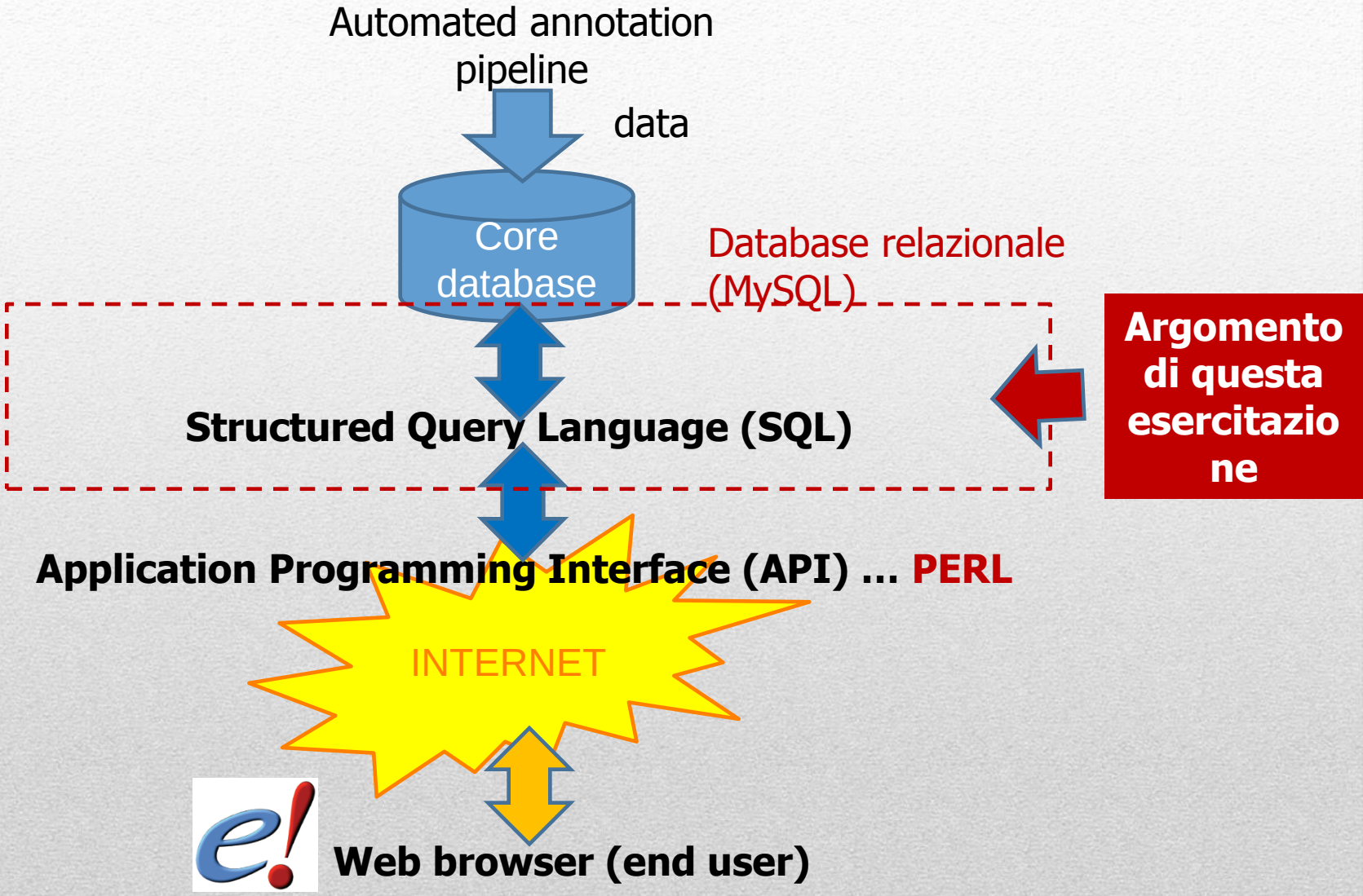


DDT: InfoBiotech supplement

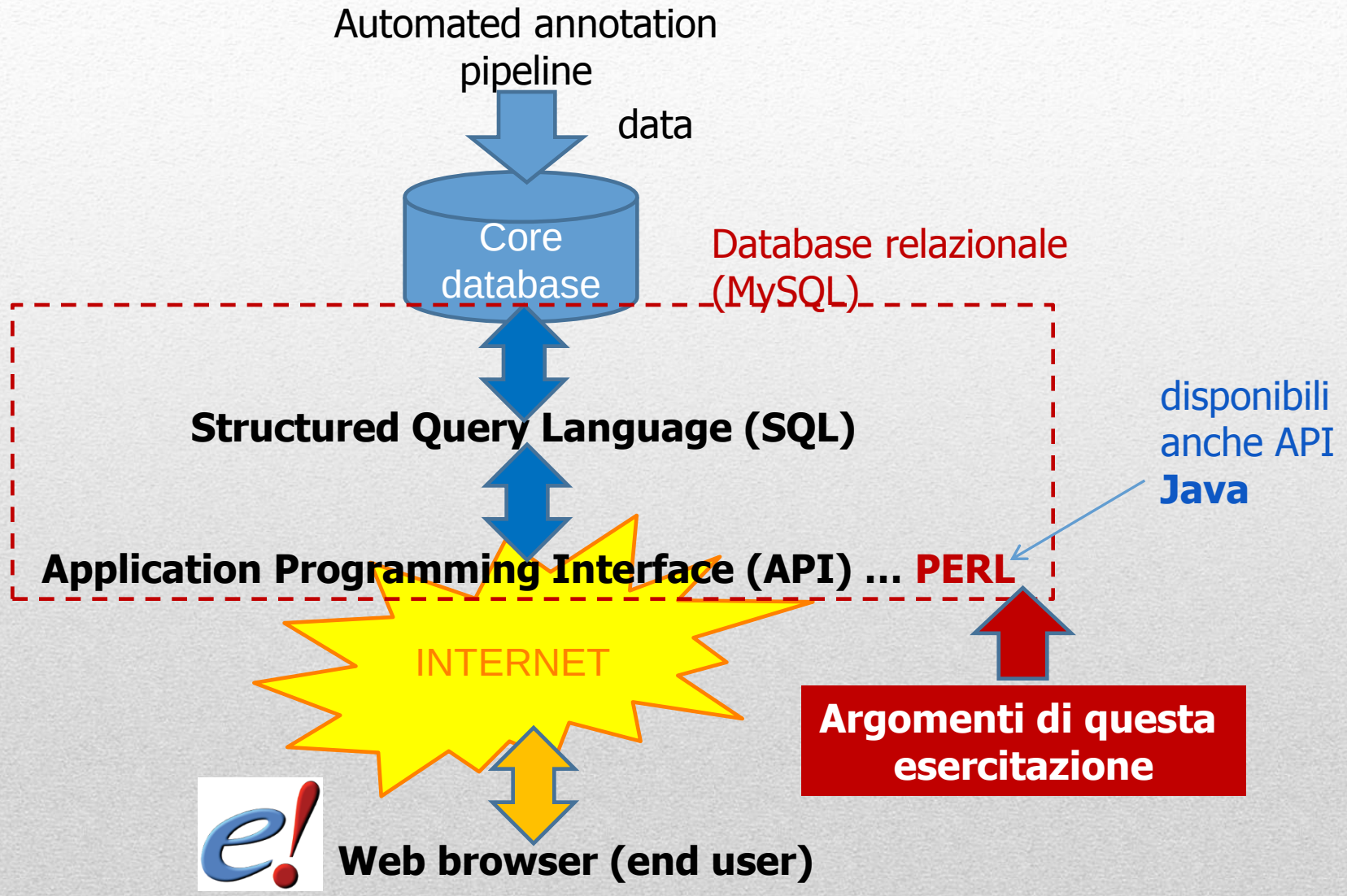
Architettura del browser genomico Ensembl



Architettura del browser genomico Ensembl



Architettura del browser genomico Ensembl



Structured Query Language (SQL) e (R)DBMS

I database sono estremamente eterogenei per quanto riguarda la loro struttura e la quantità di dati contenuta. Essi possono essere costituiti da file di testo ASCII o file che rappresentano complesse strutture composte da alberi binari (ad es. Oracle o Sybase). In ogni caso un database è un contenitore di dati.

PROBLEMA:

Se un database è una semplice collezione di dati … chi tiene traccia del cambiamento dei dati stessi?

Questo è il ruolo dei sistemi di gestione delle basi di dati (database management systems o DBMS). Alcuni DBMS sono relazionali. In tal caso ci si riferisce ad essi come relational DBMS o RDBMS. Le relazioni su cui si basano i sistemi RDBMS assicurano che diverse colelzioni di dati (ad es. tabelle) possano essere interrogate "all'unisono". Le relazioni, di fatto, rappresentano delle regole di integrità referenziale tra collezioni di dati. Supponiamo di avere un RDBMS che contiene I dati di tutti gli impiegati di un azienza e di avere 2 tabelle: reparto e impiegato. Tra di esse potrebbe esistere una relazione che permette l'inserimento di un nuovo impiegato SOLO se esso è assegnato ad un reparto esistente.

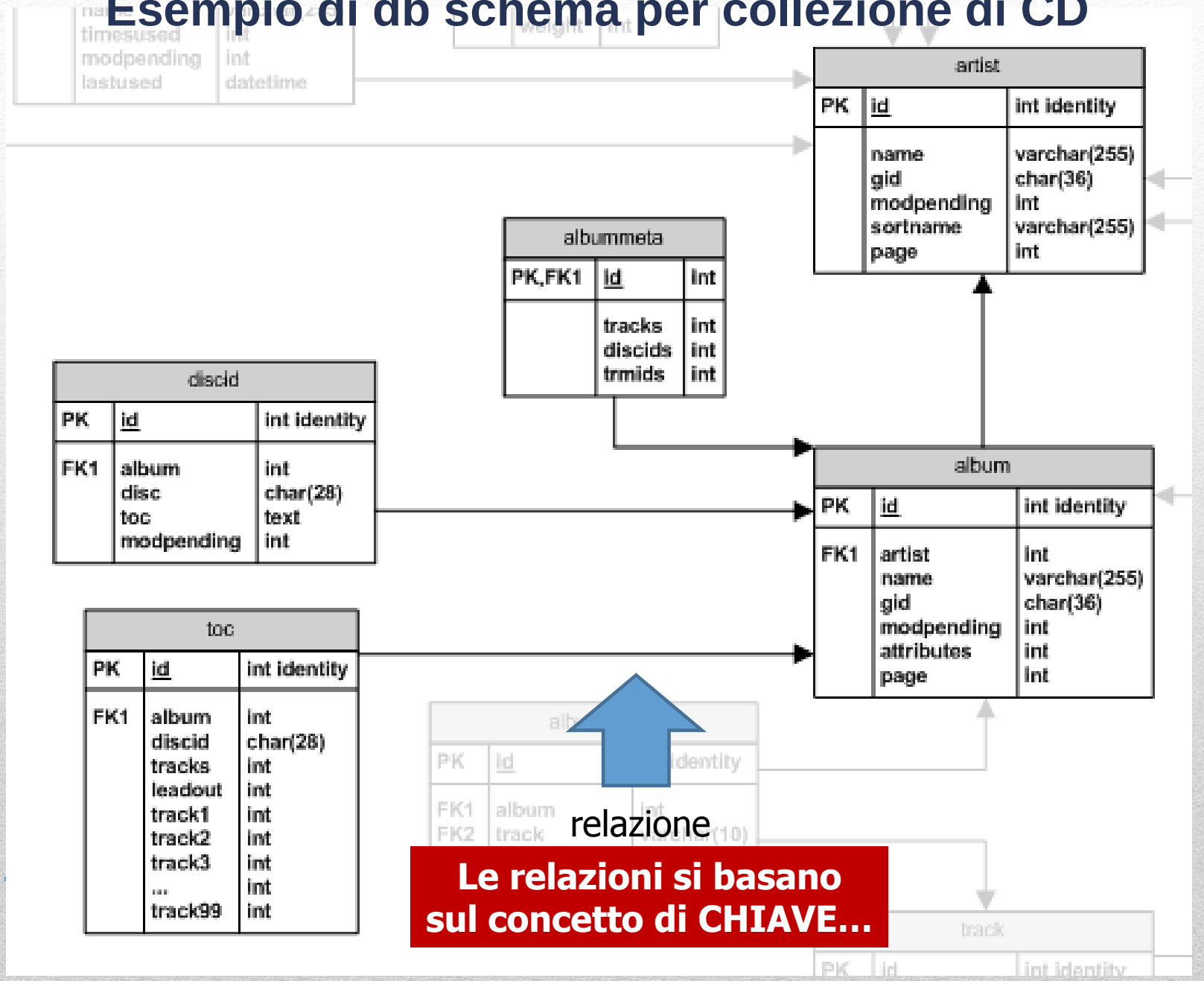
Structured Query Language (SQL) e (R)DBMS

Un database relazionale (come quello associato alla maggioranza delle banche dati genomiche) è costituito da :

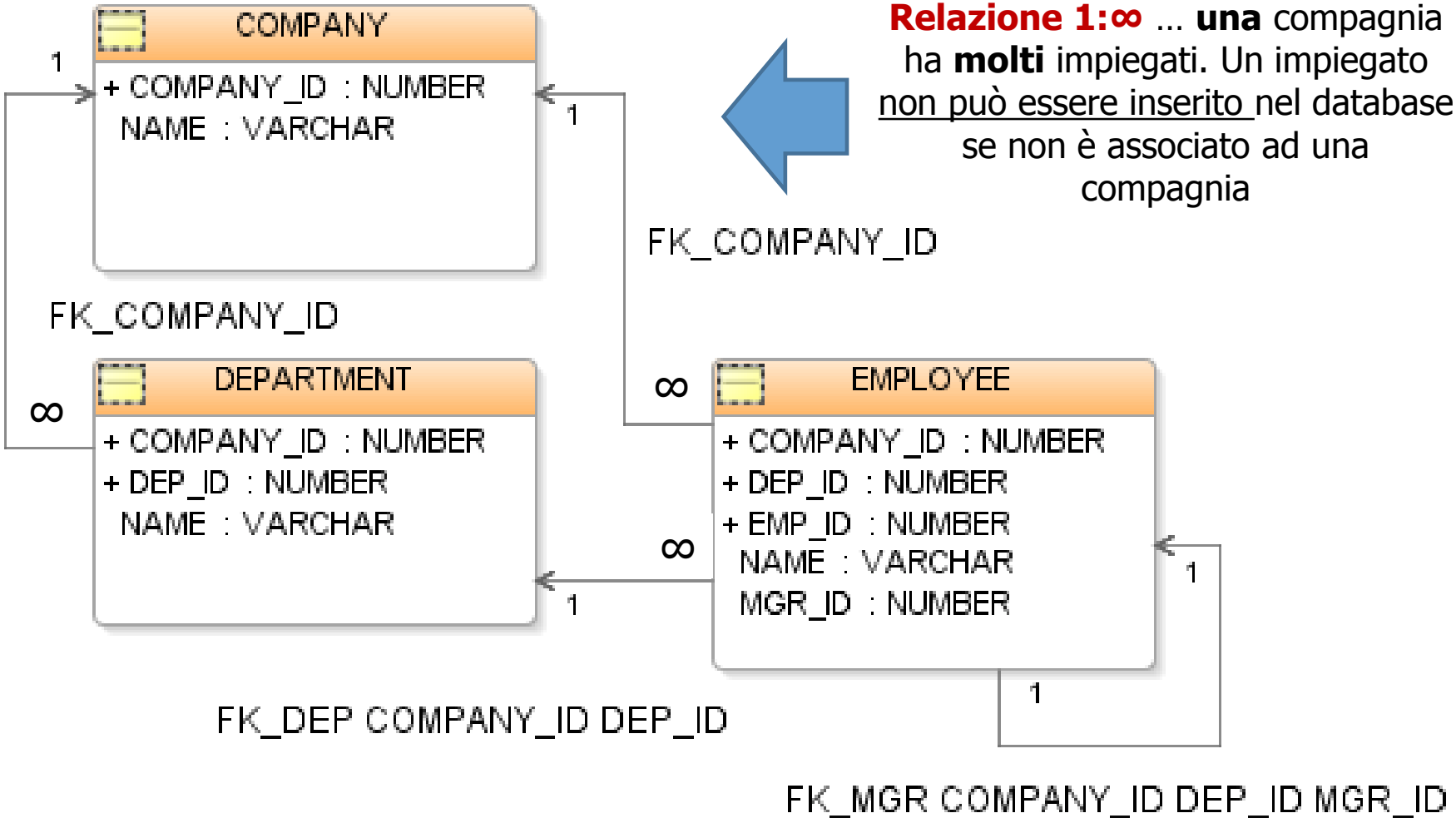
1. **Una parte INVARIANTE nel tempo detta database schema. Essa definisce la struttura logica delle unità di memorizzazione delle informazioni. Tale struttura, di solito, viene rappresentata sottoforma di tabella. La rappresentazione tabulare permette di nascondere I dettagli del formato reale di memorizzazione su disco.**

