Relational Databases for Biologists Tutorial – ISMB02

Aaron J. Mackey
amackey@virginia.edu
and William R. Pearson
wrp@virginia.edu

http://www.people.virginia.edu/~wrp/papers/ismb02_sql.pdf

Why Relational Databases ?

- Large collections of well-annotated data
- Most public databases provide cross-links to other databases
  - NCBI GenBank:NCBI taxonomy
  - Gene Ontology:SwissProt human, mouse, fly, FlyBase, SGD
  - SwissProt:PFAM, SwissProt:Prosite
- Although cross-linking data is available, one cannot integrate all the related data in one query
- Individual research lab "Boutique" databases, integrating data of interest, are needed
- One-off, disposable, databases

Goals for the tutorial – Surveying the tools necessary to build “Boutique” databases

- Design and use of simple relational databases
- some theoretical background – What are “relations”, how can we manipulate them?
- using the entity relationship model for building cross-referenced databases
- building databases using mySQL–from very simple to a little more complicated
- resources for biological databases

🎓 = Advanced material

Tutorial Overview

- Introduction to Relational Databases
  - Relational implementations of Public databases
  - Motivation
    - Better search sensitivity
    - Better annotation
    - Managing results
  - Flatfiles are not relational
  - Glimpses of a relational database
- Relational Database Fundamentals
  - The Relational Model
    - operands - relations (tables)
      - tuples (records)
      - attributes (fields, columns)
    - operators - (select, join, …)
  - Basic SQL
  - Other SQL functions
- Designing Relational Databases
  - Designing a Sequence database
  - Entity-Relationship Models
  - Beyond Simple Relationships
    - hierarchical data
    - temporal data – historical integrity
- Using Relational Databases
  - Database Products
    - mySQL
    - postgreSQL
    - Commercial databases
  - Programming/Application interfaces
  - Prepackaged databases
    - bioSQL
    - ensembl
- Glossary

Tutorial Overview

- Introduction to Relational Databases
- Designing Relational Databases

Introduction to Relational Databases

- Relational Database Fundamentals
- Using Relational Databases

Introduction to Relational Databases

Relational databases in Biology – A brief history

- 1970's - 1985 The earliest "biological databases" – PIR protein database, Doolittle's protein database, Los Alamos GenBank, were distributed as "flat files"
- ~1990, when NCBI took over GenBank, moved to a relational implementation (Sybase)
- ~1991 (human) Genome Database (GDB, Sybase) at JHU, now at www.gdb.org (Hospital for Sick Children)
- ~1993 Mouse Genome Database (MGD) at informatics.jax.org
- Today, major public databases GenBank, EMBL, SwissProt, PIR, ENSEMBL are relational
- PIR ftp://nbrfa.georgetown.edu/pir_databases/psd/mysql/ and ENSEMBL www.ensembl.org provide relational downloads

Relational Databases in the Lab – Why?

- Too much data - work on subsets
  - Improving similarity search sensitivity
  - Improving similarity search strategies
- Interpreting results – finding all the annotations
  - adding functional annotations with ProSite
  - from expression to function
- Managing results

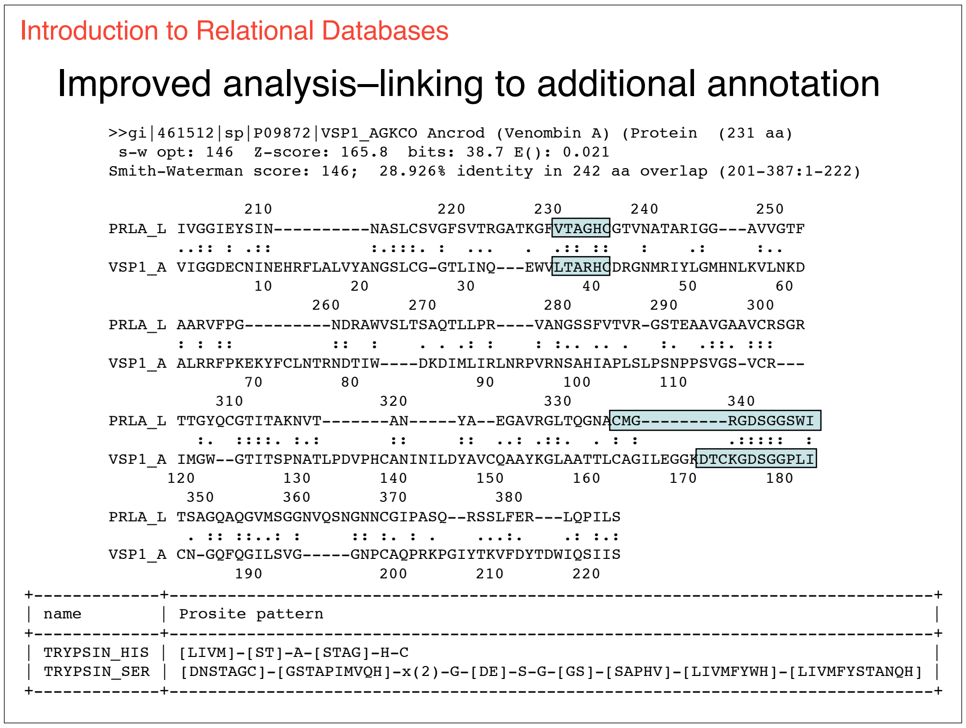
Too much data – work on subsets

- In similarity searching, the statistical significance of a result is linearly related to the size of the database searched.

  $E(x) = P(x)\, D$ $\qquad$ $P = 1x10^{-6}$

  $P(x)=1-\exp(-K\, m\, n\, \exp(-\lambda x))$ $\qquad$ *E. coli*: $D = \sim 4500,\ E = 4.5x10^{-3}$

  D= number of sequences $\qquad$ *nr*: $D = \sim 950{,}000,\ E = 0.95$
- Scoring matrices can be set to focus on evolutionary distances (BLOSUM62 and BLOSUM50 are effectively set to infinity. PAM20 – PAM40 are appropriate for distances of 100 – 200 My)
  - taxonomic subsets allow partial sequences (ESTs) to be identified more effectively
  - help distinguish orthologs from paralogs
- Gene expression measurements on large (6,000 – 30,000 genes) datasets reduce sensitivity. Search on pathways using Gene Ontology annotations



