CMSC423: Bioinformatic Algorithms, Databases and Tools
Lecture 5

Biological databases

Sequence alignment

How the data get accessed

- Gene by gene/object by object – targeted at manual inspection of data
  - usually lots of clicking involved
  - simple search capability
  - similarity searches in addition to text queries
- Bulk – targeted at computational analyses
  - often programatic access through web server
  - most frequently – just bulk download (ftp)

NCBI - National Center for Biotech. Info.

- Virtually all biological data generated in the US gets stored here!
- One-stop-shop for biological data
- Primarily focused on gene-by-gene analyses
- Provides simple scripts for programatic access
- Provides ftp access for bulk downloads

http://www.ncbi.nlm.nih.gov

EMBL European Molecular Biology Lab.

- European version of NCBI
- BioMart query builder

http://www.ebi.ac.uk/embl/

Expasy proteomics server

- Home of Swisprot and other useful information on proteins

http://www.expasy.org

Kyoto Encyclopedia of Genes & Genomes

- Central repository of pathway information

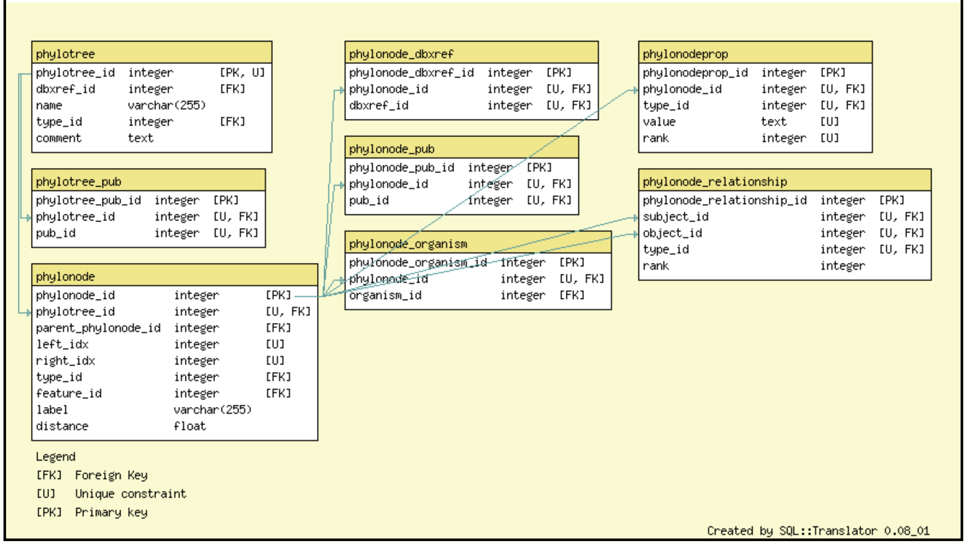
http://www.genome.jp/kegg/

Genome browsers

- UCSC Genome Browser – http://genome.ucsc.edu
- ENSEMBL Genome Browser – http://www.ensemble.org
- Gbrowse http://www.gmod.org

Direct database access - SQL

- CHADO schema – www.gmod.org



phylotree		
phylotree_id	integer	[PK, U]
dbxref_id	integer	[FK]
name	varchar(255)	
type_id	integer	[FK]
comment	text	

phylotree_pub		
phylotree_pub_id	integer	[PK]
phylotree_id	integer	[U, FK]
pub_id	integer	[U, FK]

phylonode		
phylonode_id	integer	[PK]
phylotree_id	integer	[U, FK]
parent_phylonode_id	integer	[FK]
left_idx	integer	[U]
right_idx	integer	[U]
type_id	integer	[FK]
feature_id	integer	[FK]
label	varchar(255)	
distance	float	

phylonode_dbxref		
phylonode_dbxref_id	integer	[PK]
phylonode_id	integer	[U, FK]
dbxref_id	integer	[U, FK]

phylonode_pub		
phylonode_pub_id	integer	[PK]
phylonode_id	integer	[U, FK]
pub_id	integer	[U, FK]

phylonode_organism		
phylonode_organism_id	integer	[PK]
phylonode_id	integer	[U, FK]
organism_id	integer	[FK]