2. **I dati veri e propri: ad essi ci si riferisce con il termine generico di istanze. Per ogni tabella presente nella banca dati è disponibile una DEFINIZIONE composta da numero e nomi dei campi (colonne) della tabella, tipo di dato ammesso in ogni campo e altre caratteristiche (che descrivono ad esempio, il coinvolgimento di una relazione di integrità associata ad un dato campo). Prima di inserire una nuova riga in una tabella il sistema RDBMS verifica che la collezione di dati (la riga della tabella) tispetti tutte le specifiche della tabella stessa. Un altro modo comune di riferirsi alle righe delle tabelle è il termine RECORD.**

Esempio di db schema per collezione di CD



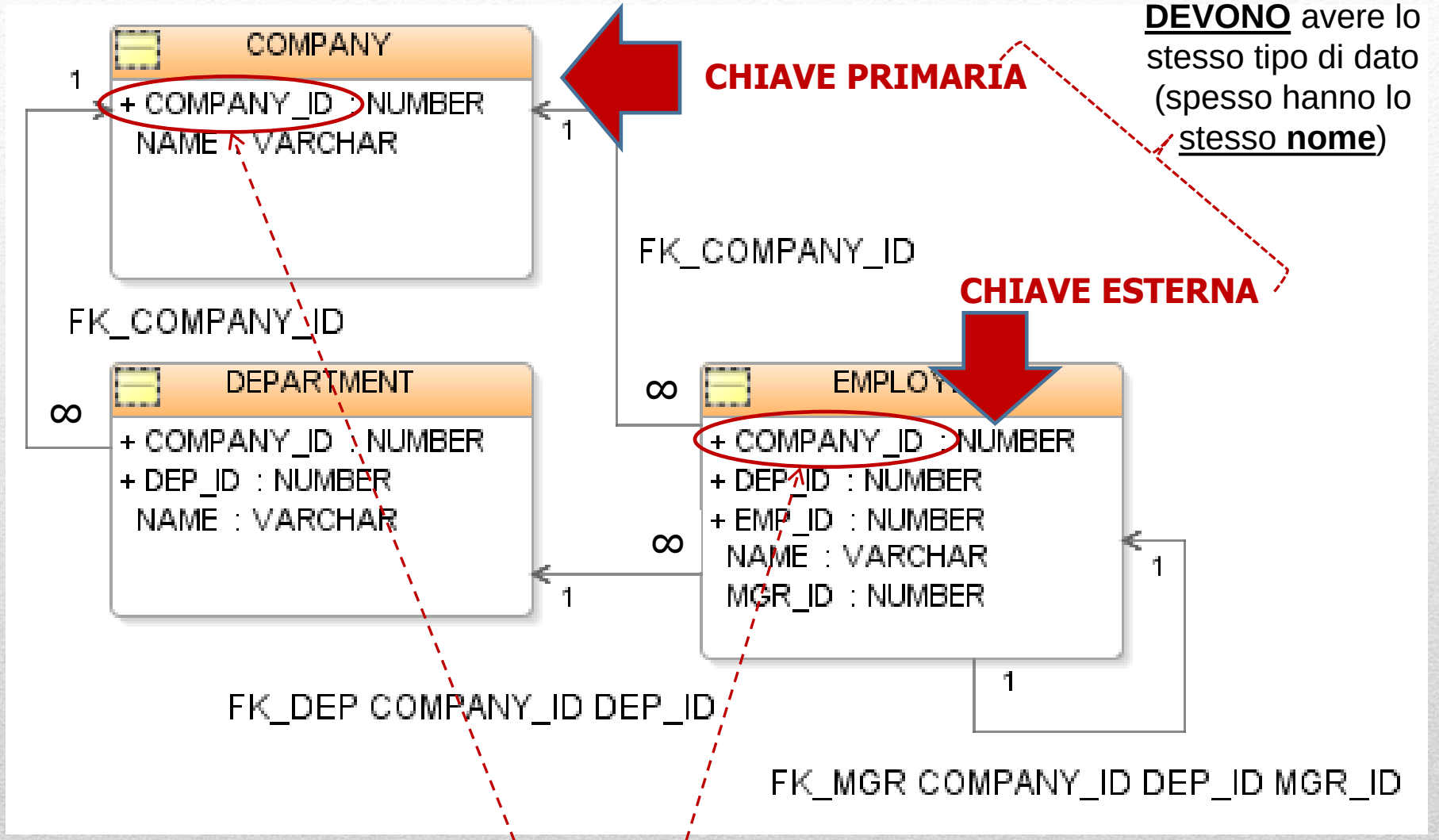
Esempio di db schema per collezione di CD



Relazione 1:∞ … **una** compagnia ha **molti** impiegati. Un impiegato non può essere inserito nel database se non è associato ad una compagnia

Le relazioni si basano sul concetto di CHIAVE…

Chiavi primarie e chiavi esterne



Questa relazione si basa su due campi:
COMPANY_ID (tabella COMPANY) e COMPANY_ID (tabella EMPLOYEE)

Interazione con RDBMS e ruolo di SQL

E’ necessaro uno strumento che permetta di interagire con la banca dati. Questo ruolo è svolto da un linguaggio standardizzato detto Structured Query Language (SQL). SQL permette non solo l’estrazione dei dati ma anche la creazione/modifica di database e tabelle nonchè la definizione di vincoli relazionali. SQL si divide in:

- **DATA DEFINITION LANGUAGE (DDL): linguaggio di definizione dei dati, serve per creare databases, definizioni di tabelle e vincoli di integrità referenziale. Permette inoltre di modificare la struttura di tabelle esistenti.**
- **DATA MANIPULATION LANGUAGE (DML): insieme di enunciati che permettono, principalmente, di estrarre informazioni da una banca dati.**

A noi interessa DML (DDL non verrà trattato)

Operazioni realizzabili mediante SQL DML

SQL DML permette di realizzare diverse operazioni che possono essere attribuite a tre grandi macrocategorie:

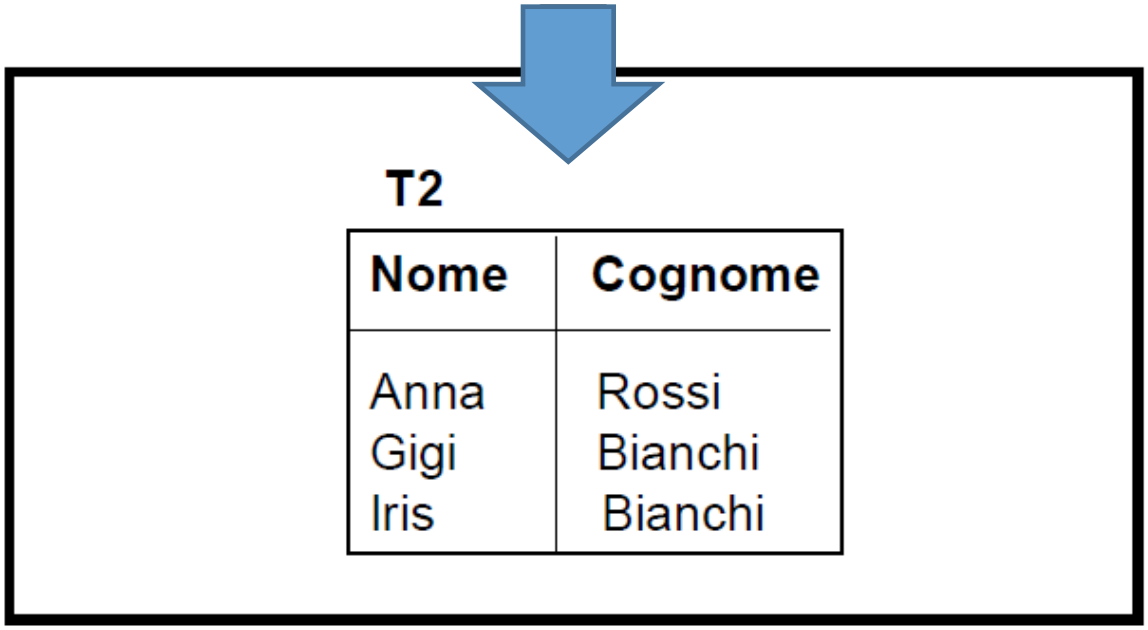
- **PROIEZIONE: Estrazione di attributi (valori contenuti in un sottoinsieme di colonne di una tabella specificate dall'utente)**
- **ESTRAZIONE: Selezione di alcune righe (record) da una tabella nel caso in cui queste corrispondano ad alcuni criteri specificati dall'utente**
- **JOIN: Interrogazione simultanea di più tabelle basata su relazioni. Concettualmente equivale a creare in memoria una macrotabella costituita dai dati contenuti in più tabelle. Solitamente dopo il join viene effettuata un'estrazione.**

Esempio di PROIEZIONE

T1

Nome	Cognome	Nato il	Nato a
Anna	Rossi	2/2/71	TO
Gigi	Bianchi	23/4/80	Ivrea
Iris	Bianchi	15/9/45	CN

La **proiezione** di T1 sugli attributi Nome e Cognome restituisce



T2

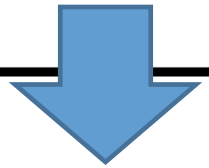
Nome	Cognome
Anna	Rossi
Gigi	Bianchi
Iris	Bianchi

Esempio di ESTRAZIONE (o selezione)

T1

Nome	Cognome	Nato il	Nato a
Anna	Rossi	2/2/71	TO
Gigi	Bianchi	23/4/80	Ivrea
Iris	Bianchi	15/9/45	CN

La **selezione** dei record di T1 tali che **"Nato il >= 1/1/1960"** restituisce



T2

Nome	Cognome	Nato il	Nato a
Anna	Rossi	2/2/71	TO
Gigi	Bianchi	23/4/80	Ivrea

Esempio di JOIN

Chiave primaria

Chiave esterna

T1

Titolo	Autore	Codice
Poesie	Rossi	111
Prosa	Verdi	222
Elegie	Verdi	333

T2

Utente	Cod libro
Pippo	111
Pippo	222
Pluto	111

Il **join** fra le due tabelle restituisce

NB: nessun utente ha acquistato questo libro

T3

Titolo	Autore	Codice	Utente	Cod libro
Poesie	Rossi	111	Pippo	111
Poesie	Rossi	111	Pluto	111
Prosa	Verdi	222	Pippo	222

Il libro non è presente nei risultati

Estrazione di dati mediante SQL: enunciato SELECT

E' lo strumento **principale** per estrarre records. Ha una struttura **composta da 3 parti**:

1. Nella prima parte vengono specificate le operazioni di **proiezione** (nel senso che specifichiamo **quali** campi (colonne) vogliamo estrarre e di **quali tabelle**)

2. Nella seconda parte viene specificata la **tabella** (o la macrotabella definita mediante una o più operazioni di **join**) da cui vogliamo estrarre i dati

3. Nella terza ed ultima parte è possibile specificare i **criteri di estrazione** ossia l'insieme di regole a cui un record **DEVE** essere conforme perchè venga restituito tra i risultati dell'interrogazione (query) SQL.

Struttura :

questa parte è **opzionale**

SELECT 1 FROM 2 WHERE 3 ;

I nomi dei campi sono separati da ,

WARNING: alcuni programmi richiedono che la stringa di interrogazione SQL termini con **;**

Esempio di utilizzo di enunciato SELECT

Il database **Ensembl core** contiene una tabella **gene**:

Field	Type		Null	Key	Default		Extra
gene_id	int(10) unsigned	16B	NO	PRI	(NULL)	0K	auto_increment
biotype	varchar(40)	11B	NO		(NULL)	0K	
analysis_id	smallint(5) unsigned	20B	NO	MUL	(NULL)	0K	
seq_region_id	int(10) unsigned	16B	NO	MUL	(NULL)	0K	
seq_region_start	int(10) unsigned	16B	NO		(NULL)	0K	
seq_region_end	int(10) unsigned	16B	NO		(NULL)	0K	
seq_region_strand	tinyint(2)	10B	NO		(NULL)	0K	
display_xref_id	int(10) unsigned	16B	YES	MUL	(NULL)	0K	
source	varchar(20)	11B	NO		(NULL)	0K	
status	enum('KNOWN','NOVEL','PUTATIVE','PREDICTED','KNOWN_BY_PRO...	76B	YES		(NULL)	0K	
description	text	4B	YES		(NULL)	0K	
is_current	tinyint(1)	10B	NO		1	1B	
canonical_transcript_id	int(10) unsigned	16B	NO		(NULL)	0K	
canonical_annotation	varchar(255)	12B	YES		(NULL)	0K	

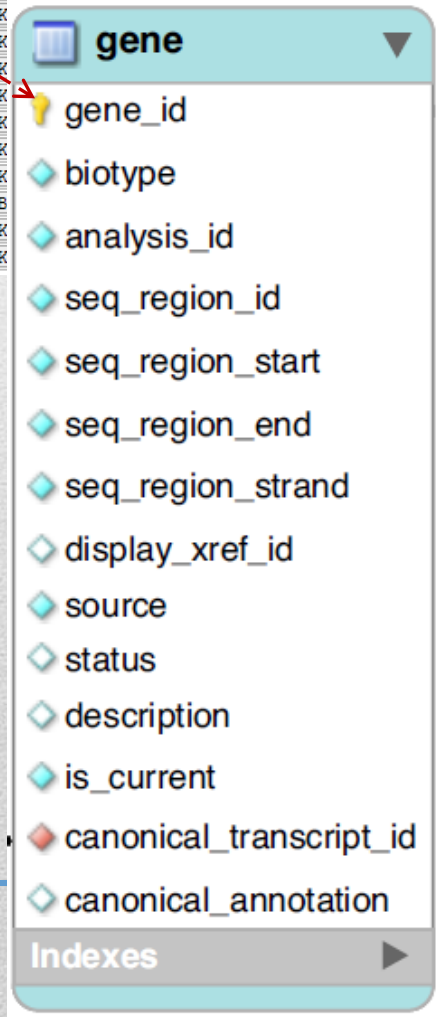
chiave primaria

chiavi esterne

descrizione dettagliata (nomi campi, tipi di dato …)

SQL: DESCRIBE gene;

Rappresentazione **semplificata** (nomi campo + simboli ma non tipo di dato). Comune in molti strumenti ad interfaccia grafica ed **estremamente comune** nei diagrammi che descrivono gli schemi delle banche dati



Esempio di utilizzo di enunciato SELECT

Conoscendo la definizione (struttura) della tabella(e) a cui siamo interessati possiamo scrivere la query SQL per estrarre dati da essa(e):

proiezione

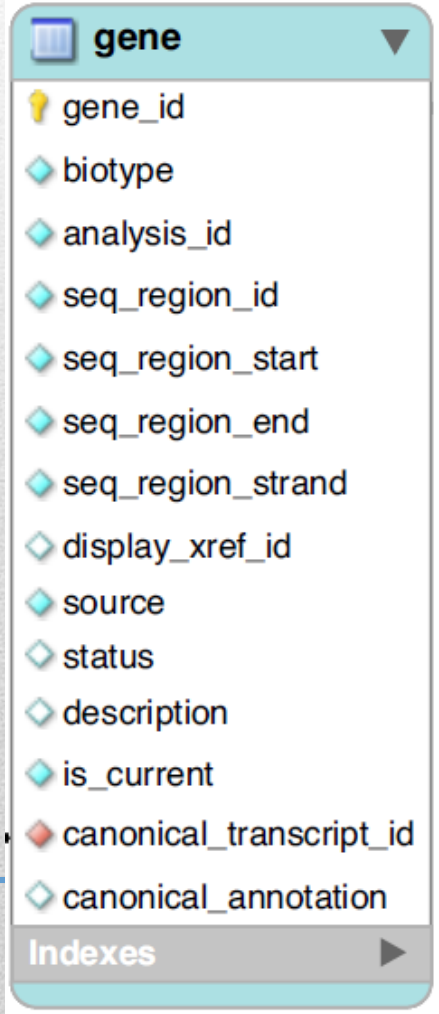
```
SELECT gene_id, biotype, status FROM gene;
```

Nessuna proiezione: estrae tutti i campi

Nessun criterio di selezione … estrae tutti i record disponibili

```
SELECT * FROM gene WHERE biotype = 'protein_coding';
```

Criterio di selezione: estrae **solo** i geni che codificano per proteine



Strumenti free per l'accesso a banche dati relazionali

Proveremo ad effettuare alcuni esperimenti pratici utilizzando uno strumento free: **SQLyog**