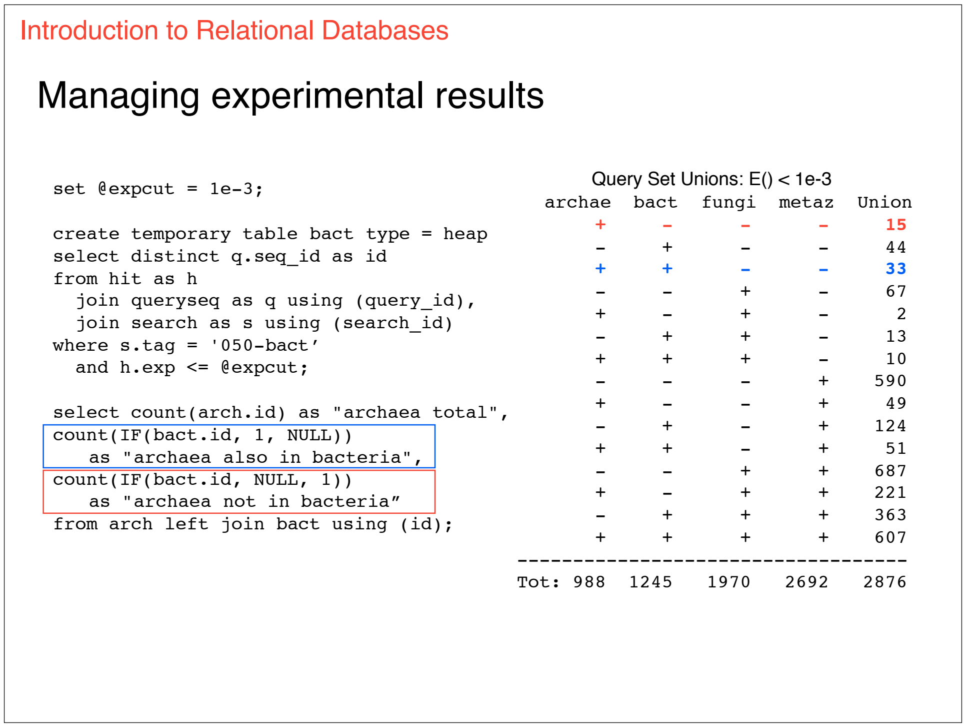
Introduction to Relational Databases

Improved analysis–linking to additional annotation

```
>>gi|461512|sp|P09872|VSP1_AGKCO Ancrod (Venombin A) (Protein  (231 aa)
 s-w opt: 146  Z-score: 165.8  bits: 38.7 E(): 0.021
Smith-Waterman score: 146;  28.926% identity in 242 aa overlap (201-387:1-222)

              210                 220       230       240          250
PRLA_L IVGGIEYSIN----------NASLCSVGFSVTRGATKGFVTAGHCGTVNATARIGG---AVVGTF
       ..:: : .::          :.:::. :  ...   .  .:: ::   :    .:     :..
VSP1_A VIGGDECNINEHRFLALVYANGSLCG-GTLINQ---EWVLTARHCDRGNMRIYLGMHNLKVLNKD
               10        20         30           40        50        60
                     260       270           280      290       300
PRLA_L AARVFPG---------NDRAWVSLTSAQTLLPR----VANGSSFVTVR-GSTEAAVGAAVCRSGR
       : : ::          ::  :    . . .: :    : :.. .. .   :.  .::. :::
VSP1_A ALRRFPKEKYFCLNTRNDTIW----DKDIMLIRLNRPVRNSAHIAPLSLPSNPPSVGS-VCR---
              70        80            90      100       110
           310              320            330                340
PRLA_L TTGYQCGTITAKNVT-------AN-----YA--EGAVRGLTQGNACMG---------RGDSGGSWI
         :.  ::::. :.:       ::     ::  ..: .::.  . : :         .:::::  :
VSP1_A IMGW--GTITSPNATLPDVPHCANINILDYAVCQAAYKGLAATTLCAGILEGGKDTCKGDSGGPLI
      120         130       140       150       160       170       180
        350       360       370         380
PRLA_L TSAGQAQGVMSGGNVQSNGNNCGIPASQ--RSSLFER---LQPILS
        . :: ::..: :     :: :. : .    ...:.    .: :.:
VSP1_A CN-GQFQGILSVG-----GNPCAQPRKPGIYTKVFDYTDWIQSIIS
             190            200       210       220
```

```
+-------------+-------------------------------------------------------------------------------+
| name        | Prosite pattern                                                               |
+-------------+-------------------------------------------------------------------------------+
| TRYPSIN_HIS | [LIVM]-[ST]-A-[STAG]-H-C                                                      |
| TRYPSIN_SER | [DNSTAGC]-[GSTAPIMVQH]-x(2)-G-[DE]-S-G-[GS]-[SAPHV]-[LIVMFYWH]-[LIVMFYSTANQH] |
+-------------+-------------------------------------------------------------------------------+
```



Introduction to Relational Databases

Managing experimental results

```
set @expcut = 1e-3;

create temporary table bact type = heap
select distinct q.seq_id as id
from hit as h
  join queryseq as q using (query_id),
  join search as s using (search_id)
where s.tag = '050-bact'
  and h.exp <= @expcut;

select count(arch.id) as "archaea total",
count(IF(bact.id, 1, NULL))
   as "archaea also in bacteria",
count(IF(bact.id, NULL, 1))
   as "archaea not in bacteria"
from arch left join bact using (id);
```

Query Set Unions: E() < 1e-3

archae	bact	fungi	metaz	Union
+	-	-	-	15
-	+	-	-	44
+	+	-	-	33
-	-	+	-	67
+	-	+	-	2
-	+	+	-	13
+	+	+	-	10
-	-	-	+	590
+	-	-	+	49
-	+	-	+	124
+	+	-	+	51
-	-	+	+	687
+	-	+	+	221
-	+	+	+	363
+	+	+	+	607
Tot: 988	1245	1970	2692	2876

Introduction to Relational Databases

Introduction to Relational Databases

- What is a relational database?
  - sets of tables and links (the data)
  - a language to query the database (**S**tructured **Q**uery **L**anguage)
  - a program to manage the data (RDBMS)
- Relational databases – the traditional view
  - manage transactions (bank deposits/withdrawals, airline reservations, Amazon purchases/inventory)
  - A C I D – Atomicity Consistency Isolation Durability
- Biological databases are "Read Only"
  - most data from other archival sources
  - few transactions
  - queries 99.999% select/join/where

Introduction to Relational Databases

Most Biological "databases" are "flat files"

FASTA format:

attribute type	*data*
annotation:	`>gi|121735|sp|P09488|GTM1_HUMAN Glutathione S-transferase Mu (GSTM1-1)(GTH4) (GSTM1A-1A) (GSTM1B-1B) (GST class-Mu 1)`

sequence:	`MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL`
	`PYLIDGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNpef`
	`eklkpkyleelpeklklYSEFLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPN`
	`LKDFISRFEGLEKISAYMKSSRFLPRPVFSKMAVWGNK`

annotation:	`>gi|232204|sp|P28161|GTM2_HUMAN Glutathione S-transferase Mu 2 (GSTM2-2) (GST class-Mu 2)`

sequence:	`MPMTLGYWNIRGLAHSIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL`
	`PYLIDGTHKITQSNAILRYIARKHNLCGESEKEQIREDILENQFMDSRMQLAKLCYDPDF`
	`EKLKPEYLQALPEMLKLYSQFLGKQPWFLGDKITFVDFIAYDVLERNQVFEPSCLDAFPN`
	`LKDFISRFEGLEKISAYMKSSRFLPRPVFTKMAVWGNK`

```
>gi|232204|sp|P28161|GTM2_HUMAN Glutathione S-transferase Mu 2 (GST class-Mu 2)
    ↑       ↑  ↑      ↑           ↑
    gi      db sp_acc sp_name     description
```

Introduction to Relational Databases

EMBL/
Swissprot
flatfiles

attribute type *data*

```
ID   GTM1_HUMAN     STANDARD;      PRT;   217 AA.
AC   P09488;
DT   01-MAR-1989 (REL. 10, CREATED)
DT   01-FEB-1991 (REL. 17, LAST SEQUENCE UPDATE)
DT   01-NOV-1995 (REL. 32, LAST ANNOTATION UPDATE)
DE   GLUTATHIONE S-TRANSFERASE MU 1 (EC 2.5.1.18) (GSTM1-1) (HB SUBUNIT 4)
DE   (GTH4) (GSTM1A-1A) (GSTM1B-1B) (CLASS-MU).
GN   GSTM1 OR GST1.
OS   HOMO SAPIENS (HUMAN).
OC   EUKARYOTA; METAZOA; CHORDATA; VERTEBRATA; TETRAPODA; MAMMALIA;
OC   EUTHERIA; PRIMATES.
RN   [2]
RP   SEQUENCE FROM N.A.
RX   MEDLINE; 89017184.
RA   SEIDEGAERD J., VORACHEK W.R., PERO R.W., PEARSON W.R.;
RL   PROC. NATL. ACAD. SCI. U.S.A. 85:7293-7297(1988).
CC   -!- FUNCTION: CONJUGATION OF REDUCED GLUTATHIONE TO A WIDE NUMBER
CC       OF EXOGENOUS AND ENDOGENOUS HYDROPHOBIC ELECTROPHILES.
CC   -!- CATALYTIC ACTIVITY: RX + GLUTATHIONE = HX + R-S-G.
CC   -!- SUBUNIT: HOMODIMER.
CC   -!- SUBCELLULAR LOCATION: CYTOPLASMIC.
CC   -!- TISSUE SPECIFICITY: THIS IS A LIVER ISOZYME.
CC   -!- SIMILARITY: BELONGS TO THE GST SUPERFAMILY, MU FAMILY.
DR   EMBL; X08020; G31924; -.
DR   PIR; S01719; S01719.
DR   HSSP; P28161; 1HNA.
DR   MIM; 138350; -.
KW   TRANSFERASE; MULTIGENE FAMILY; POLYMORPHISM.
FT   INIT_MET      0      0
FT   VARIANT     172    172       K -> N (IN ALLELE B).
FT   CONFLICT     43     43       S -> T (IN REF. 3).
SQ   SEQUENCE   217 AA;  25580 MW;  9A7AAFCB CRC32;
     PMILGYWDIR GLAHAIRLLL EYTDSSYEEK KYTMGDAPDY DRSQWLNEKF KLGLDFPNLP
 .□.□.
//
```