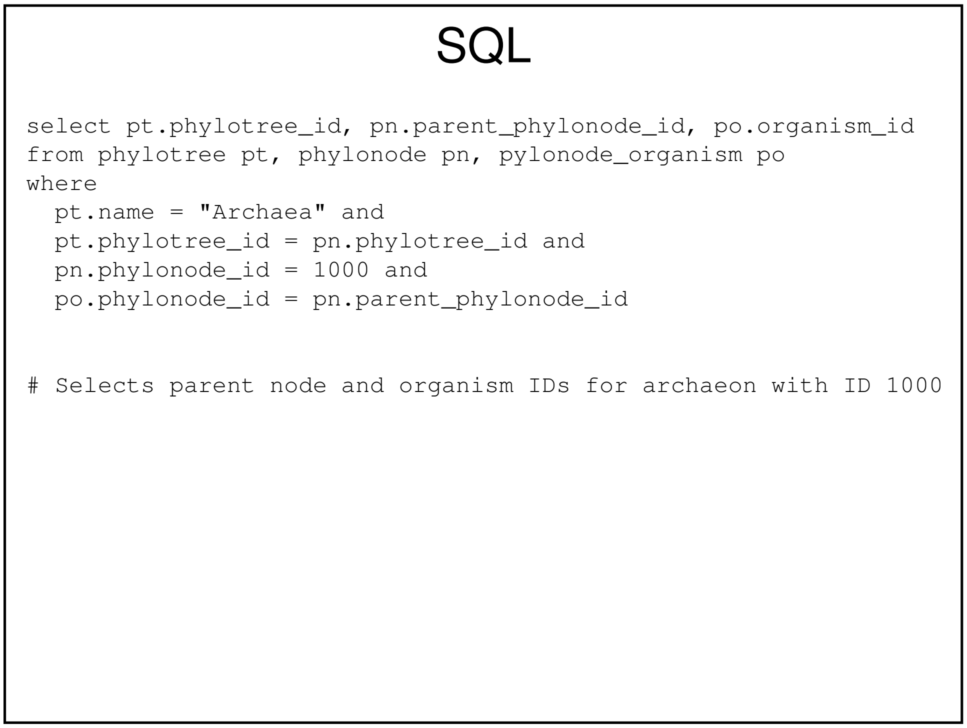
phylonodeprop		
phylonodeprop_id	integer	[PK]
phylonode_id	integer	[U, FK]
type_id	integer	[U, FK]
value	text	[U]
rank	integer	[U]

phylonode_relationship		
phylonode_relationship_id	integer	[PK]
subject_id	integer	[U, FK]
object_id	integer	[U, FK]
type_id	integer	[U, FK]
rank	integer	