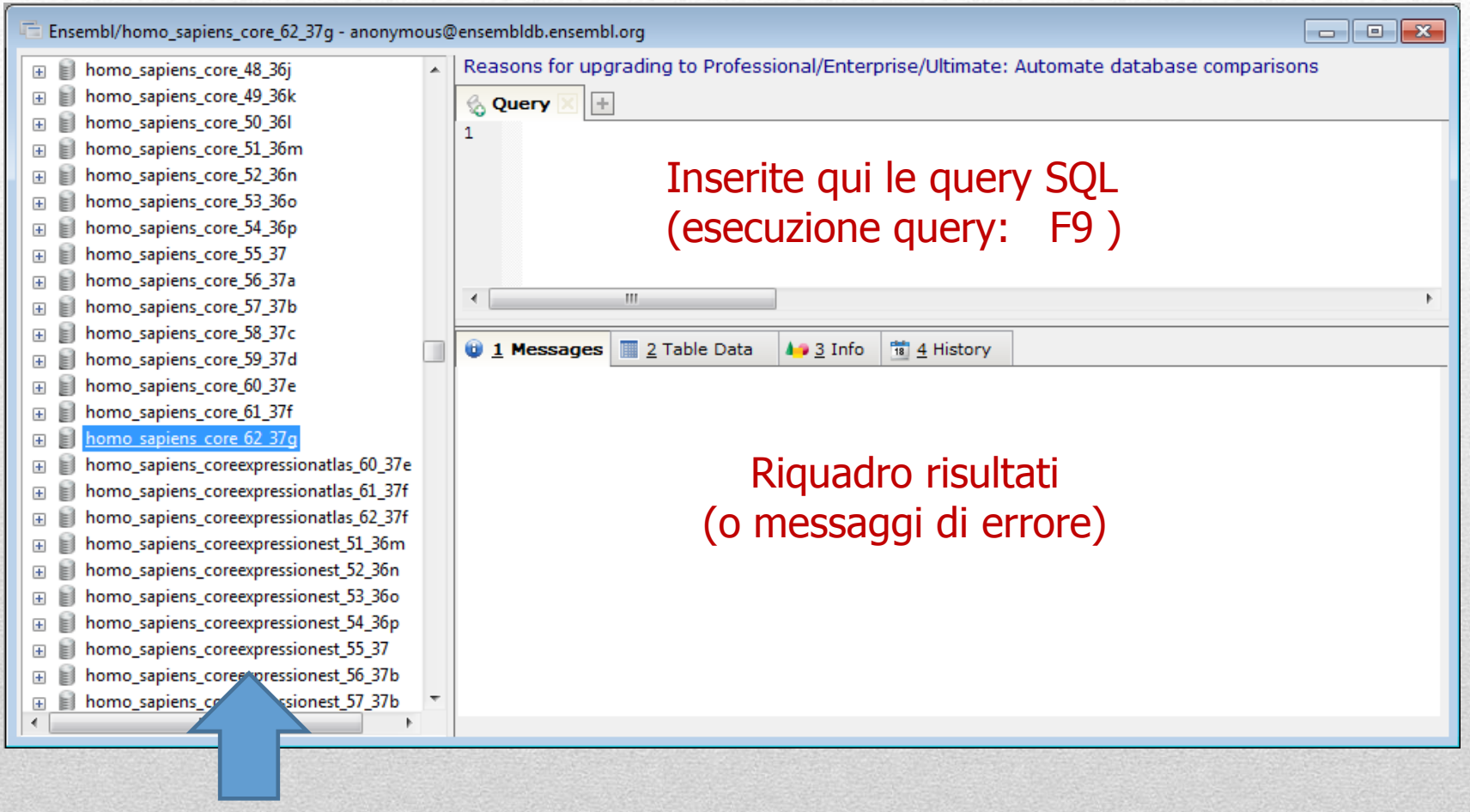
Scaricatelo da questo sito:
http://code.google.com/p/sqlyog/downloads/list
(scaricate l'ultima versione : SQLyog-9.0.1-1Community.exe)

Provate ad installarlo in una directory in cui **avete i permessi di scrittura** (es. Documenti).

Una volta installato definite i parametri per una nuova connessione:

File -> New **connection**
Valori:
Nome connessione EnsEMBL
MySQL Host Address ensembldb.ensembl.org
Username anonymous
Port 5306
(lasciate vuota la password che non serve)

SQLyog: finestra principale

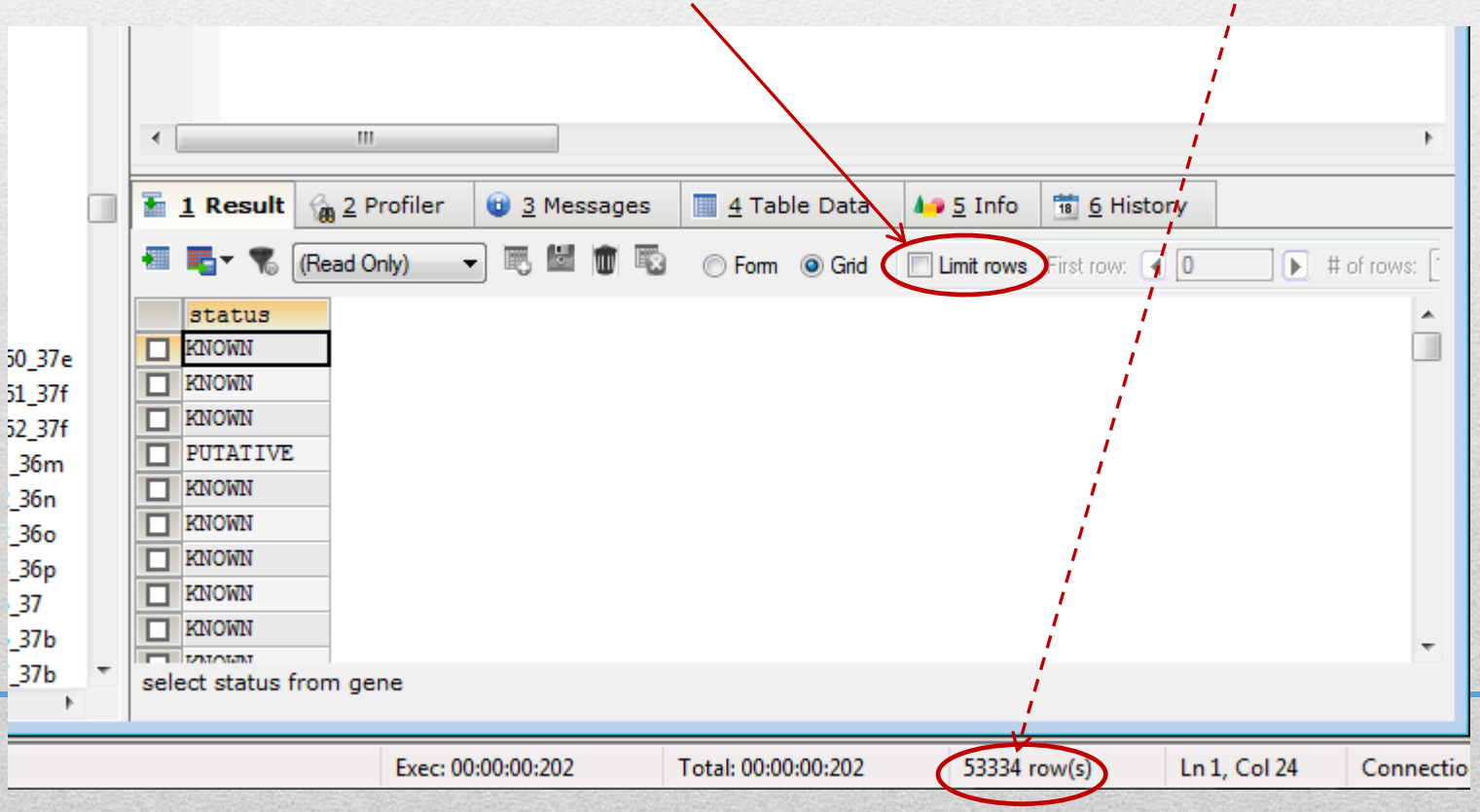


Database disponibili: a noi interessa **homo_sapiens_core_62_37g**
(click sx per selezionarlo)

Interrogazione diretta di Ensembl

Proviamo ad utilizzare le seguenti query SQL:

1. SELECT * FROM gene LIMIT 100;
2. SELECT status FROM gene; (1000 record)
3. Deselezioniamo la casella 'Limit rows' (quanti record otteniamo?)

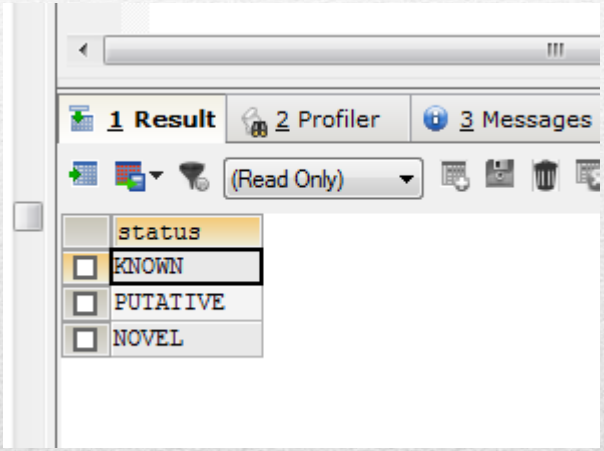


Interrogazione diretta di Ensembl

Quali sono i possibili valori presenti un una data colonna?

SELECT **DISTINCT(**status**)** FROM gene;

Quanti record ottenete?



NON utilizzeremo SQLyog per realizzare i nostri accessi diretti ad Ensembl (utilizzeremo Perl), ma esso è uno strumento molto comodo per testare le query SQL **prima** di inserirle in uno script, in modo da essere sicuri che si comportino secondo le attese.

Esecuzione di query SQL da remoto in Perl

```
# PERL MODULES
use DBI;
use DBD::mysql;

# CONFIG VARIABLES
$platform = "mysql";
$host = "ensembldb.ensembl.org";
$port = "5306";
$user = "anonymous";
$pw = "";
$database="homo_sapiens_core_62_37g";

#DATA SOURCE NAME ##############################
$dsn = "dbi:mysql:$database:$host:5306";

#CONNECTION ####################################
$DBIconnect = DBI->connect($dsn, $user, $pw);

#Query #########################################

$sqlquery = "select * from gene limit 10";

$sth = $DBIconnect->prepare($sqlquery);

$sth->execute;

#PRINT RESULTS #################################
while (@row = $sth->fetchrow_array) {
print "@row\n";
}
```

Librerie SQL

Parametri di connessione

Connessione

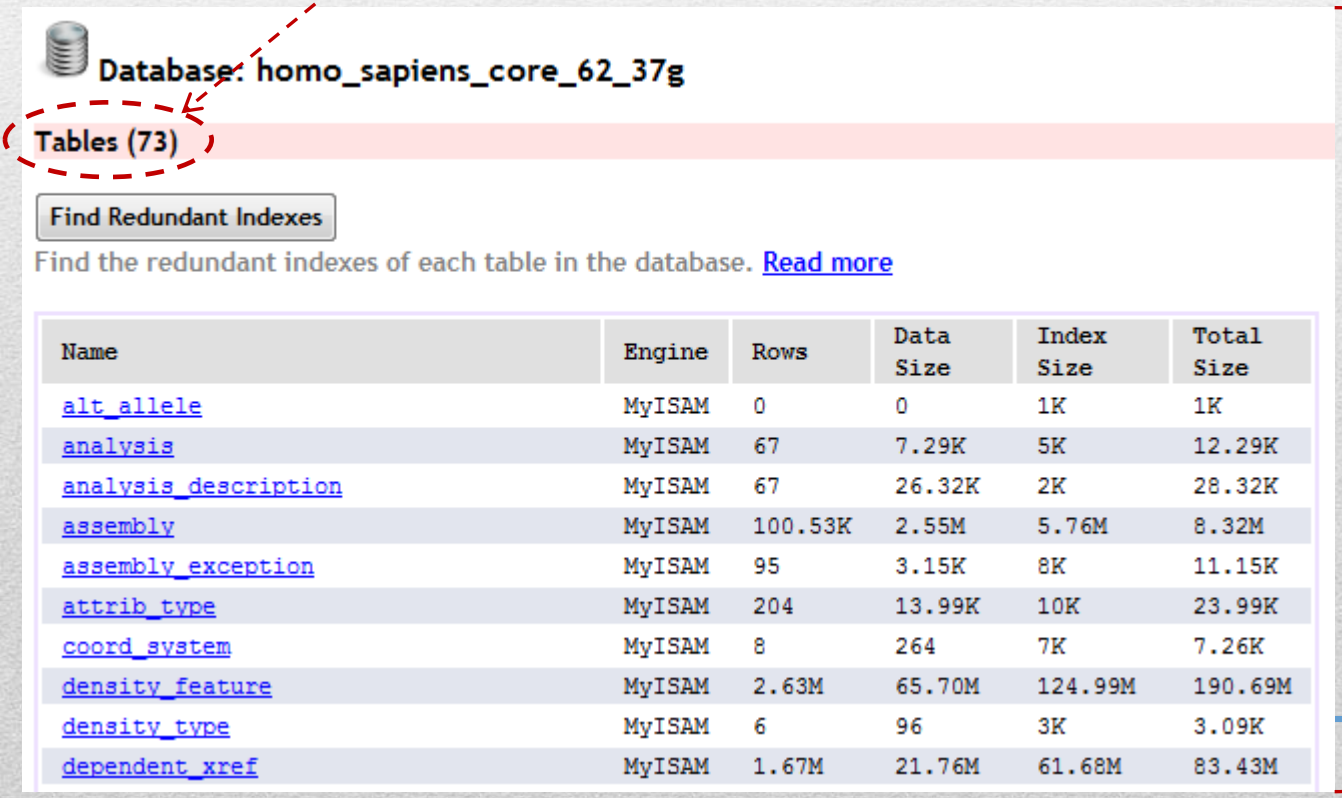
Interrogazione

Stampa risultati

Esecuzione di query SQL da remoto in Perl

Sembra tutto relativamente semplice … quindi dov'è la difficoltà? La difficoltà sta nella costruzione delle stringhe che contengono le query SQL. Noi abbiamo visto un esempio che interroga **1 tabella** del database core di Ensembl dedicato ad homo sapiens.

Ma **quante tabelle contiene** questo database?



Database: homo_sapiens_core_62_37g

Tables (73)

Find Redundant Indexes

Find the redundant indexes of each table in the database. Read more

Name	Engine	Rows	Data Size	Index Size	Total Size
alt_allele	MyISAM	0	0	1K	1K
analysis	MyISAM	67	7.29K	5K	12.29K
analysis_description	MyISAM	67	26.32K	2K	28.32K
assembly	MyISAM	100.53K	2.55M	5.76M	8.32M
assembly_exception	MyISAM	95	3.15K	8K	11.15K
attrib_type	MyISAM	204	13.99K	10K	23.99K
coord_system	MyISAM	8	264	7K	7.26K
density_feature	MyISAM	2.63M	65.70M	124.99M	190.69M
density_type	MyISAM	6	96	3K	3.09K
dependent_xref	MyISAM	1.67M	21.76M	61.68M	83.43M

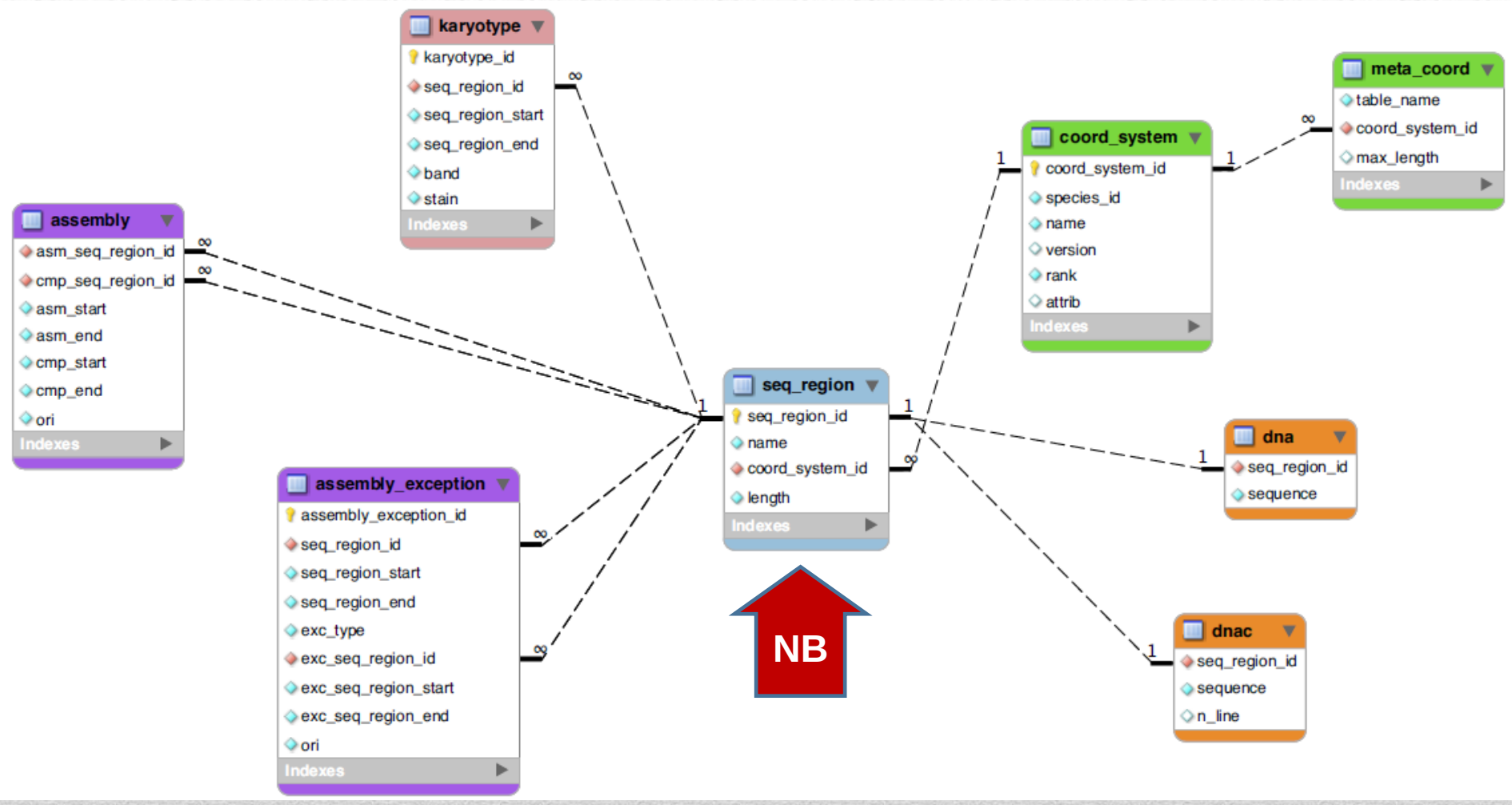
…

Reazioni comuni:

I) Sono troppe …
II) Potevano costruire un database meno complesso
III) AIUTO!!! MI SERVE UNA MAPPA

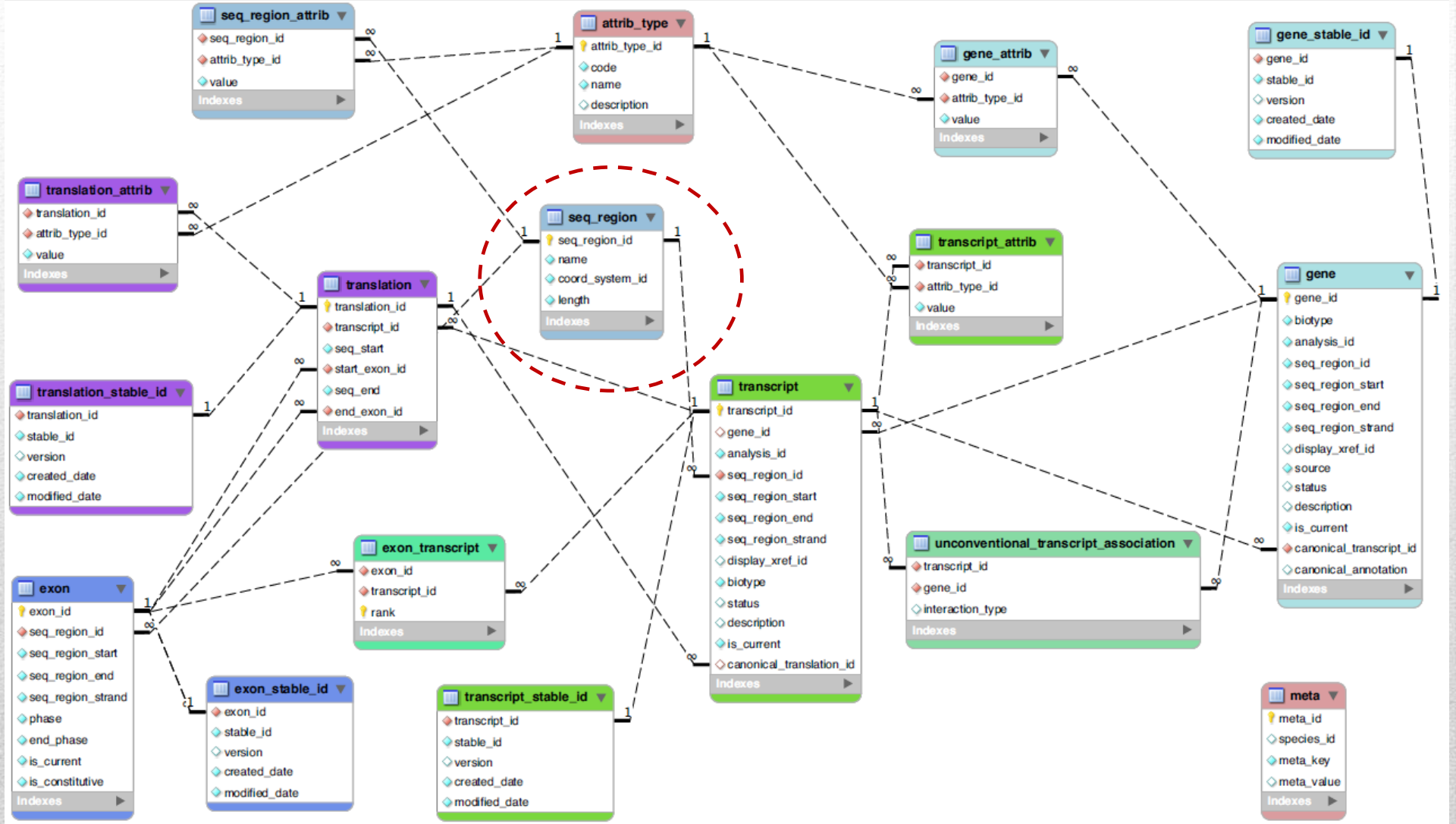
Database schema

Ensembl core db schema (I)



NB: il processo di annotazione genomica **non viene effettuato <u>unicamente</u> sulle sequenze genomiche assemblate**. Parte di esso viene effettuato su cloni, contigui, supercontigui ecc.. OGNI ANNOTAZIONE esiste in uno specifico sistema di coordinate

Ensembl core db schema (II)



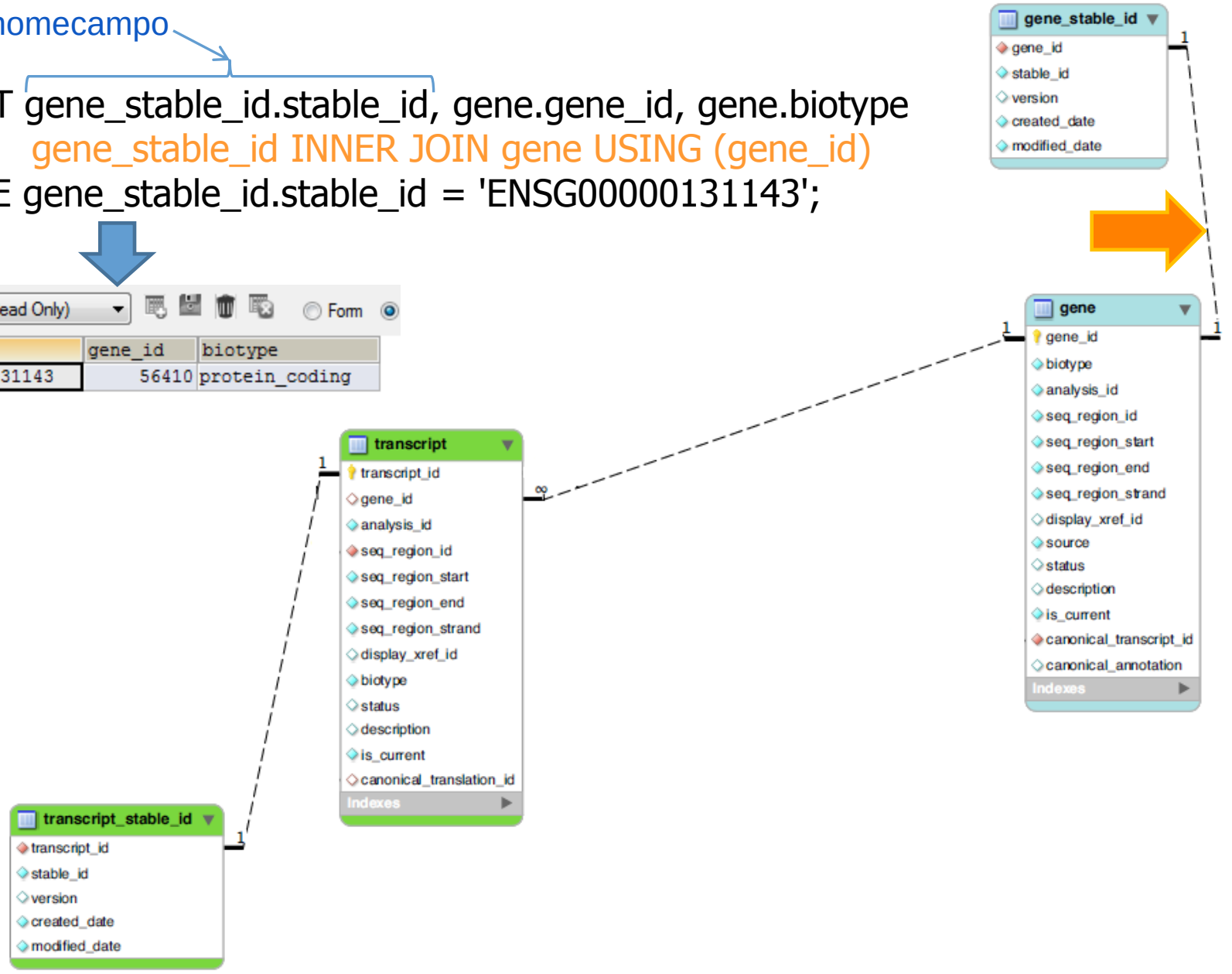
Ensembl core db schema (II) : JOIN

nometabella.nomecampo

```
SELECT gene_stable_id.stable_id, gene.gene_id, gene.biotype
FROM   gene_stable_id INNER JOIN gene USING (gene_id)
WHERE gene_stable_id.stable_id = 'ENSG00000131143';
```

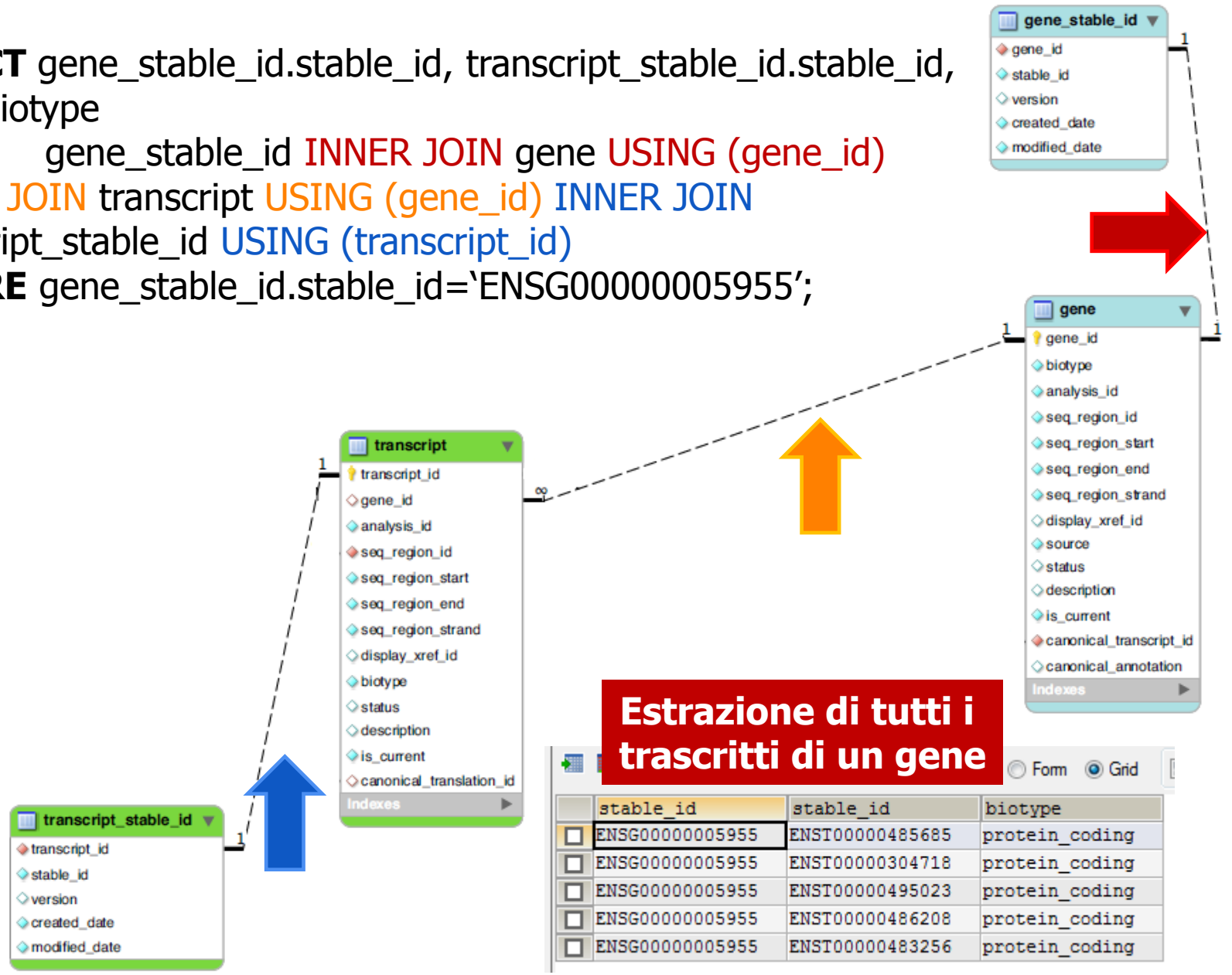
(Read Only) Form

stable_id	gene_id	biotype
ENSG00000131143	56410	protein_coding



Ensembl core db schema (II) : JOIN

SELECT gene_stable_id.stable_id, transcript_stable_id.stable_id, gene.biotype
FROM gene_stable_id INNER JOIN gene USING (gene_id) INNER JOIN transcript USING (gene_id) INNER JOIN transcript_stable_id USING (transcript_id)
WHERE gene_stable_id.stable_id='ENSG00000005955';

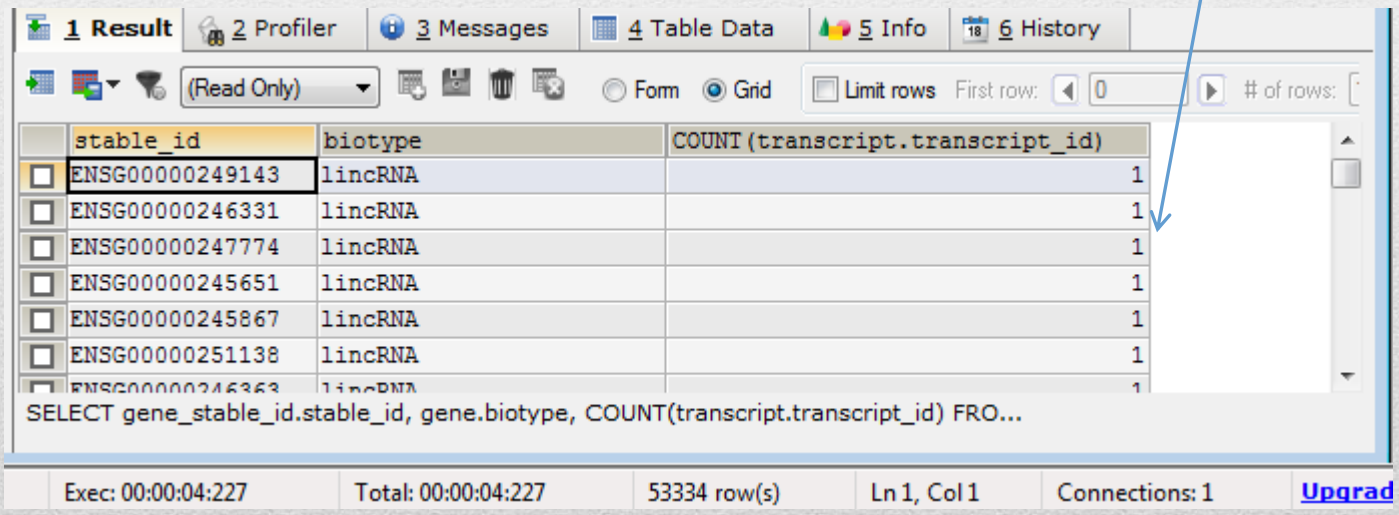


Ensembl : AGGREGAZIONE

```
SELECT gene_stable_id.stable_id, gene.biotype, COUNT(transcript.transcript_id)
FROM    gene_stable_id INNER JOIN gene USING (gene_id)
INNER JOIN transcript USING (gene_id) GROUP BY gene_stable_id.stable_id
ORDER BY COUNT(transcript.transcript_id);
```