Introduction to Relational Databases

Genbank/
Genpept
flatfiles

attribute type *data*

```
LOCUS       GTM1_HUMAN               218 aa            linear   PRI 16-OCT-2001
DEFINITION  Glutathione S-transferase Mu 1 (GSTM1-1) (HB subunit 4) (GTH4)
            (GSTM1A-1A) (GSTM1B-1B) (GST class-Mu 1).
ACCESSION   P09488
VERSION     P09488  GI:121735
DBSOURCE    swissprot: locus GTM1_HUMAN, accession P09488;
            created: Mar 1, 1989.
            xrefs: gi: gi: 31923, gi: gi: 31924, gi: gi: 183668, gi: gi:
            xrefs (non-sequence databases): MIM 138350, InterPro IPR004046,
            InterPro IPR004045, InterPro IPR003081, Pfam PF00043, Pfam PF02798,
            PRINTS PR01267
KEYWORDS    Transferase; Multigene family; Polymorphism; 3D-structure.
SOURCE      human.
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
REFERENCE   2  (residues 1 to 218)
  AUTHORS   Seidegard,J., Vorachek,W.R., Pero,R.W. and Pearson,W.R.
  TITLE     Hereditary differences in the expression of the human glutathione
            transferase active on trans-stilbene oxide are due to a gene deletion
  JOURNAL   Proc. Natl. Acad. Sci. U.S.A. 85 (19), 7293-7297 (1988)
  MEDLINE   89017184
FEATURES             Location/Qualifiers
     source          1..218
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606"
     Protein         1..218
                     /product="Glutathione S-transferase Mu 1"
                     /EC_number="2.5.1.18"
     Region          173
                     /region_name="Variant"
                     /note="K -> N (IN ALLELE B). /FTId=VAR_003617."
ORIGIN
        1 mpmilgywdi rglahairll leytdssyee kkytmgdapd ydrsqwlnek fklgldfpnl
//
```

Flat files are not Relational

- Data type (attribute) is part of the data
- Record order matters
- Multiline records
- Massive duplication–60,000 duplicate lines:

```
SOURCE      human.
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
```

- Some records are hierarchical

```
DBSOURCE    swissprot: locus GTM1_HUMAN, accession P09488;
            created: Mar 1, 1989.
            xrefs: gi: gi: 31923, gi: gi: 31924, gi: gi: 183668, gi: gi:
            xrefs (non-sequence databases): MIM 138350, InterPro IPR004046,
            InterPro IPR004045, InterPro IPR003081, Pfam PF00043, Pfam PF02798,
            PRINTS PR01267
```

- Records contain multiple “sub-records”
- Implicit “Key”

A relational database for sequences

```
mysql> show tables;
+---------------------+
| Tables_in_seq_demo  |
+---------------------+
| annot, prot, sp     |
+---------------------+
```

```
mysql> describe prot;
+---------+------------------+-----+---------+----------------+
| Field   | Type             | Key | Default | Extra          |
+---------+------------------+-----+---------+----------------+
| prot_id | int(10) unsigned | PRI | NULL    | auto_increment |
| seq     | text             |     |         |                |
| len     | int(10) unsigned |     | 0       |                |
+---------+------------------+-----+---------+----------------+
mysql> describe annot;
+---------+-----------------------------------+-----+---------+-------+
| Field   | Type                              | Key | Default | Extra |
+---------+-----------------------------------+-----+---------+-------+
| prot_id | int(10) unsigned                  | MUL | 0       |       |
| gi      | int(10) unsigned                  | MUL | 0       |       |
| db      | enum('gb','emb','pdb','pir','sp') | MUL | gb      |       |
| acc     | varchar(255)                      | PRI | ''      |       |
| descr   | text                              |     |         |       |
+---------+-----------------------------------+-----+---------+-------+
mysql> describe sp;
+-------+------------------+-----+---------+-------+
| Field | Type             | Key | Default | Extra |
+-------+------------------+-----+---------+-------+
| gi    | int(10) unsigned | PRI | 0       |       |
| name  | varchar(10)      |     | NULL    |       |
+-------+------------------+-----+---------+-------+
```

Introduction to Relational Databases

NCBI nr entry for human GSTM1:

```
>gi|11428198|ref|XP_002155.1| similar to glutathione S-transferase M4 (H. sapiens)[Homo sapiens]
 gi|121735|sp|P09488|GTM1_HUMAN GLUTATHIONE S-TRANSFERASE MU 1 (GSTM1-1) (GTH4) (GST CLASS-MU)
 gi|87551|pir||S01719 glutathione transferase (EC 2.5.1.18) class mu, GSTM1 - human
 gi|31924|emb|CAA30821.1| (X08020) glutathione S-transferase (AA 1-218) [Homo sapiens]

MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNLPYLIDGAHKI
TQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEFEKLKPKYLEELPEKLKLYSE
FLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFS
KMAVWGNK
```

mySQL tables:

```
prot:
+-----------+-----+-----+---------+---------------------------------------------------+
| prot_id   | len | pi  | mw      | seq                                               |
+-----------+-----+-----+---------+---------------------------------------------------+
| 6906      | 218 | 6.2 | 25712.1 | MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRS ... |
+-----------+-----+-----+---------+---------------------------------------------------+

annot:
+---------+----------+-----+-------------+----------------------------------------------------+
| prot_id | gi       | db  | acc         | descr                                              |
+---------+----------+-----+-------------+----------------------------------------------------+
|    6906 | 11428198 | ref | XP_002155.1 | glutathione S-transferase M4 [Homo sapiens]        |
|    6906 |   121735 | sp  | P09488      | GLUTATHIONE S-TRANSFERASE MU 1    (GST CLASS-MU)   |
|    6906 |    87551 | pir | S01719      | glutathione transferase class mu, GSTM1 - human    |
|    6906 |    31924 | emb | CAA30821.1  | glutathione S-transferase (AA 1-218) [Homo sapiens]|
+---------+----------+-----+-------------+----------------------------------------------------+
```

Introduction to Relational Databases

Moving through a relational database

```
Annot:
+------------+--------+------------+-----+----------------------------------------------------+
| protein_id | gi     | acc        | db  | descr                                              |
+------------+--------+------------+-----+----------------------------------------------------+
|       6906 | 121735 | P09488     | sp  | GLUTATHIONE S-TRANSFERASE MU 1 (GTM1)(GST CLASS-MU)|
|       6906 |  87551 | S01719     | pir | glutathione transferase (EC 2.5.1.18)  GSTM1 human |
|       6906 |  31924 | CAA30821.1 | emb | glutathione S-transferase (AA 1-218) [Homo sapiens]|
+------------+--------+------------+-----+----------------------------------------------------+

mysql> select * from sp where sp.gi=121735;
+--------+------------+
| gi     | name       |
+--------+------------+
| 121735 | GTM1_HUMAN |
+--------+------------+

mysql> select * from swisspfam where sp_acc = "P09488";
+--------+----------+-------+-----+
| sp_acc | pfam_acc | begin | end |
+--------+----------+-------+-----+
| P09488 | PF00043  |    87 | 191 |
| P09488 | PF02798  |     1 |  81 |
| P09488 | PB002869 |   192 | 217 |
+--------+----------+-------+-----+

mysql> select * from pfam where acc = "PF00043";
+---------+-------+----------------------------------------------+-------+-----+
| acc     | name  | descr                                        | class | len |
+---------+-------+----------------------------------------------+-------+-----+
| PF00043 | GST_C | Glutathione S-transferase, C-terminal domain | A     | 121 |
+---------+-------+----------------------------------------------+-------+-----+
```