Legend

[FK]	Foreign Key
[U]	Unique constraint
[PK]	Primary key

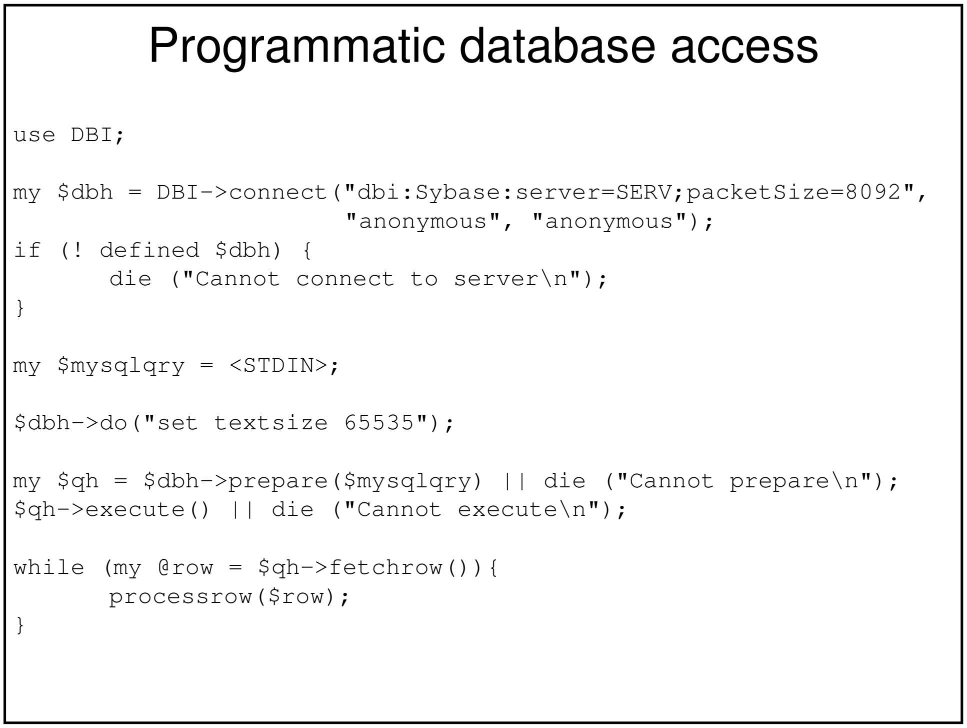
Created by SQL::Translator 0.08_01



SQL

```
select pt.phylotree_id, pn.parent_phylonode_id, po.organism_id
from phylotree pt, phylonode pn, pylonode_organism po
where
  pt.name = "Archaea" and
  pt.phylotree_id = pn.phylotree_id and
  pn.phylonode_id = 1000 and
  po.phylonode_id = pn.parent_phylonode_id


# Selects parent node and organism IDs for archaeon with ID 1000
```



Programmatic database access

```
use DBI;

my $dbh = DBI->connect("dbi:Sybase:server=SERV;packetSize=8092",
                       "anonymous", "anonymous");
if (! defined $dbh) {
       die ("Cannot connect to server\n");
}

my $mysqlqry = <STDIN>;

$dbh->do("set textsize 65535");

my $qh = $dbh->prepare($mysqlqry) || die ("Cannot prepare\n");
$qh->execute() || die ("Cannot execute\n");

while (my @row = $qh->fetchrow()){
       processrow($row);
}
```

NCBI programmatic access

- http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html
  - must write your own HTTP client (LWP Perl module helps)
  - queries go directly to web server
  - data returned in XML
- http://www.ncbi.nlm.nih.gov/Traces/trace.cgi?cmd=show&f=doc&m=obtain&s=stips
  - stub script provided (query_tracedb)
  - queries still go through web server
  - data returned in a variety of user selected formats
- For both, limits are set on the amount of data retrieved, e.g. less than 40,000 records at a time
- Download procedure:
  - figure out # of records to be retrieved ("count" query)
  - read data in allowable chunks
  - combine the chunks

Sequence alignment: exact matching

```
ACAGGTACAGTTCCCTCGACACCTACTACCTAAG        Text
CCTACT                                    Pattern
 CCTACT
  CCTACT
   CCTACT
```

```
for i = 0 .. len(Text) {
  for j = 0 .. len(Pattern) {
    if (Pattern[j] != Text[i]) go to next i
  }
  if we got there pattern matches at i in Text
}
```

Running time = O(len(Text) * len(Pattern)) = O(mn)

Worst case?

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAT
```

(m – n + 1) * n comparisons

Can we do better?

the Z algorithm (Gusfield)

For a string T, Z[i] is the length of the longest prefix of T[i..m] that matches a prefix of T.

T[1 .. Z[i]] = T[i .. i+Z[i] -1]