CONTEGGIO di tutti i trascritti di OGNI gene

ordinamento crescente

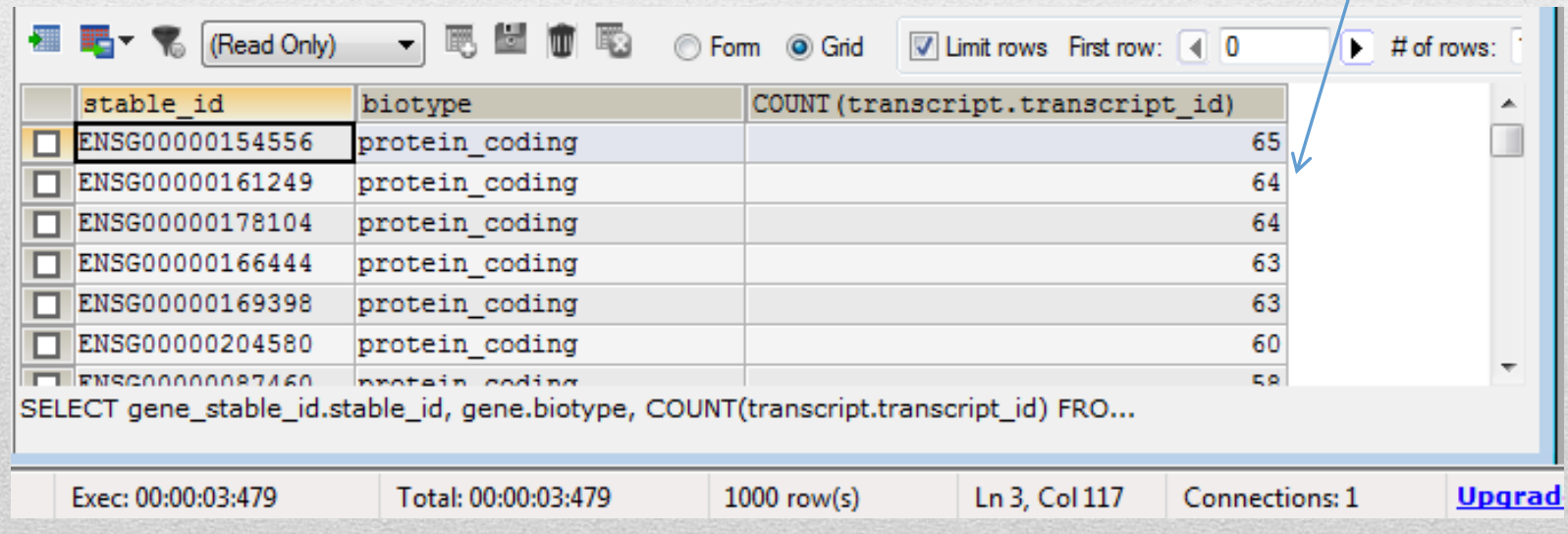


Ensembl : AGGREGAZIONE

```
SELECT gene_stable_id.stable_id, gene.biotype, COUNT(transcript.transcript_id)
FROM    gene_stable_id INNER JOIN gene USING (gene_id)
INNER JOIN transcript USING (gene_id) GROUP BY gene_stable_id.stable_id
ORDER BY COUNT(transcript.transcript_id) DESC;
```

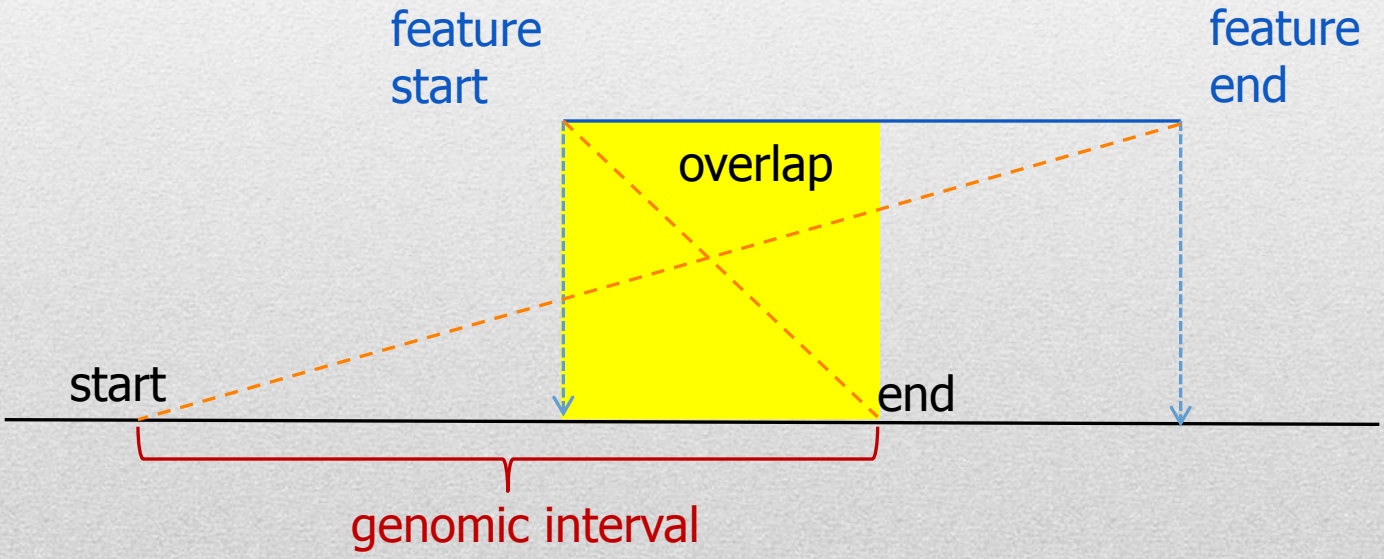
CONTEGGIO di tutti i trascritti di OGNI gene

ordinamento **DE**crescente

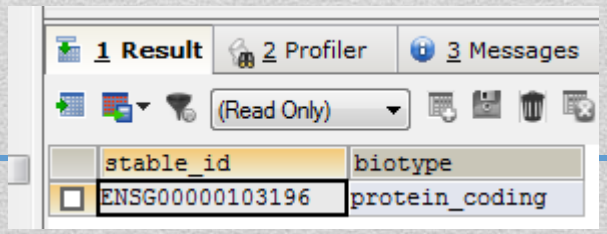


Ensembl core db schema (II) : estrazione su base *posizionale*

```
SELECT gene_stable_id.stable_id, gene.biotype
FROM    seq_region INNER JOIN gene USING (seq_region_id) INNER JOIN
gene_stable_id USING (gene_id)
WHERE NOT(gene.seq_region_start>84966302 OR
gene.seq_region_end<84826528) AND seq_region.name = '16';
```

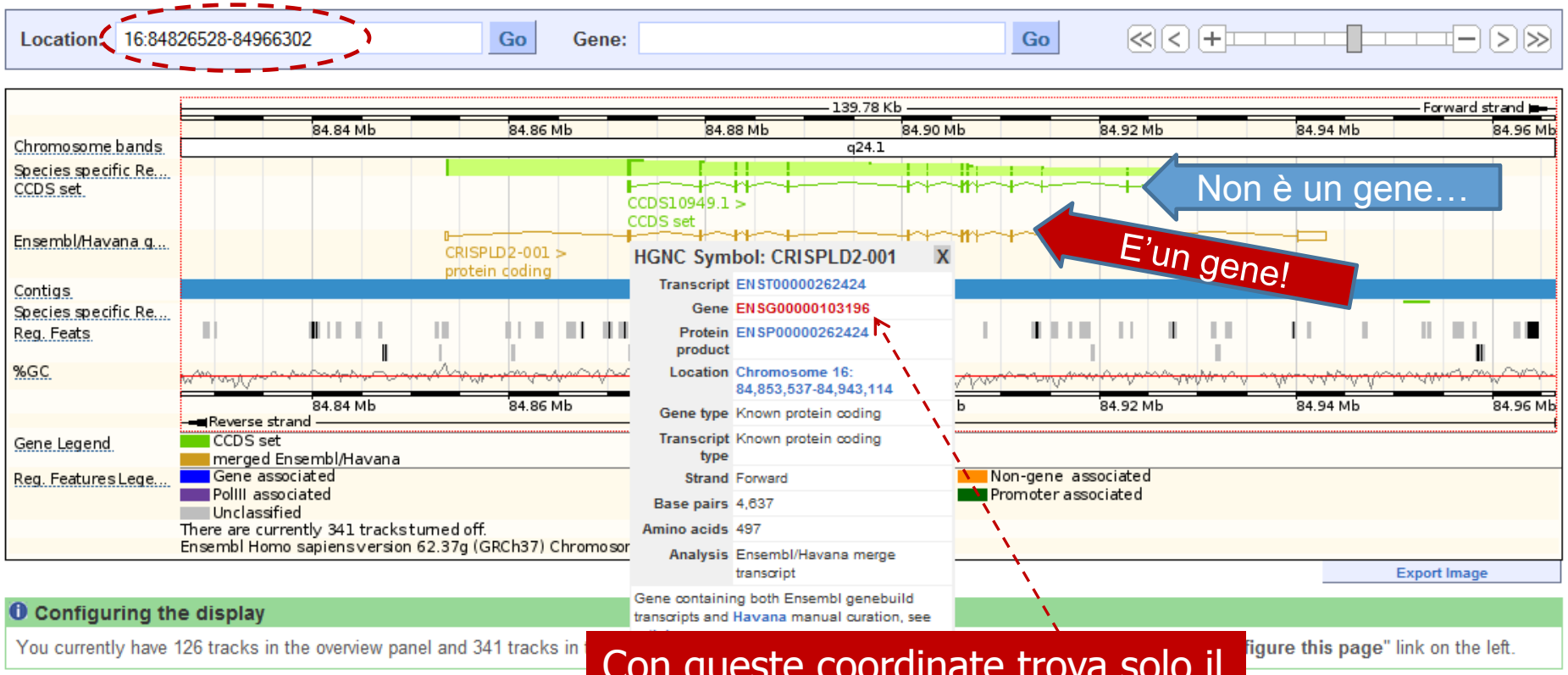


Con queste coordinate trova solo il gene **ENSG00000103196** … sarà vero?



Ensembl core db schema (II) : estrazione su base *posizionale*

SELECT gene_stable_id.stable_id, gene.biotype
FROM seq_region INNER JOIN gene USING (seq_region_id) INNER JOIN gene_stable_id USING (gene_id)
WHERE NOT(gene.seq_region_start>**84966302** OR gene.seq_region_end<**84826528)** AND seq_region.name = '**16**';



Con queste coordinate trova solo il gene **ENSG00000103196** … (conferma dal browser genomico)

Ensembl SQL :
estrazione basata su pattern di testo

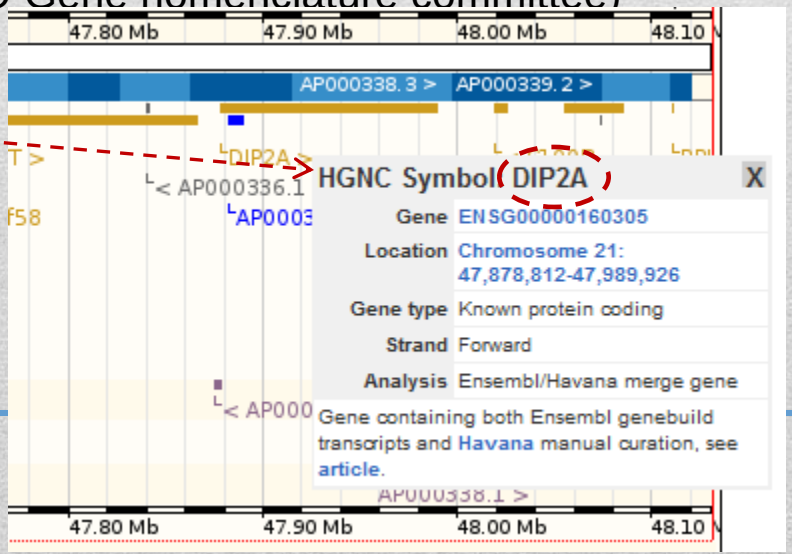
SELECT gene_stable_id.stable_id, gene.biotype, xref.display_label **FROM** gene_stable_id INNER JOIN gene USING (gene_id) INNER JOIN xref ON (gene.display_xref_id = xref.xref_id) **WHERE** xref.display_label **LIKE 'DIP2%'**;

1 Result | 2 Profiler | 3 Messages | 4 Table Data

(Read Only) Form Grid

stable_id	biotype	display_label
ENSG00000160305	protein_coding	DIP2A
ENSG00000066084	protein_coding	DIP2B
ENSG00000151240	protein_coding	DIP2C

NBB: Tabella **xref** (**external reference**) fa riferimento ad un dato mappato sul genoma ma **ORIGINARIO** di un'altra banca dati. Nel caso (ad es.) di **DIP2A** si tratta di un gene symbol definito da **HGNC** (HUGO Gene nomenclature committee)



Esercizi (SQL)

- Scrivete una query SQL che restituisca tutti i trascritti di un gene a vostra scelta (**3 pt**)
- Scrivete una query che restituisca **IL NUMERO** degli pseudogeni umani annotati in Ensembl (**3 pt**)
- Scrivete una query che restituisca tutti i geni del cromosoma 1 di tipo **diverso** da protein_coding (**3 pt**)
- Scrivete **uno script** a cui passare come parametro il nome di un gene e che restituisca il numero dei suoi trascritti e, per ciascun trascritto, I nomi e le posizioni dei suoi esoni . Potete realizzare l'esercizio mediante **più query** successive (gene → trascritti, per ogni trascritto → esoni) (6 **pt**).
- Scrivete **uno script** a cui passare come parametro delle coordinate genomiche e che restituisca le simple_feature annotate nella regione genomica, la loro posizione ed il loro tipo . NB: questo esercizio **richiede l'utilizzo di INNER JOIN da una tabella che dovete identificare ad altre due tabelle: seq_region e analysis** (**6 pt**).

Docente: **Giorgio Valentini**
Istruttore: **Matteo Re**

UNIVERSITÀ DEGLI STUDI DI MILANO

C.d.l. Biotecnologie Industriali e Ambientali

Biologia computazionale

A.A. 2010-2011 semestre II

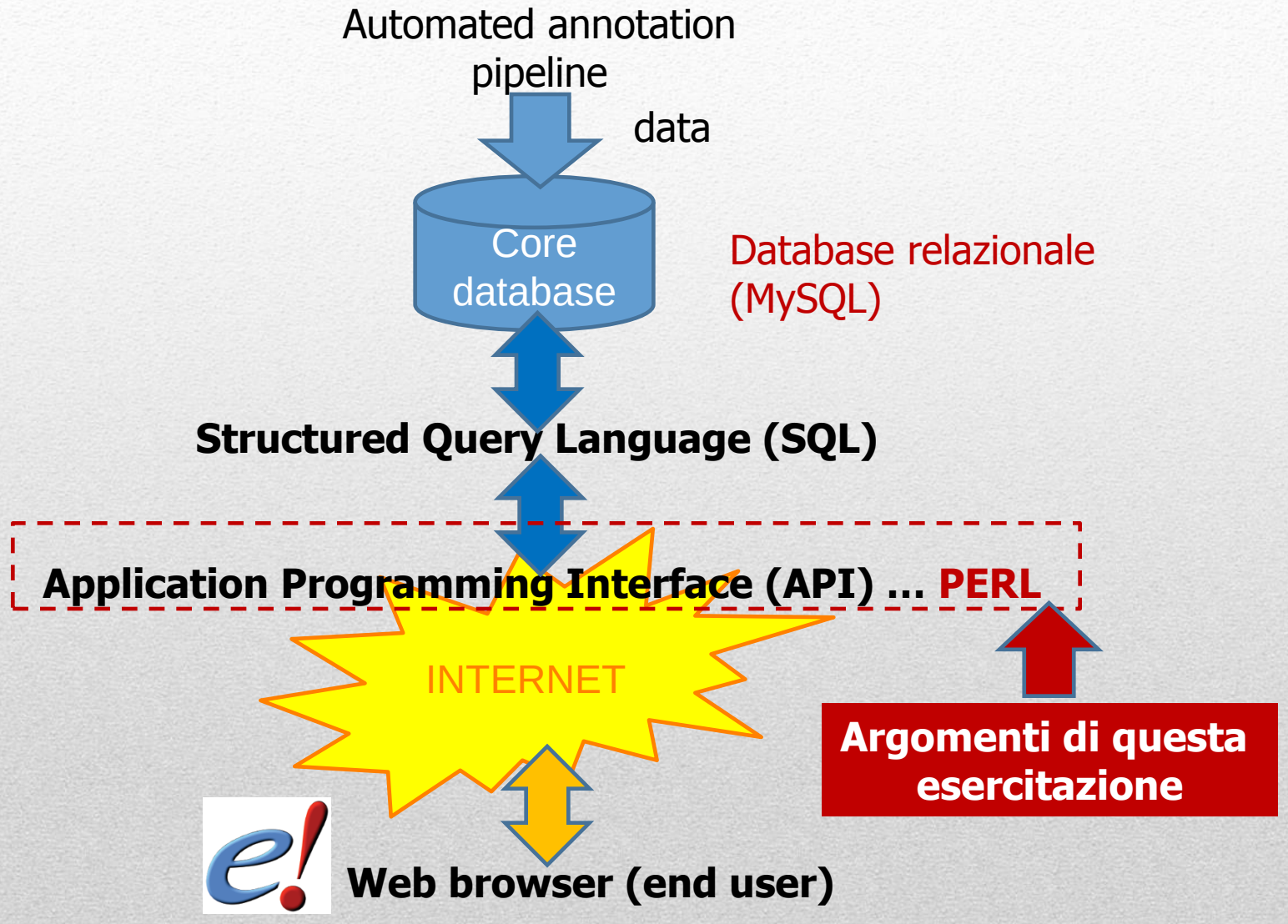
p7 Interrogazione diretta di banche dati biologiche - API

Ensembl API :

Application Programming Interface

*"An **Application Programming Interface (API)** is a set of definitions of the ways in which one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software. One of the primary purposes of an API is to provide a set of commonly-used functions (…). Programmers can then take advantage of the API by making use of its functionality, saving them the task of programming everything from scratch."*