Tutorial Overview

- Introduction to Relational Databases
- Designing Relational Databases

Relational Database Fundamentals

- Relational Database Fundamentals
- Using Relational Databases

Relational Database Fundamentals

- The Relational Model – relational algebra
  - operands - **relations** (tables)
    - **tuples** (records)
    - **attributes** (fields, columns)
  - operators - (select, join, …)
- Basic SQL
  - `SELECT` [attribute list] (columns)
  - `FROM` [relation]
  - `WHERE` [condition]
  - `JOIN` - `NATURAL, INNER, OUTER`
- Other SQL functions
  - `COUNT()`
  - `MAX(), MIN(), AVE()`
  - `DISTINCT`
  - `ORDER BY`
  - `GROUP BY`
  - `LIMIT`

A simpler relational database

protein relation (table)

degree = 4

tuples (rows)

cardinality = 4

prot_id	name	seq	species_id
1	`GTM1_HUMAN`	`MGTSHSMT...`	1
2	`GTM1_RAT`	`MGYTVSIT...`	3
3	`GTM1_MOUSE`	`MGSTKMLT...`	2
4	`GTM2_HUMAN`	`MGTSHSMT...`	1

species relation (table)

species_id	name	scientific_name
1	`human`	`Homo sapiens`
2	`mouse`	`Mus musculus`
2	`house mouse`	`Mus musculus`
3	`rat`	`Rattus rattus`

Properties of *Relations* (tables)

- No two *tuples* (records, rows) are exactly the same; at least one *attribute* (field, column) value will differ between any two *tuples*
- *tuples* are in no particular order;
- Within each *tuple* the *attributes* have no particular order
- Each *attribute* contains exactly one value; no aggregate or complex values are allowed (e.g. lists or other composite structures).

Relational Algebra – Operations

1. **Restrict**: remove *tuples* (rows) that don't satisfy some criteria.
2. **Project**: remove specified attributes (columns, fields);
3. **Product**: merge *tuple* pairs from two relations in all possible ways; both degree and cardinality increase;
4. **Join**: Like ``Product'', but merged *tuple* pairs must satisfy some criteria for joining, otherwise the pair is removed
5. **Union**: concatenation of all *tuples* from two relations; degree remains the same, cardinality increases;
6. **Intersection**: remove *tuples* that are not shared by both relations
7. **Difference**: remove *tuples* that are not shared by one of the relations
8. **Divide**: Difficult to explain and generally unused.

Relational Algebra – Operations

1. **Restrict**: remove *tuples* (rows) that don't satisfy some criteria.

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1

```
restrict on (species_id = 1)
```

=

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
4	GTM2_HUMAN	MGTSHSMT...	1

Relational Algebra – Operations

1. **Restrict**: remove *tuples* (rows) that don't satisfy some criteria.
2. **Project**: remove specified attributes (columns, fields);

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
4	GTM2_HUMAN	MGTSHSMT...	1

```
project over (name, sequence)
```

=

name	sequence
GTM1_HUMAN	MGTSHSMT...
GTM2_HUMAN	MGTSHSMT...

Relational Algebra – Operations

3. **Product**: merge *tuple* pairs from two relations in all possible ways; both degree and cardinality increase;

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1

X

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

=

protein_id	name	sequence	p.sid	s.sid	name	scientific name
1	GTM1_HUMAN	MGTSHSMT...	1	1	human	Homo sapiens
2	GTM1_RAT	MGYTVSIT...	3	1	human	Homo sapiens
3	GTM1_MOUSE	MGSTKMLT...	2	1	human	Homo sapiens
4	GTM2_HUMAN	MGTSHSMT...	1	1	human	Homo sapiens
1	GTM1_HUMAN	MGTSHSMT...	1	2	mouse	Mus musculus
2	GTM1_RAT	MGYTVSIT...	3	2	mouse	Mus musculus
3	GTM1_MOUSE	MGSTKMLT...	2	2	mouse	Mus musculus
4	GTM2_HUMAN	MGTSHSMT...	1	2	mouse	Mus musculus
1	GTM1_HUMAN	MGTSHSMT...	1	3	rat	Rattus rattus
2	GTM1_RAT	MGYTVSIT...	3	3	rat	Rattus rattus
3	GTM1_MOUSE	MGSTKMLT...	2	3	rat	Rattus rattus
4	GTM2_HUMAN	MGTSHSMT...	1	3	rat	Rattus rattus

Relational Database Fundamentals

Relational Algebra – Operations

4. **Join**: Like ``Product", but merged *tuple* pairs must satisfy some criteria for joining, otherwise the pair is removed

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

```
join on (A.species_id = B.species_id)
```

=

protein_id	name	sequence	p.sid	s.sid	name	scientific name
1	GTM1_HUMAN	MGTSHSMT...	1	1	human	Homo sapiens
4	GTM2_HUMAN	MGTSHSMT...	1	1	human	Homo sapiens
3	GTM1_MOUSE	MGSTKMLT...	2	2	mouse	Mus musculus
2	GTM1_RAT	MGYTVSIT...	3	3	rat	Rattus rattus

Relational Database Fundamentals

From relational *algebra* to SQL:

Both sets of operations below accomplish the same thing:
"Show me the descriptions from human sequences"

1. **Join** sequence and species *tuples* over `species_id (from)`
2. **Restrict** the result on `(where) species□name□=□"human"`
3. **Project** the result over the attribute `(select) "description"`

1. **Restrict** the species *tuples* on `species□name□=□"human"`
2. **Project** the result over the attribute `species_id`
3. **Project** the sequence *tuples* over the attributes `sequence_id` and `species_id`
4. **Join** the two projections over the attribute `species_id`
5. **Project** the result over the attribute `sequence_id`
6. **Join** the result to the sequence table over `sequence_id`
7. **Project** the result over the attribute `description`

SQL is a declarative language: describe what you want, not how to obtain it:

```
select description
from sequence join species using (species_id)
where species.name = 'human"
```

Relational Database Fundamentals

SQL - Structured Query Language

- DDL - Data Definition Language
  - `CREATE DATABASE seqdb`

  -
    ```
    CREATE TABLE protein (
      id INT PRIMARY KEY AUTOINCREMENT
      seq TEXT
      len INT)
    ```
  - `ALTER TABLE ...`
  - `DROP TABLE protein, DROP DATABASE seqdb`

- DML - Data Manipulation Language
  - `SELECT` : calculate new relations via *Restrict*, *Project* and *Join* operations
  - `UPDATE` : make changes to existing tuples
  - `INSERT` : add new tuples to a relation
  - `DELETE` : remove tuples from a relation

Relational Database Fundamentals

Extracting data with SQL: `SELECT`-ing attributes

```
SELECT [attribute list]
FROM   [relation]
```

```
SELECT prot_id, protein.description,
 species.name
FROM   [relation]
```

```
SELECT prot_id, protein.description AS
  descr, species.name AS sname
FROM   [relation]
```

```
SELECT *
FROM   [relation]
```

```
SELECT protein.*, species.name AS sname
FROM   [relation]
```

Relational Database Fundamentals

Extracting data with SQL: specifying relations with FROM

```
SELECT [attribute list]
FROM   [relation]
```

Return attributes from all tuples:

```
SELECT prot_id                SELECT name
FROM   protein                FROM   species
```

Return attributes from tuples with conditions:

```
SELECT name FROM protein
WHERE name LIKE “glutathione %”

SELECT species_id FROM species
WHERE name LIKE “%mouse%”

SELECT name, seq FROM protein
WHERE species_id = 2
```

Relational Database Fundamentals

Extracting data: combining relations with JOIN

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

- Product: merge *tuple* pairs from two relations in all possible ways