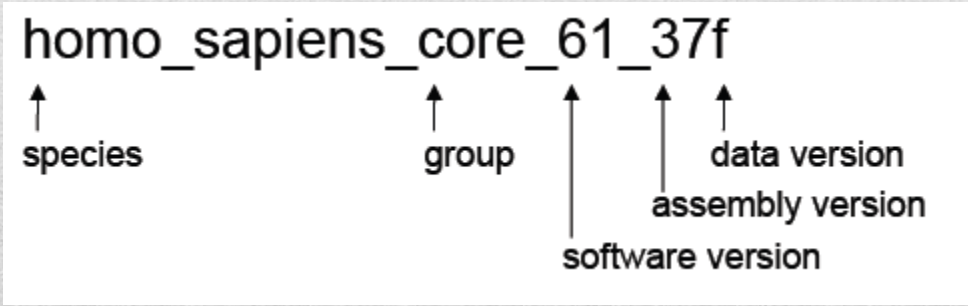
Architettura del browser genomico Ensembl



Ensembl core database(s) e API

- Ensembl core database contiene:
  - Sequenze genomiche
  - Informazioni riguardanti l'assemblaggio delle sequenze genomiche
  - Modelli di geni, trascritti e proteine
  - Allineamenti di cDNA e proteine
  - Bande citogenetiche, marcatori, repeats, isole CpG ecc.
  - Riferimenti a banche dati esterne

Il nome del database:



Ensembl core Perl API

- Utilizzata per:
  - Estrarre informazioni dal database core
  - Salvare informazioni nel database core
  - E' parte della pipeline di annotazione automatica Ensembl
- E' scritta in Object-oriented Perl
  - Basata in parte su (e compatibile con) obbetti BioPERL (versione 1.2.3)
- Costituisce la **base** delle **ALTRE** API Ensembl

NB:

- La **versione** dell'API utilizzata **DEVE** corrispondere a quella del **database** da cui si vogliono estrarre i dati (database e API vengono rilasciati in coppia)

Informazioni installazione API Ensembl (PERL) :
www.ensembl.org/info/docs/api/api_installation.html

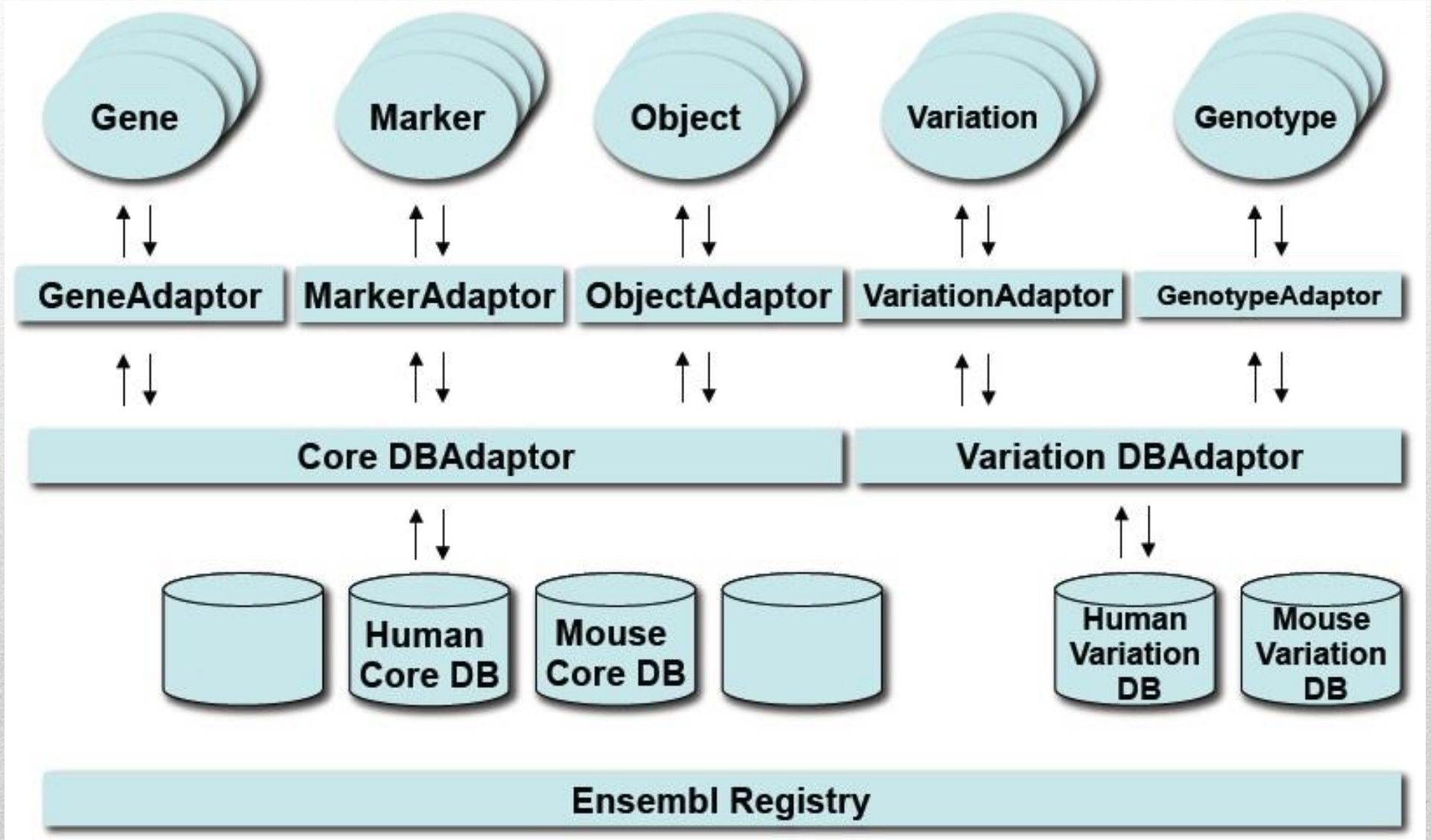
Ensembl core Perl API

- Oggetti associati ai dati (data objects):
  - Modellano **entità biologiche** come geni, marcatori, SNP, elementi regolatori
  - Ogni data object **incapsula** informazioni derivanti da una sola tabella (o da un numero molto ristretto di tabelle)
  - Name space: Ogni nome di data object inizia con Bio::Ensembl (ad esempio Bio::Ensembl::Gene per gli oggetti di tipo gene)
  - I data objects sono estratti dal database o scritti nel database mediante degli **Object Adaptors** (adattatori)
  - Gli Object Adaptors sono **creatori di Data Objects**
  - Ogni Object Adaptor può produrre **1 solo tipo di oggetto**
  - **Un Object Adaptor** può essere creato solo da un Database Adaptor
  - **Un Database Adaptor** può connettersi solo ad un **singolo database**

Ensembl core Perl API

- Un adattatore speciale: Registry
  - E’ un adattatore in grado di creare **tutti gli altri adattatori**
  - Fornisce la connessione alla banca dati
  - Permette all’utente di specificare file di configurazione

Architettura del sistema



Installazione Ensembl API

- Seguire le istruzioni riportate in questa pagina web:

http://www.ensembl.org/info/docs/api/api_installation.html

NB: in aula di calcolo riceverete l'API Ensembl (l'ultima versione disponibile) mediante una **chiave USB**.

Per usare l'API da Perl è necessario aggiungere il percorso della cartella che contiene l'API Ensembl alla lista dei path in cui l'interprete cerca le librerie. Invece di cambiare le impostazioni di sistema aggiungeremo l'indirizzo della **directory direttamente negli script Perl**:

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/', 'C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
```

Ensembl core API:
1 : ottenere tutti i sistemi di coordinate

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;

my $registry = 'Bio::EnsEMBL::Registry';

$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $coordsystem_adaptor = $registry->get_adaptor( 'Human', 'Core','CoordSystem' );

my $coordsystems = $coordsystem_adaptor->fetch_all;

while ( my $coordsystem = shift @{$coordsystems} ){
print $coordsystem->name, "\t", $coordsystem->version, "\n";
}
```

```
chromosome      GRCh37
supercontig     GRCh37
clone
contig
chromosome      NCBI36
chromosome      NCBI35
chromosome      NCBI34
lrg
```

Ensembl core API: sequenze

- Le sequenze immagazzinate nel sistema Ensembl sono associate ad oggetti di tipo **Sequence Regions**
- Gli oggetti Sequence Regions vengono mappati posizionalmente in molti sistemi di coordinate organizzati **gerarchicamente**
- Il livello più alto del sistema di coordinate è il livello ‘chromosome’ ma ne esistono molti altri : contiguo, supercontiguo …

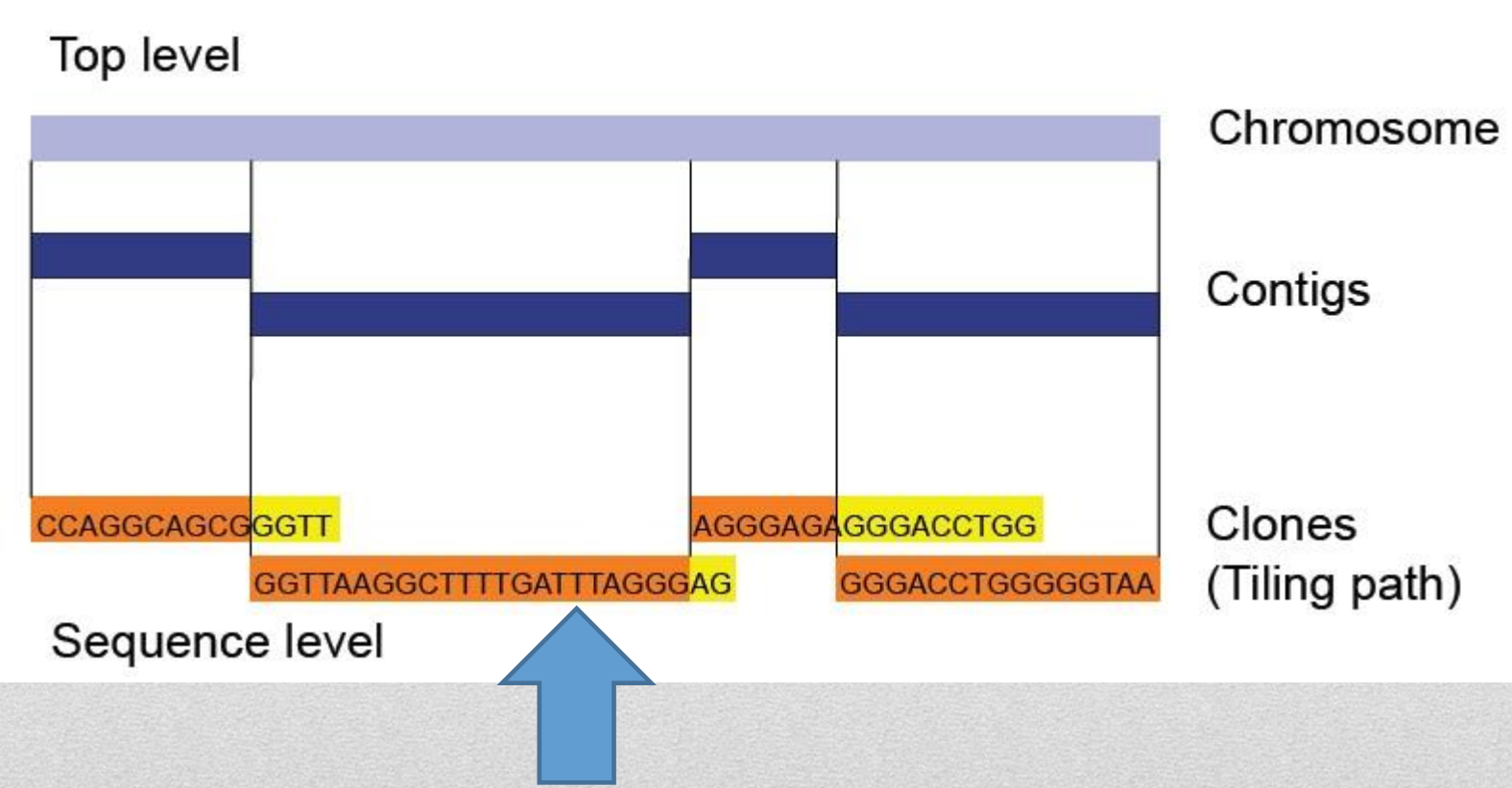
Ensembl core API:
sistemi di coordinate

- I sistemi di corodinate **variano da specie a specie** poichè si devono adattare ai dati prodotti durante specifici progetti di sequenziamento genomico

  - human: chromosome, supercontig, clone, contig
  - zebrafish: chromosome, scaffold, contig

- Mediante l'API core è possibile estrarre la sequenza **in un sistema di coordinate e mappare la sequenza negli altri sistemi di coordinate**. Metodo molto flessibile per muoversi tra cromosomi, cloni, contigui ecc.

Ensembl core API: sistemi di coordinate



La sequenza è immagazzinata nel database **al livello più basso** … ma mediante API possiamo estrarla utilizzando coordinate di alto livello (es. chromosome level) L'assemblaggio viene effettuato in automatico dall'API.

Ensembl core API: oggetti Slice

- Un oggetto di tipo Slice rappresenta una regione **arbitraria** del genoma
- Gli oggetti Slice non sono fisicamente presenti (come dato) nel database (vengono creati a runtime)
- Gli oggetti Slice vengono utilizzati per **ottenere SEQUENZE o FEATURE** associate ad una specifica regione inun dato sistema di coordinate.

FEATURE: entità **mappata** sul genoma
(allineamenti, trascritti, isole CpG, marcatori,…)

Ensembl core API:
2 : creazione di oggetti Slice (intero cromosoma)

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;

my $registry = 'Bio::EnsEMBL::Registry';

$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', 'Y' );
print
"Coord system:\t", $slice->coord_system_name,
"\nSeq region:\t", $slice->seq_region_name,
"\nStart:\t\t", $slice->start,
"\nEnd:\t\t", $slice->end,
"\nStrand:\t\t", $slice->strand,
"\nSlice:\t\t", $slice->name, "\n";
```

Connessione

API Slice

Ensembl core API:
2 : creazione di oggetti Slice (intero cromosoma)

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;

my $registry = 'Bio::EnsEMBL::Regi

$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $slice_adaptor = $registry->get
my $slice = $slice_adaptor->fetch_
print
"Coord system:\t", $slice->coord_s
"\nSeq region:\t", $slice->seq_reg
"\nStart:\t\t", $slice->start,
"\nEnd:\t\t", $slice->end,
"\nStrand:\t\t", $slice->strand,
"\nSlice:\t\t", $slice->name, "\n"
```

OUTPUT:

```
Coord system:
chromosome
Seq region:
Y
Start:
1
End:
59373566
Strand:
1
Slice:
chromosome:GRCh37:Y:1:59373566:1
```

Ensembl core API: oggetti Feature

- Un oggetto di tipo **Feature** rappresenta oggetti per I quali è disponibile una serie di **coordinate** genomiche
- Per ogni oggetto Feature sono disponibili **start, end, strand ed un oggetto slice** che lo contiene.
- La coordinata start **è sempre minore** della coordinata end (indipendentemente dalla strand). Eccezione: feature di tipo **Insertion**
- Gli oggetti Feature sono salvati nel database **in un unico sistema di coordinate** (poi, grazie all' API, si possono ottenere le coordinate negli altri sistemi di coordinate)

Ensembl core API: oggetti Feature

Alcuni esempi di Feature:

- Gene, Transcript, Exon
- ProteinFeature
- PredictionTranscript, PredictionExon
- DNAAlignFeature, ProteinAlignFeature
- RepeatFeature
- MarkerFeature
- OligoFeature
- KaryotypeBandFeature
- SimpleFeature (CpG, tRNAscan, FirstEF, Eponine)
- MiscFeature (Encode regions, clonesets)

Ensembl core API:
3 : Estrazione di features su base posizionale

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;

my $registry = 'Bio::EnsEMBL::Regis

$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $slice_adaptor = $registry->get_
my $slice = $slice_adaptor->fetch_b
10000000 );
my $markers = $slice->get_all_Marke
foreach my $marker( @{$markers} ){
print
$marker->slice->name, "\t", $marker
"\n";
}
```

OUTPUT:

```
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:12722:12949:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:14421:15222:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:49786:49915:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:559430:559611:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:564739:565363:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:565031:566006:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:566868:567469:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:567676:568223:1
chromosome:GRCh37:1:1:10000000:1
chromosome:GRCh37:1:567909:568488:1
Etc.etc.
```

Ensembl core API:
Geni, transcritti ed esoni

- Geni, trascritti ed esoni sono oggetti **Feature**

- Un **Gene** è un **raggruppamento di trascritti** che condividono alcuni esoni (parzialmente) sovrapposti

- Un Trascritto è un **set di esoni**
- Gli introni **non sono definiti esplicitamente nel database**

Ensembl core API:
Proteine (Translations)

- Le sequenze proteiche (Translations) non esistono nel database … **non esistono feature di tipo Translation**.
- Le entità Translation **definiscono le regioni UTR e CDS dei trascritti**
- La **sequenza** delle entità Translation viene **CALCOLATA MEDIANTE API DAGLI OGGETTI DI TIPO Transcript!**

Ensembl core API:
External References

External references (Xrefs) sono dei riferimenti incrociati tra gli oggetti del sistema Ensembl ed **identificativi** di **annotazioni** presenti in **database esterni** (ad es. HGNC, WikiGenes, UniProtKB/Swiss-Prot, RefSeq, MIM, …).

- Le Xrefs vengono annotate a livello di **gene, trascritto o proteina**

Ensembl core API:
4 : Estrazione delle Xrefs associate ad un gene

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;

my $registry = 'Bio::EnsEMBL::Registry';

$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);
my $gene = $gene_adaptor->fetch_by_stable_id( 'ENSG00000139618' );
my $gene_xrefs = $gene->get_all_DBEntries;
print "Xrefs on the gene level: \n\n";
foreach my $gene_xref( @{$gene_xrefs} ){
print $gene_xref->dbname, ":", $gene_xref->display_id, "\n";
}
my $all_xrefs = $gene->get_all_DBLinks;
print "\nXrefs on the gene, transcript and protein level: \n\n";
foreach my $all_xref( @{$all_xrefs} ){
print $all_xref->dbname, ":", $all_xref->display_id, "\n";
}
```