```
SELECT protein.*,
       species.*
FROM   protein
       JOIN species
```

protein_id	name	sequence	p.sid	s.sid	name
1	GTM1_HUMAN	MGTSHSMT...	1	1	human
2	GTM1_RAT	MGYTVSIT...	3	1	human
3	GTM1_MOUSE	MGSTKMLT...	2	1	human
4	GTM2_HUMAN	MGTSHSMT...	1	1	human
1	GTM1_HUMAN	MGTSHSMT...	1	2	mouse
2	GTM1_RAT	MGYTVSIT...	3	2	mouse
3	GTM1_MOUSE	MGSTKMLT...	2	2	mouse
4	GTM2_HUMAN	MGTSHSMT...	1	2	mouse
1	GTM1_HUMAN	MGTSHSMT...	1	3	rat
2	GTM1_RAT	MGYTVSIT...	3	3	rat
3	GTM1_MOUSE	MGSTKMLT...	2	3	rat
4	GTM2_HUMAN	MGTSHSMT...	1	3	rat

Relational Database Fundamentals

Extracting data: combining relations with `JOIN`

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

- Product: merge *tuple* pairs from two relations in all possible ways
- Join: Like ``Product'', but merged tuple pairs must satisfy some criteria for joining, otherwise the pair is removed

```
SELECT protein.*,
       species.name
FROM   protein
       JOIN species USING (species_id)
```

protein_id	name	sequence	species_id	name
1	GTM1_HUMAN	MGTSHSMT...	1	human
4	GTM2_HUMAN	MGTSHSMT...	1	human
3	GTM1_MOUSE	MGSTKMLT...	2	mouse
2	GTM1_RAT	MGYTVSIT...	3	rat

Relational Database Fundamentals

Combining relations with `JOIN`

```
SELECT protein.name, protein.sequence
FROM   protein JOIN species USING (species_id)
WHERE  species.name = ‘mouse’;
```

`JOIN:`

protein_id	name	sequence	species_id	name	scientific_name
1	GTM1_HUMAN	MGTSHSMT...	1	human	Homo sapiens
2	GTM1_RAT	MGYTVSIT...	3	rat	Rattus rattus
3	GTM1_MOUSE	MGSTKMLT...	2	mouse	Mus musculus
4	GTM2_HUMAN	MGTSHSMT...	1	human	Homo sapiens

`WHERE:`

protein_id	name	sequence	species_id	name	scientific_name
3	GTM1_MOUSE	MGSTKMLT...	2	mouse	Mus musculus

`SELECT:`

name	sequence
GTM1_MOUSE	MGSTKMLT...

WHERE clauses further restrict the relation

```
SELECT protein.description
FROM   protein JOIN species USING (species_id)
WHERE  species.name = "human"
AND    (
         protein.length > 100
OR       protein.pI < 8.0
       )

SELECT protein.description
FROM   ( protein
        JOIN species USING (species_id)
       )
WHERE  species.name = "human"
AND    ( protein.length > 100 OR protein.pI < 8.0 )
```

Output modifiers

```
SELECT sequence
FROM   protein
LIMIT  10


SELECT sequence
FROM   protein
ORDER BY length ASC


SELECT species.name, protein.description, protein.length
FROM   protein JOIN species USING (species_id)
WHERE  length > 100
ORDER BY species.name ASC, length DESC
LIMIT 1
```

Relational Database Fundamentals

Different forms of “JOIN”

- `A JOIN B USING (attribute)`
  `(join with condition A.attr = B.attr)`
- `A NATURAL JOIN B`
  (join using <u>all</u> common attributes)
- `A INNER JOIN B ON (condition)`
  (join using a specified condition)

- `A LEFT [OUTER] JOIN B ON (condition)`
- `A RIGHT [OUTER] JOIN B ON (condition)`
- `A FULL OUTER JOIN B ON`
  - Avoid losing tuples with `NULL` attributes
  - Retain tuples lost by `[INNER] JOIN`
  - `LEFT JOIN` – maintain tuples to left
  - `RIGHT JOIN` – maintain tuples to right

Relational Database Fundamentals

protein_id	name	sequence	species_id
1	GTM1_HUMAN	MGTSHSMT...	1
2	GTM1_RAT	MGYTVSIT...	3
3	GTM1_MOUSE	MGSTKMLT...	2
4	GTM2_HUMAN	MGTSHSMT...	1
5	GTT1_DROME	MVDFYYLP...	NULL

species_id	name	scientific_name
1	human	Homo sapiens
2	mouse	Mus musculus
3	rat	Rattus rattus

```
SELECT protein.name,
       species.name
FROM   protein
       JOIN species
       USING (species_id)
```

name	name
GTM1_HUMAN	human
GTM2_HUMAN	human
GTM1_MOUSE	mouse
GTM1_RAT	rat

```
SELECT protein.name,
       species.name
FROM   protein
       LEFT JOIN species
       USING (species_id)
```

name	name
GTM1_HUMAN	human
GTM2_HUMAN	human
GTM1_MOUSE	mouse
GTM1_RAT	Rat
GTT1_DROME	NULL

Relational Database Fundamentals

Additional SQL functions

- **DISTINCT** (or DISTINCTROW)

  This statement …
  ```
  SELECT species.name
  FROM   species JOIN protein USING (species_id)
  WHERE  sequence.length < 100
  ```
  … produces duplicated species lines for each protein, but this one …
  ```
  SELECT DISTINCT species.name
  FROM   species JOIN protein USING (species_id)
  WHERE  sequence.length < 100
  ```
  … only produces unique (or *distinct*) species lines.
- **COUNT(*)** returns the number of *tuples*, rather than their values
  ```
  SELECT COUNT(*) FROM protein
  ```
- **COUNT**(**DISTINCT** *attribute*)
  ```
  SELECT COUNT(DISTINCT species.name)
  FROM   species JOIN protein USING (species_id)
  WHERE  sequence.length < 100
  ```
- **MAX**(), **MIN**(), **AVE**() - aggregate functions on “grouped” tuples:
- **GROUP BY**
  ```
  SELECT species.name, MIN(length), MAX(length), AVE(length)
  FROM   species JOIN protein USING (species_id)
  GROUP BY species.name
  ORDER BY species.name ASC
  LIMIT 10
  ```

Tutorial Overview

- Introduction to Relational Databases
- Designing Relational Databases

Short Break

- Relational Database Fundamentals
- Using Relational Databases

Tutorial Overview

- Introduction to Relational Databases
- Designing Relational Databases

Designing Relational Databases

- Relational Database Fundamentals
- Using Relational Databases

Designing Relational Databases

- Reducing data redundancy: Normalization
- Maintaining connections between data: Primary and Foreign Keys
- Normalization by semantics: the *Entity Relationship* Model
- “One-to-Many” and “Many-to-Many” Relationships
- Entity Polymorphism and Relational Mappings
- More challenging relationships:
  - Hierarchical Data
  - Temporal Data

Reducing Redundancy

One big table (the "spreadsheet" view):

Sequence	Description	Species scientific name	Species common name
DIQMTQSPSS…	Ig kappa chain	Homo sapiens	Human
MGDVEKGKKI…	Cytochrome c	Homo sapiens	Human
DTQQAEARSY…	Troponin C	Mus musculus	Mouse
AYVINDSCIA…	Ferrodoxin	Mus musculus	Mouse
GNAAAAKKGS…	Protein kinase C	Mus spretus	Mouse

Consider big table as a join from tables of smaller degree:

Sequence	Description	Species scientific name
DIQMTQSPSS…	Ig kappa chain	Homo sapiens
MGDVEKGKKI…	Cytochrome c	Homo sapiens
DTQQAEARSY…	Troponin C	Mus musculus
AYVINDSCIA…	Ferrodoxin	Mus musculus
GNAAAAKKGS…	Protein kinase C	Mus spretus

Species scientific name	Species common name
Homo sapiens	Human
Mus musculus	Mouse
Mus spretus	Mouse

Normalization

- Aim: avoid redundancy, make data manipulation "atomic"
- Method: identify functional dependencies (scientific name => common name), and group them together such that no two determinants (candidate keys) exist in the same *tuple*.
- "well normalized": A *tuple* consists of a primary key to provide identification and zero or more mutually independent attributes that describe the entity in some way.

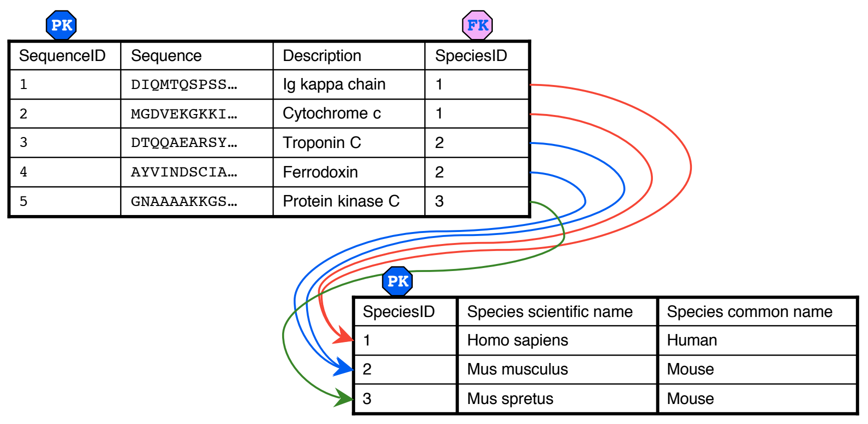
Designing Relational Databases

Primary and Foreign Keys

- Scientific name guaranteed to be unique for each organism => good primary key; sequence table uses scientific name as foreign key into species name table.
- Problem: updates made to primary key values must also be made to foreign keys
- Solution: surrogate primary keys; numeric identifiers or otherwise encoded accession numbers; read-only!
- Foreign Keys provide links between tables: species_id is a Primary Key PK in the species table and a Foreign Key FK in the sequence table.

Designing Relational Databases

Normalization via Surrogate PKs



SequenceID	Sequence	Description	SpeciesID
1	DIQMTQSPSS…	Ig kappa chain	1
2	MGDVEKGKKI…	Cytochrome c	1
3	DTQQAEARSY…	Troponin C	2
4	AYVINDSCIA…	Ferrodoxin	2
5	GNAAAAKKGS…	Protein kinase C	3

SpeciesID	Species scientific name	Species common name
1	Homo sapiens	Human
2	Mus musculus	Mouse
3	Mus spretus	Mouse

Getting back the “spreadsheet” view

- Use SQL to apply the relational algebra:

```
SELECT sequence, description, scientific_name,
  common_name
FROM proteins JOIN species USING (species_id)
```

- SQL queries more powerful than a single spreadsheet: easily obtain different views of the same data.

Simple Sequence Database

- Design a database structure to “hold” NCBI’s non-redundant protein database “nr”
- One table, two fields: description line, and protein sequence.
- Primary key for sequences? Auto-numbered surrogate key.

prot_id	descr	seq
1	gi|121735|sp|P09488|GTM1_HUMAN Glutathione S-transferase ...	MPMIL...
2	gi|232204|sp|P28161|GTM2_HUMAN Glutathione S-transferase ...	MPMTL...
...	...	...

One Protein Sequence; Many Names

- One protein has 1 or more "descriptions"

```
gi|11428198|ref|XP_002155.1| (XM_002155) glutathione S-transferase M1
gi|121735|sp|P09488|GTM1_HUMAN Glutathione S-transferase Mu 1 (GSTM1-1)
gi|87551|pir||S01719 glutathione transferase (EC 2.5.1.18) class mu
gi|31924|emb|CAA30821.1| (X08020) glutathione S-transferase (AA 1-218)
```

- First try: repeat the protein for each description:

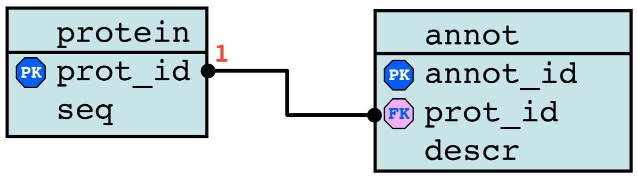
prot_id	descr	seq
1	gi|11428198|ref|XP_002155.1| (XM_002155) glutathione S-tr...	MPMIL...
2	gi|121735|sp|P09488|GTM1_HUMAN Glutathione S-transferase ...	MPMIL...
3	gi|87551|pir||S01719 glutathione transferase (EC 2.5.1.18...	MPMIL...
4	gi|31924|emb|CAA30821.1| (X08020) glutathione S-transfera...	MPMIL...
...	...	...

Entities and Relationships

- Our table is not well-normalized; protein sequences are redundant.
- How do we decide what to split out?
- Analyzing mathematical functional dependencies is too hard; enter the *Entity-Relationship* semantic model.
- Goal: try to identify distinct "*Entities*" present within the data, and try to imagine all allowable "*Relationships*" between them (regardless of whether you have examples in your data yet).

E/R analysis of the database

- Entities? *proteins* and *descriptions* or, more generally, *annotations* (abbrev: *annot*)
- Relationships?
  - 1 *protein* can have many *annotations*;
  - 1 *annotation* applies to only 1 *protein*
  - "One-to-Many" relationship
- Two tables (*protein*, *annot*), with foreign keys in the "many" table (*annot*) pointing to the primary key of the "one" table (*protein*).



Richer Annotations

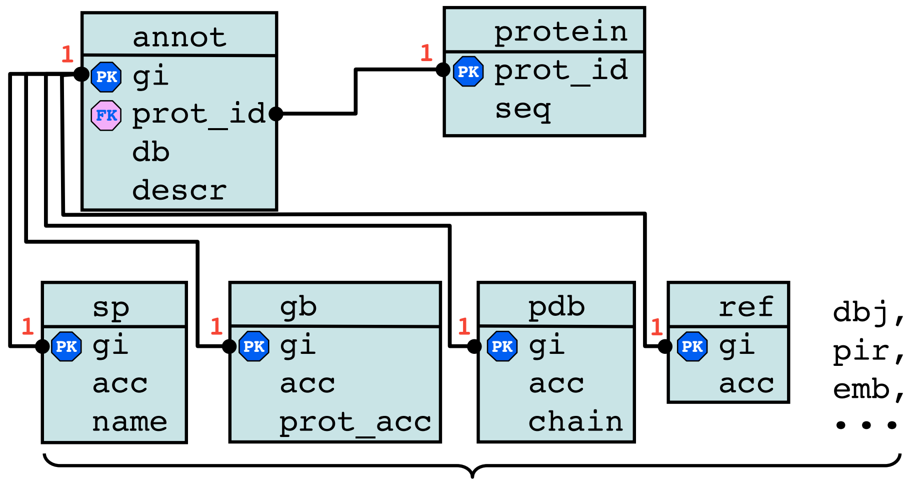
- nr annotations have useful embedded information (multi-valued, in a way):
  - NCBI gi number
  - external database source info (including accession and other identifiers for cross-referencing)
  - textual description
- First try: break these out into their own attributes ("gi" and "dbxref") in the annotation table:

annot_id	prot_id	gi	dbxref	descr
1	1	11428198	ref|XP_002155.1| (XM_002155)	glutathione S-tran...
2	1	121735	sp|P09488|GTM1_HUMAN	Glutathione S-tran...
3	1	87551	pir||S01719	glutathione transf...
4	1	31924	emb|CAA30821.1| (X08020)	glutathione S-tran...
...	...	...	...	...

A better structure

- “gi” looks like a good, natural, “read-only” primary key; dispense with surrogate PK “annot_id”.
- “dbxref” is multi-valued; with different sets of non-overlapping attributes between them, e.g. PDB (accession, chain), SP (accession, name) and EMBL (DNA accession, protein accession). Each distinct attribute requires its own column; many rows remain empty (NULL) in those columns.