API: Xrefs

Ensembl core API:
4 : Estrazione delle Xrefs associate ad un gene

Output:

Xrefs on the gene level:

OTTG:OTTHUMG00000017411
HGNC:BRCA2
DBASS3:BRCA2
HGNC_curated_gene:BRCA2

Xrefs on the gene, transcript and protein level:

OTTG:OTTHUMG00000017411
HGNC:BRCA2
DBASS3:BRCA2
HGNC_curated_gene:BRCA2
shares_CDS_and_UTR_with_OTTT:OTTHUMT00000046000
UCSC:uc001uub.1
CCDS:CCDS9344.1
RefSeq_dna:NM_000059.3
HGNC:BRCA2
UniGene:Hs.34012
HGNC_curated_transcript:BRCA2-001
EntrezGene:BRCA2
WikiGene:BRCA2
MIM_MORBID:#114480
MIM_MORBID:#137800
MIM_MORBID:#155255
MIM_MORBID:#155720
MIM_MORBID:#176807
MIM_MORBID:#194070
MIM_MORBID:#227650
MIM_MORBID:#260350
MIM_GENE:*600185
MIM_MORBID:#605724
MIM_MORBID:#612555
MIM_MORBID:#613029
MIM_MORBID:#613347
RefSeq_peptide:NP_000050.2
Uniprot/SPTREMBL:A1YBP1_HUMAN
EMBL:DQ897648
protein_id:ABI74674.1
EMBL:AL445212
Uniprot/SPTREMBL:B2ZAH0_HUMAN
EMBL:EU625579
protein_id:ACD01217.1
EMBL:AL137247
Uniprot/SPTREMBL:Q8IU64_HUMAN
EMBL:AY151039
protein_id:AAN28944.1
EMBL:AF489725
protein_id:AAN61409.1
EMBL:AF489726
protein_id:AAN61410.1
EMBL:AF489727
protein_id:AAN61411.1
EMBL:AF489728
protein_id:AAN61412.1
EMBL:AF489729
protein_id:AAN61413.1
EMBL:AF489730
protein_id:AAN61414.1
EMBL:AF489731
protein_id:AAN61415.1
EMBL:AF489732
protein_id:AAN61416.1
EMBL:AF489733
protein_id:AAN61417.1
EMBL:AF489734
protein_id:AAN61418.1
EMBL:AF489735
protein_id:AAN61419.1
EMBL:AF489736
protein_id:AAN61420.1
EMBL:AF489737
protein_id:AAN61421.1
EMBL:AF489738
protein_id:AAN61422.1
Uniprot/SPTREMBL:Q8IU77_HUMAN
EMBL:AF507079
protein_id:AAN61426.1
EMBL:AF507080
protein_id:AAN61427.1
EMBL:AF507081
protein_id:AAN61428.1
EMBL:AF507082
protein_id:AAN61429.1

Etc.etc.

Ensembl core API:
Mapping di feature tra sistemi di coordinate

- Le API forniscono gli strumenti per la conversione automatica delle coordinate delle features tra tutti i sistemi di coordinate disponibili.

- Esistono **diversi metodi** per realizzare un mapping. Essi sono disponibili per particolari tipi di oggetto:

  - **Oggetti Feature**: metodi **transfer**, **transform** e **project**
  - **Oggetti Slice**: metodo **project**

Ensembl core API:
Mapping di feature tra sistemi di coordinate

- Le API forniscono gli strumenti per la conversione automatica delle coordinate delle features tra tutti i sistemi di coordinate disponibili.

- Esistono **diversi metodi** per realizzare un mapping. Essi sono disponibili per particolari tipi di oggetto:

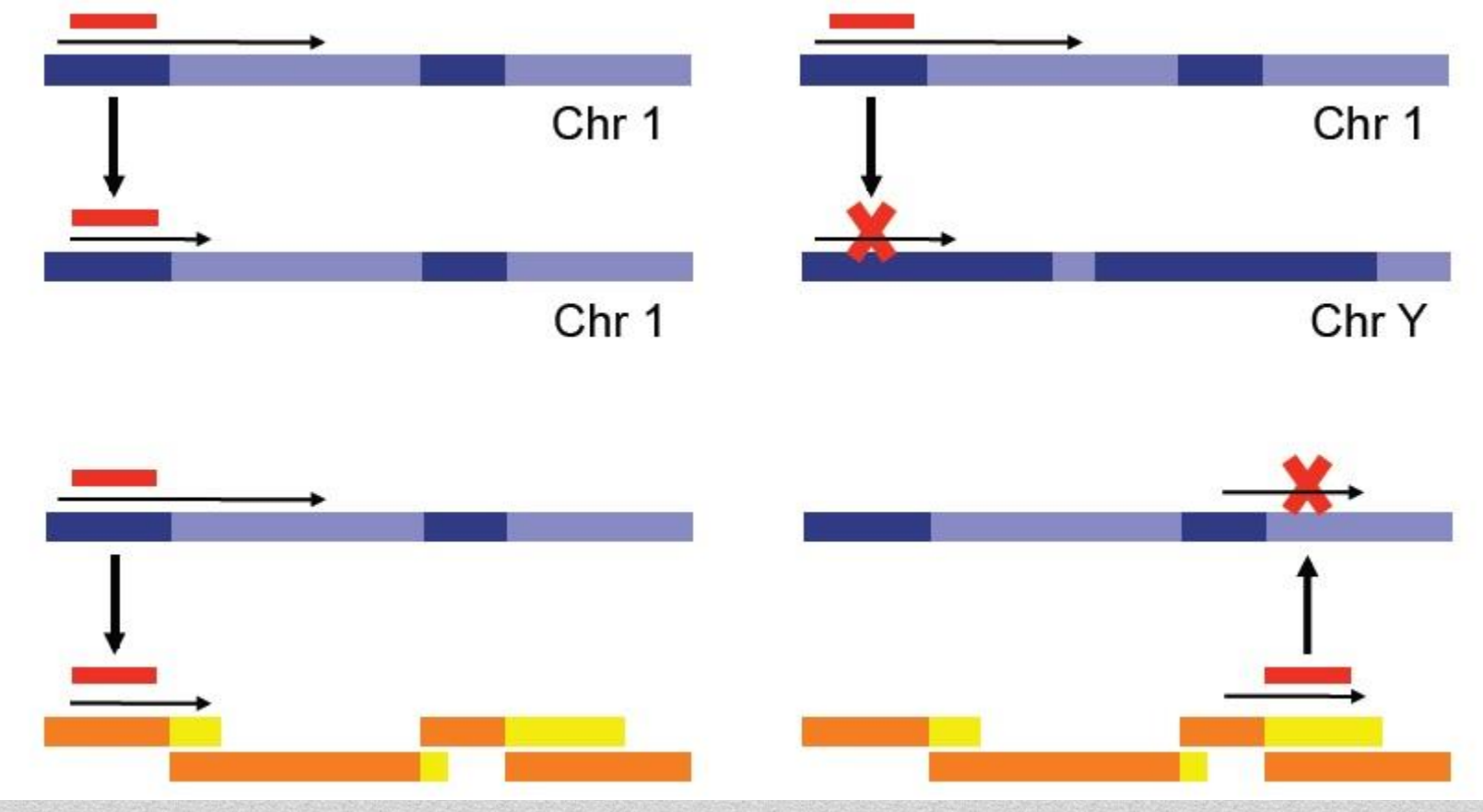
  - **Oggetti Feature**: metodi **transfer**, **transform** e **project**
  - **Oggetti Slice**: metodo **project**

Ensembl core API:
Transfer

- Transfer “sposta” una **feature** da una **slice** in un dato sistema di coordinate ad un’altra slice nello stesso sistema di coordinate **o in un sistema di coordinate differente**

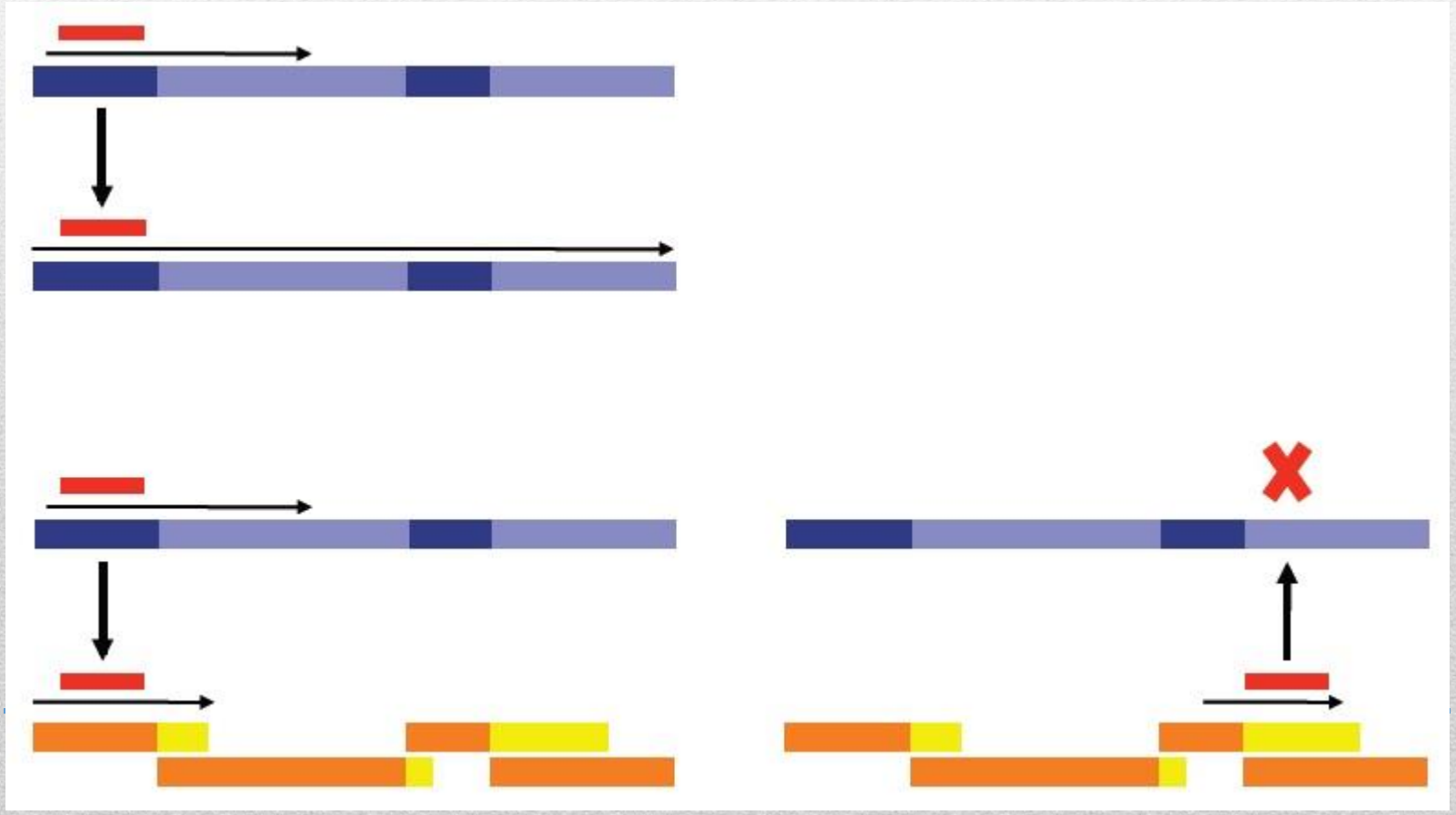
- **NB:** La feature **DEVE** essere definita nel sistema di coordinate di destinazione. In caso contrario Transfer fallisce

Ensembl core API:
Transfer



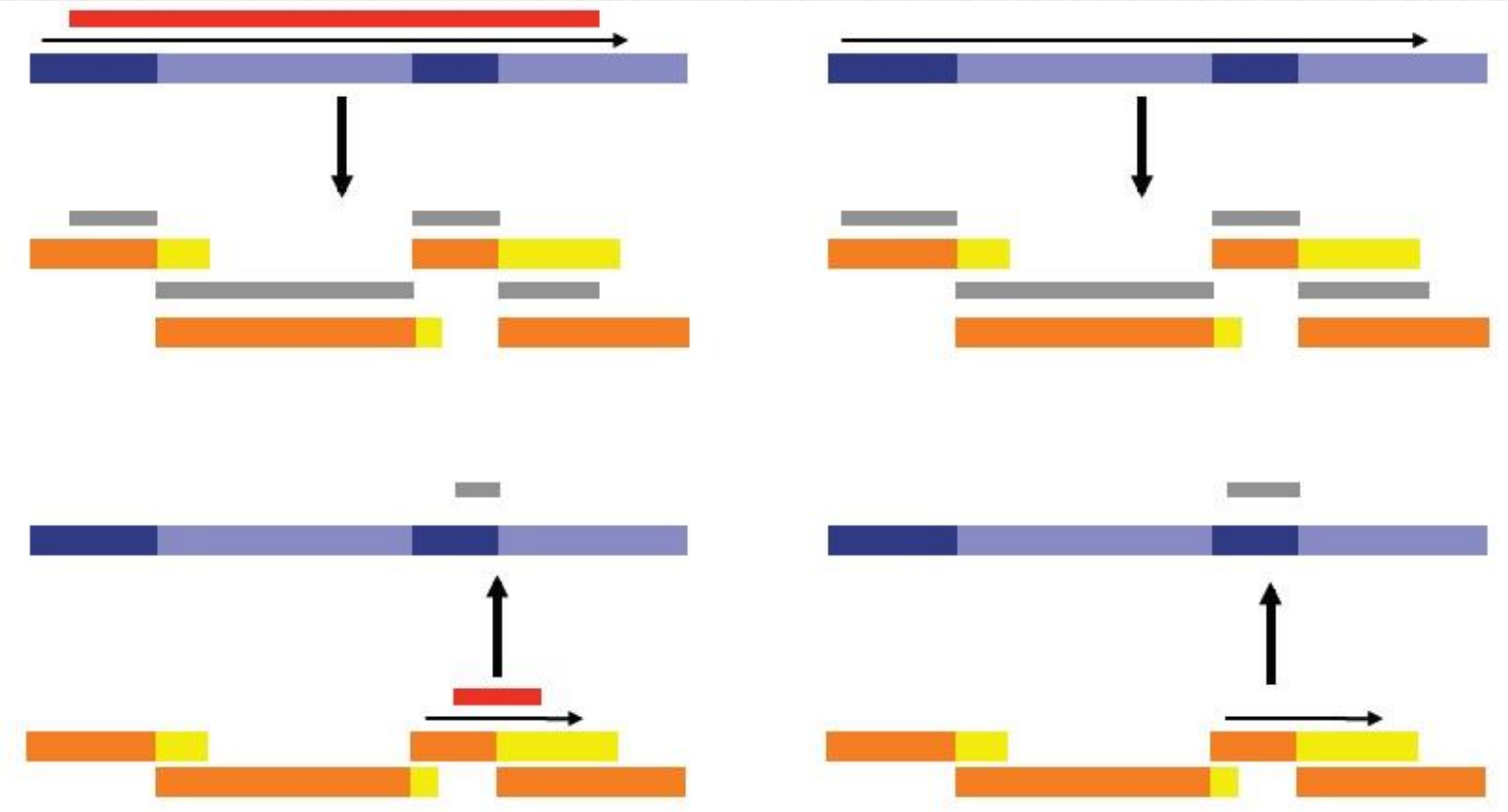
Ensembl core API:
Transform

- Agisce come Transfer ma Transform posiziona la feature su una **Slice** che copre tutta l'estensione della feature **nel sistema di coordinate di destinazione**.



Ensembl core API:
Project

- Project (nel senso di “proiezione”) **non “muove”** una feature o una slide da un sistema di coordinate ad un altro ma **restituisce le coordinate dell’oggetto nell’altro sistema di coordinate**.



Ensembl core API:
5 : Proiezione di un gene sui cloni

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $gene = $gene_adaptor->fetch_by_stable_id( 'ENSG00000155657' );
my $projection = $gene->project( 'clone' );
foreach my $segment ( @{$projection} ) {
   my $to_slice = $segment->to_Slice;
   print
   $gene->stable_id, ":",
   $segment->from_start, "-",
   $segment->from_end, " projects to ",
   $to_slice->coord_system_name, " ",
   $to_slice->seq_region_name, ":",
   $to_slice->start, "-",
   $to_slice->end, "[",
   $to_slice->strand, "]\n";
}
```

API: projection

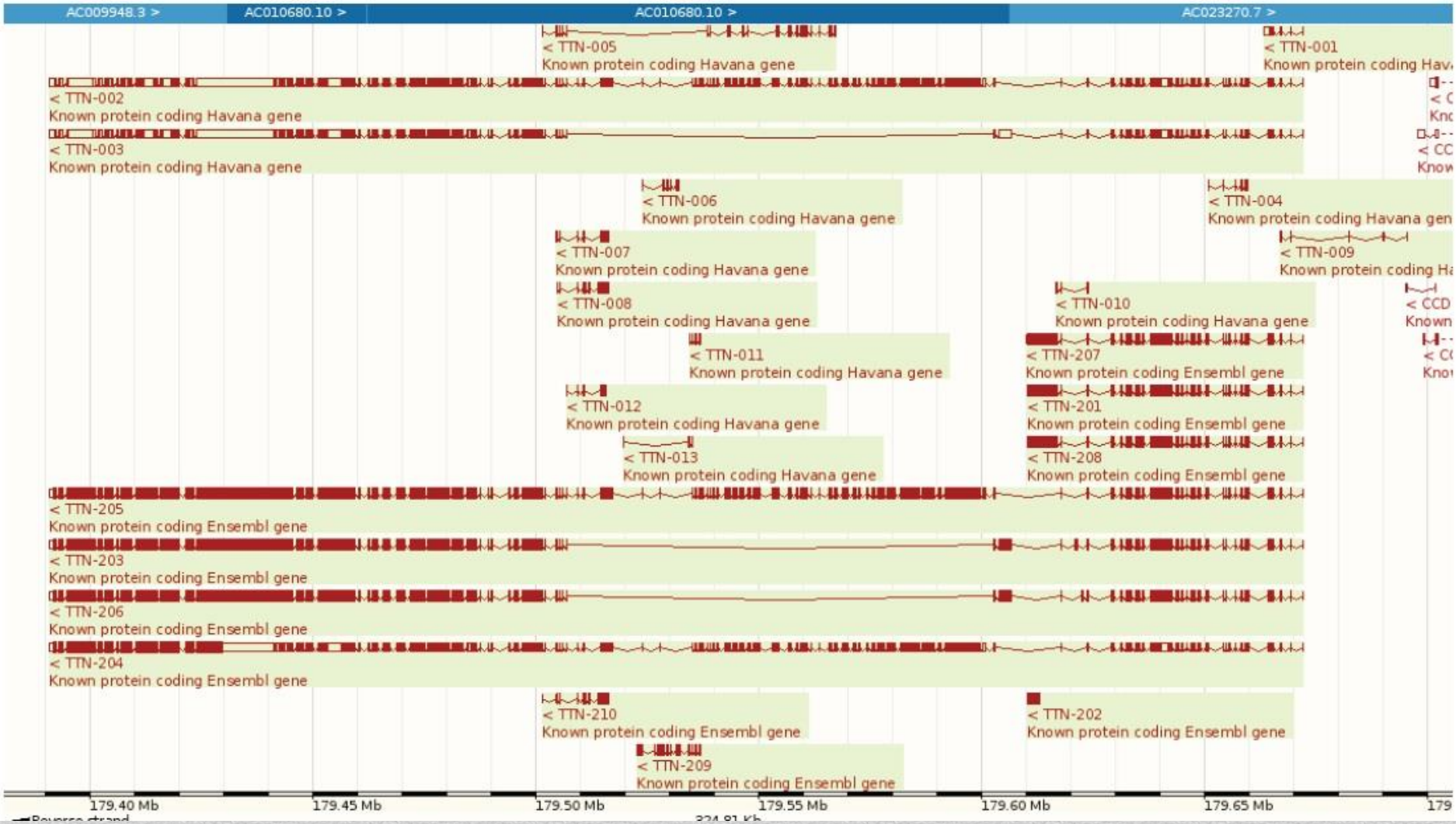
Ensembl core API:
5 : Proiezione di un gene sui cloni

Output:

ENSG00000155657 1-89287 projects to clone AC023270.7:1-89287[-1]
ENSG00000155657 89288-233135 projects to clone AC010680.10:31629-175476[-1]
ENSG00000155657 233136-233328 projects to clone FJ695199.1:1-193[-1]
ENSG00000155657 233329-264764 projects to clone AC010680.10:1-31436[-1]
ENSG00000155657 264765-304814 projects to clone AC009948.3:132579-172628[-1]

NBB: Projects mappa mediante segmentazione, ed è il metodo più flessibile per ottenere le coordinate di una feature/slice in un altro sistema di coordinate!

Ensembl core API:



Ensembl core API: esempi di utilizzo

Slice

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for the first 100 kb of chromosome 20
my $slice1 = $slice_adaptor->fetch_by_region( 'chromosome', '20', 1, 100000 );
## Fetch the slice for the gene ENSG00000101266 with 2kb of flanking sequence
my $slice2 = $slice_adaptor->fetch_by_gene_stable_id( 'ENSG00000101266', 2000 );
## Print information about both slices
foreach my $slice( $slice1, $slice2 ){
      print     "Slice:\t\t", $slice->name, "\n",
      "Coord system:\t", $slice->coord_system_name, "\n",
      "Seq region:\t", $slice->seq_region_name, "\n",
      "Start:\t\t", $slice->start, "\n",
      "End:\t\t", $slice->end, "\n",
      "Strand:\t\t", $slice->strand, "\n\n";}
```

Ensembl core API: esempi di utilizzo

Estrazione sequenza genomica

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for the first 100 kb of chromosome 20
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 1, 100000 );
## Get the corresponding soft-masked slice
my $soft_masked_slice = $slice->get_repeatmasked_seq(undef, 1);
## Print its sequence
print $soft_masked_slice->seq, "\n\n";
## Get the corresponding hard-masked slice
my $hard_masked_slice = $slice->get_repeatmasked_seq(undef, 0);
## Print its sequence
print $hard_masked_slice->seq, "\n";
```

Ensembl core API: esempi di utilizzo

Scrittura sequenza slice in file FASTA

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for the gene ENSG00000101266 with 2kb of flanking sequence
my $slice = $slice_adaptor->fetch_by_gene_stable_id( 'ENSG00000101266', 2000 );
## Create a Bio::SeqIO instance, define output as fasta format
my $output = Bio::SeqIO->new( -file=>'>2c.fa', -format=>'FASTA' );
## Print the sequence of the slice
$output->write_seq( $slice );
```

Ensembl core API: esempi di utilizzo

Conteggio di feature in una slice

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for the first 10 Mb of chromosome 20
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 1, 10000000 );
## Get the genes on the slicemy $genes = $slice->get_all_Genes;
## Get the total number of genes
my $number_of_genes = scalar(@$genes);
## Print this information
print "The number of genes on ", $slice->name, " is ", $number_of_genes, "\n";
```

Ensembl core API: esempi di utilizzo

Estrazione feature e loro posizione

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for chromosome 20:400000-500000
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 400000, 500000 );
## Get the repeat features on the slice
my $repeat_features = $slice->get_all_RepeatFeatures;
## Get the total number of repeat features
my $number_of_repeat_features = scalar(@$repeat_features);
## Print the total number of repeat features and the name and position of each
print "Total number of repeat features: ", $number_of_repeat_features, "\n\n";
foreach my $repeat_feature( @{$repeat_features} ){
     print    $repeat_feature->display_id, "\t",
              $repeat_feature->seq_region_name, ":",
              $repeat_feature->seq_region_start, "-",
              $repeat_feature->seq_region_end, "\n";}
```

Ensembl core API: esempi di utilizzo

Estrazione di allineamenti e loro posizione

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for chromosome 20:400000-500000
my $slice = $slice_adaptor->fetch_by_region( 'chromosome', '20', 400000, 500000 );
## Get all protein alignment features on the slice
my $proteinalignfeatures = $slice->get_all_ProteinAlignFeatures;
## Print information about all protein alignment features
foreach my $proteinalignfeature( @{$proteinalignfeatures} ){
       print    $proteinalignfeature->hseqname, "\t",
                $proteinalignfeature->hstart, "-",
                $proteinalignfeature->hend, "\t",
                $proteinalignfeature->seq_region_start, "-",
                $proteinalignfeature->seq_region_end, "\t",
                $proteinalignfeature->analysis->logic_name, "\n";}
```

MOLTO UTILE!

Ensembl core API: esempi di utilizzo

Estrazione di tutti i trascritti di un gene (e relative *traduzioni*)

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);
## Get the GeneAdaptor for human
my $gene_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Gene' );
## Fetch the gene with stable ID ENSG00000101266
my $gene = $gene_adaptor->fetch_by_stable_id( 'ENSG00000101266' );
## Get all transcripts for the gene
my $transcripts = $gene->get_all_Transcripts;
## Get the translation for each transcript
## Print information about the transcripts and translations
       foreach my $transcript( @{$transcripts} ){
          if ($transcript->translation) {
              print ">",  $transcript->stable_id, "|",
              $transcript->translation->stable_id, "\n",
              $transcript->translation->seq, "\n";
          }else{
              print ">", $transcript->stable_id, "\n",
              "THIS TRANSCRIPT HAS NO PROTEIN PRODUCT\n";  }}
```

Ensembl core API: esempi di utilizzo

External references (via gene symbol)

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the GeneAdaptor for human
my $gene_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Gene' );
## Fetch the gene(s) with external reference BRCA2_HUMAN
my $genes = $gene_adaptor->fetch_all_by_external_name( 'BRCA2_HUMAN' );
## Print information about the gene(s)
while ( my $gene = shift @{$genes} ){
     print    $gene->stable_id, "\n",
              $gene->external_name, "\n",
              $gene->description, "\n";}
```

Ensembl core API: esempi di utilizzo

External references (via gene symbol): estrazione cross-links

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the GeneAdaptor for human
my $gene_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Gene' );
## Fetch the gene(s) with external reference BRCA2_HUMAN
my $genes = $gene_adaptor->fetch_all_by_external_name( 'BRCA2_HUMAN' );
## Get all external references for the gene(s)
## Print information about the external references
    while ( my $gene = shift @{$genes} ){
        my $xrefs = $gene->get_all_DBLinks;
        foreach my $xref( @{$xrefs} ){
          print $xref->dbname, "\t", $xref->display_id, "\n";
        }
    }
```

Ensembl core API: esempi di utilizzo

External references: Mapping Gene Ontology → Ensembl

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the GeneAdaptor for human
my $gene_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Gene' );
## Fetch the gene(s) with external reference GO:0007126
my $genes = $gene_adaptor->fetch_all_by_GOTerm_accession( 'GO:0007126' );
## Print information about the gene(s)
while ( my $gene = shift @{$genes} ){
    print    $gene->stable_id, "\t",
    $gene->external_name, "\t",
    $gene->description, "\n";
}
```

Ensembl core API: esempi di utilizzo

Estrazione del PRIMO gene mappato su un clone

```
#!/usr/bin/perl
BEGIN{ push @INC,'C:/Perl64/site/lib/Bundle/','C:/Users/matt/ENSEMBL/ensembl-
api/ensembl/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
compara/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-
variation/modules','C:/Users/matt/ENSEMBL/ensembl-api/ensembl-functgenomics/modules';};

use strict;
use Bio::EnsEMBL::Registry;
use Getopt::Long;
use Bio::SeqIO;
my $registry = 'Bio::EnsEMBL::Registry';
$registry->load_registry_from_db(
-host => 'ensembldb.ensembl.org',
-user => 'anonymous'
);

## Get the SliceAdaptor for human
my $slice_adaptor = $registry->get_adaptor( 'Human', 'Core', 'Slice' );
## Fetch the slice for clone AL049761.11
my $slice = $slice_adaptor->fetch_by_region( 'clone', 'AL049761.11' );
## Get the first gene located on the clone
my $gene  = @{$slice->get_all_Genes}[0];
## Print the location of the gene
print $gene->stable_id, "\t", $gene->feature_Slice->name, "\n";
```

Indice = 0

Esercizi (core API)

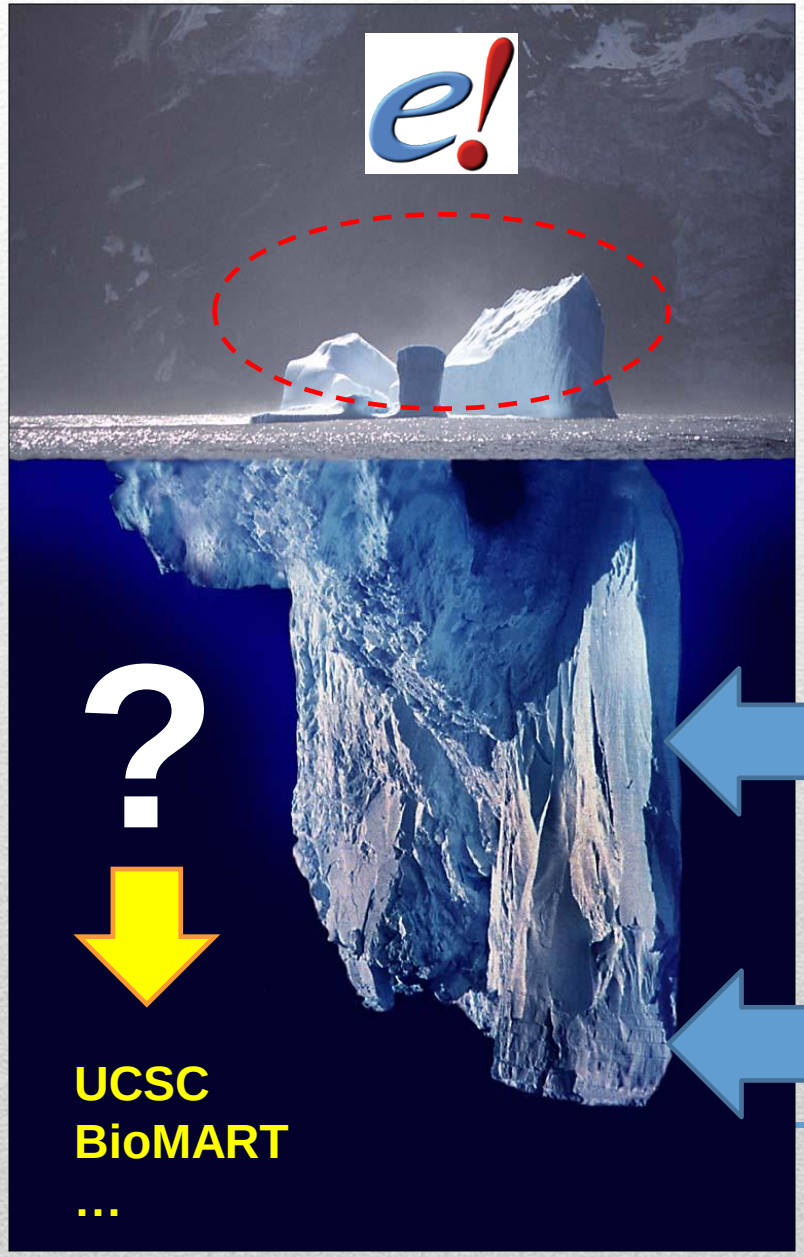
- Scrivete uno script che utilizzi l'API Ensembl core per estrarre tutti i geni codificanti proteine del cromosoma umano Y (suggerimento: per estrarre il biotipo di un oggetto gene via API potete fare così: $gene->biotype) (**4 pt**)

- Scrivete uno script che verifichi **se** sono presenti geni in almeno 3 regioni del genoma umano di 10 Kb e scelte in modo che la prima contenga esattamante un gene, la seconda contenga più di un gene e la terza non contenga nessun gene (per cercare regioni genomiche con queste caratetristiche **utilizzare il browser genomico ensembl** http://www.ensembl.org/Homo_sapiens/Info/Index), per ognuno dei geni identificati stampate posizione genomica (seq_region_name, seq_region_start, seq_region_end e seq_region_strand) (**8 pt**)

RIEPILOGO (I):

- Alcune banche dati (non tutte) sono disponibili sottoforma di database relazionali pubblici
- Possiamo interrogarle in vari modi (SQL/API dedicate)
- Il **linguaggio di programmazione** delle eventuali API disponibili varia da banca dati a banca dati
- Abbiamo visto due modalità d'accesso (SQL/API) alla componente core di un browser genomico (Ensembl) tra i più utilizzati (ma ne esistono altri!!!)
- Nel caso in cui una banca dati non permetta l'accesso diretto (SQL o API) solitamente permette scaricare I suoi dati sottoforma di file di testo… I dati sono sempre dati, ma così la loro gestione è **molto più complicata** (inoltre i files possono essere molto grandi (Gb)).

RIEPILOGO (II):



Anche se abbiamo visto «da vicino» una banca dati (Ensembl), abbiamo visto solo la «**punta dell'iceberg**» …

Per quanto riguarda **Ensembl**:

Altre API:

API **Variation** (variabilità genetica)
API **Compara** (Genomica comparata)
API **FunctionalGenomics** (il nome dice tutto)

SQL:
Databases per genomica comparata, genomica funzionale, variabilità genetica ecc … per **OGNI SPECIE !**

RIEPILOGO (II):

«INDIRIZZI UTILI»:

UCSC genome browser:

host: genome-mysql.cse.ucsc.edu
user: genome
access type: SQL

Vedere anche:

UCSC Table browser (interfaccia web per «costruire» query):
http://genome.ucsc.edu/cgi-bin/hgTables

BIOMART:

host: martdb.ensembl.org
user: anonymous
port: 5316
access type: SQL (API disponibile)