- First solution: New “entities” for every type of database cross reference; “One-to-One” relationship, keyed off “gi”.
- Advantage: New database cross references (with new, distinct attributes) can later be added to the database, without adding new columns to existing data

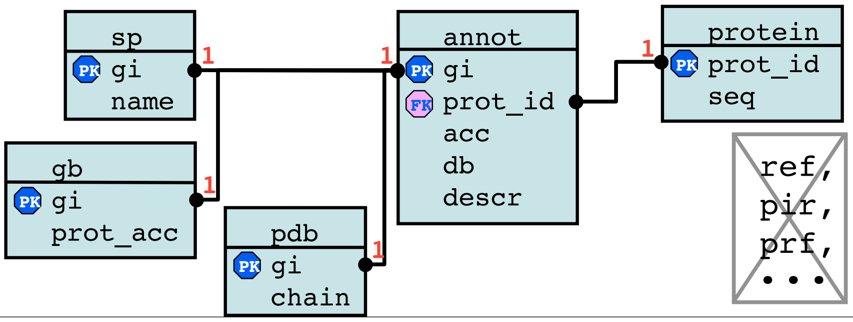
E/R Diagram with dbxref entities



One-To-One Relationships

Sorta the same, sorta different …

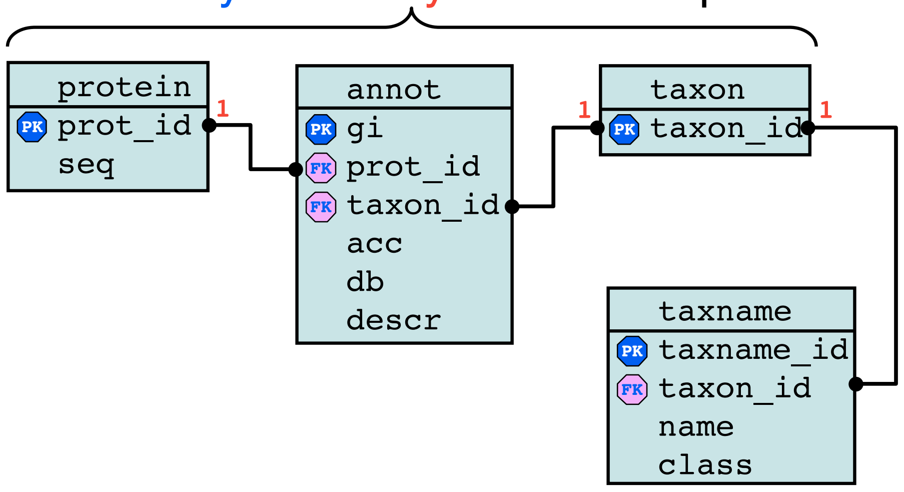
- The dbxref is a "polymorphic" datatype: the same entities in general, but slightly different attributes and semantics
- **Filtered** mapping: one large table, columns for each attribute (with many rows containing NULL values)
- **Horizontal** mapping: split each subtype into many tables, repeating the common attributes (as we did previously)
- **Vertical** mapping: split out uncommon attributes: one "superentity" and as many "subentity" tables as necessary for unique attributes
- Real life considerations (software, complexity) dictate choice



Adding Species

- Add species data to sequences using NCBI's Taxonomy database (provides taxonomy names and gi-to-taxon data)
- "One-to-Many": one species (*taxon*) may have multiple gi's; one gi has only one *taxon*; also, one *taxon* may have multiple names (but only one where `class = "scientific name"`
- Foreign key in annotations (many) table pointing to PK of taxonomy (one) table.
- Relationship between species and sequences is "Many-to-Many", which always requires an intermediate table between the two relations (in this case, the "annot" table serves).

Many-To-Many Relationship



Rules for adding tables:

Is it an "entity" or an "attribute"?

1. If "entity" relationship is 1-to-1 (gi annotation), use one table (unless the entity is polymorphic)
2. If relationship is 1-to-many (1-sequence multiple annotations), use 2 tables, with PK of 1-entity as the FK of the many-entity
3. If relationship is many-to-many (sequences species), use 3 tables; 1 for each "entity" and 1 more (FK1,FK2) for mapping the many-to-many relationship

Hierarchical Data

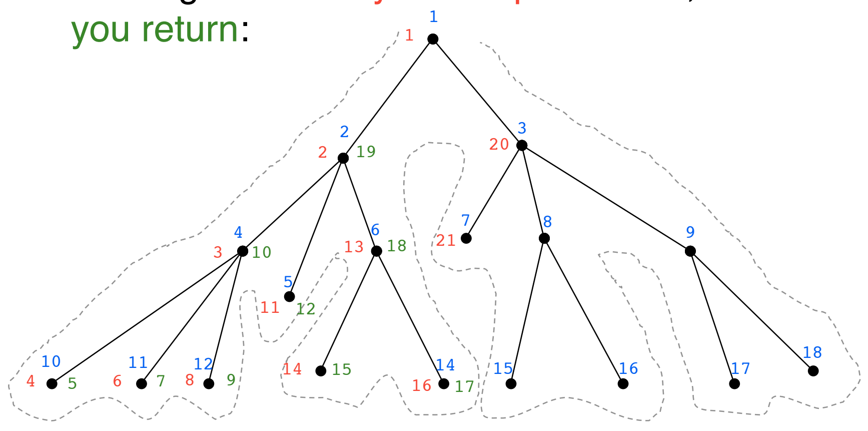
- Parent-child relationships, trees and graphs (between same entity type) - e.g. NCBI Taxonomy, gene ontologies, SCOP classifications, etc.
- Adjacency List model: every *tuple* contains a FK attribute pointing to the PK of the parent; root(s) have NULL FK:

taxon_id	parent_id	name
1	NULL	root
131567	1	cellular organisms
2	131567	Bacteria
2157	131567	Archaea
2759	131567	Eukaryota
1224	2	Proteobacteria

- Requires recursion to select subtrees

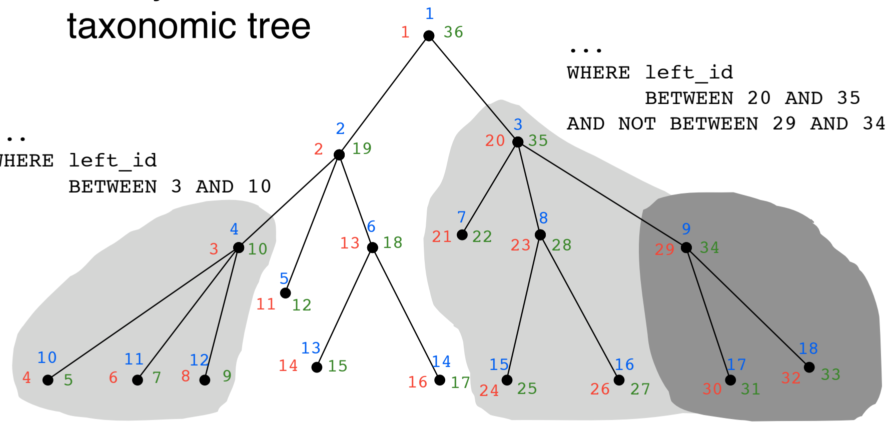
Nested-list representation of hierarchies

- Perform a “depth-first” walk around the tree, labeling nodes as you first pass them, and as you return:



Nested-list representation of hierarchies

- “left_id”, “right_id” attributes provide one-step facility to select entire subsets of the taxonomic tree



```
...
WHERE left_id
      BETWEEN 3 AND 10
```

```
...
WHERE left_id
      BETWEEN 20 AND 35
AND NOT BETWEEN 29 AND 34
```

Temporal Data

- Temporal data (interval-valued) vs. Snapshots (timestamps)
- Single attribute timestamps require difficult paired inter-tuple criteria to select time-specific tuples, and require large amounts of storage:

```
SELECT * FROM annot JOIN history USING (gi)
WHERE entrydate = (SELECT MAX(entrydate)
                   FROM annot
                   WHERE entrydate < ‘2002-01-01’)
```

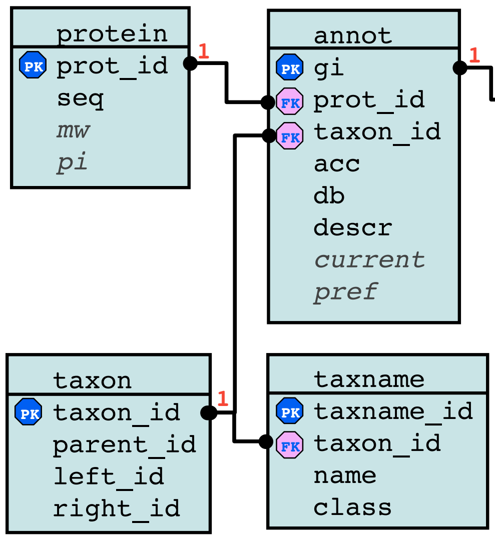
- (begin,end) intervals allow intra-tuple criteria to specify time-specific tuples:

```
SELECT annot.*
FROM annot JOIN history USING (gi)
WHERE (begindate < ‘2002-01-01’ AND enddate IS NULL)
   OR ‘2002-01-01’ BETWEEN begindate AND enddate
```

- Native interval datatypes and operations (EXTEND, DURING, COALESCE, UNFOLD) unavailable in most database products

Designing Relational Databases

seqdb Entity Relationship Diagram



Plus other relations with annot:
sp, pir, pdb,
[gb, emb, dbj],
go, go2gi,
goisa, gopartof,
pfam, SPpfam,
prosite, SPprosite,
ipro, SPipro, ipro2go,
smart, SPsmart

Designing Relational Databases

Queries on "SEQDB"

```
mysql> SELECT count(*)
    -> FROM protein JOIN annot USING (prot_id);
```

```
+-----------+
|  1986207 |
+-----------+
```

```
mysql> SELECT count(*) FROM protein;
```

```
+-----------+
|  1066845 |
+-----------+
```

```
mysql> SELECT annot.gi, annot.db, annot.descr, mid(protein.seq,1,20)
    -> FROM protein JOIN annot USING (prot_id)
    -> GROUP BY protein.id;
```

```
 +----------+-----+-------------------------------------------+-----------------------+
 | gi       | db  | descr                                     | mid(protein.seq,1,20) |
 +----------+-----+-------------------------------------------+-----------------------+
 |  7228451 | dbj | EST AU055734(S20025) corresponds to a regie| MCSYIRYDTPKLFTHVTKTP  |
 |   671595 | emb | rubisco large subunit [Perovskia abrotanoi | MSPQTETKASVGFKAGVKEY  |
 | 10732787 | gb  | homocysteine S-methyltransferase-2 [Zea ma | MVVTAAGSAEEAVRRWVDAA  |
 | 15241446 | ref | (NM_121466) putative protein [Arabidopsis  | MIVISGENVDIAELTDFLCA  |
```

```
mysql> SELECT annot.gi, annot.db, annot.acc,
    ->        sp.name, annot.descr, mid(protein.seq,1,20)
    -> FROM protein JOIN annot USING (prot_id)
    ->              JOIN sp    USING (acc)
```

Tutorial Overview

- Introduction to Relational Databases

- Designing Relational Databases

Using Relational Databases

- Relational Database Fundamentals

- Using Relational Database

Using Relational databases

- Available database products (RDBMS)
- Modes of database interaction and examples with an experimental database.
- Publically available biosequence databases

RDBM Products

- Free:
  - LEAP - DB theory instructional tool
  - MySQL - very fast, widely used, easy to jump into, but limited, nonstandard SQL (JOIN => INNER JOIN)
  - PostgreSQL - full SQL, limited OO, higher learning curve than MySQL
- Commercial:
  - MS Access - GUI interfaces, reporting features
  - MS SQL Server - full SQL, ACID compliant, NT-only
  - Sybase - full SQL, ACID compliant
  - IBM DB2 - full SQL plus hierarchical extensions, ACID compliant
  - Oracle - everything, including the kitchen sink

Manual Database Interaction

- Command line SQL; like using a calculator:

```
mysql> use seqdb;
Database changed
mysql> select count(*) from annot
    -> where current = 1;
+----------+
| count(*) |
+----------+
|  1694330 |
+----------+
1 row in set (19.09 sec)
```

- Batch SQL; keep/edit SQL in file(s), run non-interactively:

```
% mysql -N seqdb < getcounts.sql
1694330
```

Using Relational Databases

Getting a FASTA-formatted database:

```
SELECT  CONCAT( ">gi|", annot.gi, "|sp|", annot.acc, "|", sp.name, " ", annot.descr, "\n",
                protein.seq )
FROM    protein INNER JOIN annot USING (prot_id) INNER JOIN sp USING (acc)
WHERE   annot.current = 1;
```

```
% mysql seqdb -N < swissprot.sql > swissprot.fa
```

```
SELECT  CONCAT( ">gi|", annot.gi, "| ", annot.descr, " [", tn0.name, "]\n",
                protein.seq )
FROM    protein
          INNER JOIN annot USING (prot_id)
          INNER JOIN taxon AS t0 USING (taxon_id)
          INNER JOIN taxon_names AS tn0 USING (taxon_id)
        -- taxonomic inclusion criteria joins:
          INNER JOIN taxon AS t1 ON t0.left_id BETWEEN t1.left_id AND t1.right_id
          INNER JOIN taxon_name AS tn1 ON t1.id = tn1.taxon_id
        -- taxonomic exclusion criteria joins; comment out if no exclusions:
          INNER JOIN taxon AS t2 ON t0.left_id NOT BETWEEN t2.left_id AND t2.right_id
          INNER JOIN taxon_name AS tn2 ON t2.id = tn2.taxon_id
WHERE   1 -- dummy where statement so that things line up nicely below ;)
        -- taxonomic inclusion criteria:
        AND tn1.name  = 'Metazoa'
        AND tn1.class = 'scientific name'
        -- taxonomic exclusion criteria; comment out if no exclusions to be made:
        AND     tn2.name  = 'Drosophila'
        AND     tn2.class = 'scientific name'
-- optional limit statement - useful when debugging, comment out when ready
LIMIT   10
```

```
% mysql seqdb -N < metazoa-not-fruitfly.sql > metazoa-not-fruitfly.fa
```

Using Relational Databases

Can we recreate the “nr” flatfile using MySQL?

```
SELECT protein.id, annot.gi, annot.acc, annot.db, annot.descr, protein.seq
       sp.name, pdb.chain, gb.prot_acc, emb.prot_acc, [...],
FROM   protein INNER JOIN annot USING (prot_id)
       LEFT JOIN sp USING (acc)
       LEFT JOIN pdb USING (acc)
       LEFT JOIN gb USING (acc)
       [...]
WHERE  annot.current = 1
ORDER BY protein.id ASC, annot.gi DESC
```

Generate “spreadsheet view” of all fields; many null values

```
 | id | gi     | acc         | db  | name       | chain | gb.prot_acc | emb.prot_acc | ...
 | 1  | 121735 | P09488      | sp  | GTM1_HUMAN | NULL  | NULL        | NULL         | ...
 | 1  | 31924  | CAA30821.1  | emb | NULL       | NULL  | NULL        | X08020       | ...
 | 2  | 232204 | P28161      | sp  | GTM2_HUMAN | NULL  | NULL        | NULL         | ...
```

```
% mysql -N < regenerate_nr.sql | regenerate_nr.pl > nr.fa
```

```
#!usr/bin/perl -w
my @fields = qw(id gi acc db descr seq name chain gbacc embacc [...]);
my %rowdata;
while (<>) {
    @rowdata{@fields} = split(“\t”, $_, scalar @fields);
    if ($rowdata{db} eq ‘sp’) {
        print “gi|$rowdata{gi}|$rowdata{db}|$rowdata{acc}|$rowdata{name} [...]”;
    } elsif {
        [...]
    }
    [...]
}
```

[… logic to put together all rows of each unique protein sequence …]

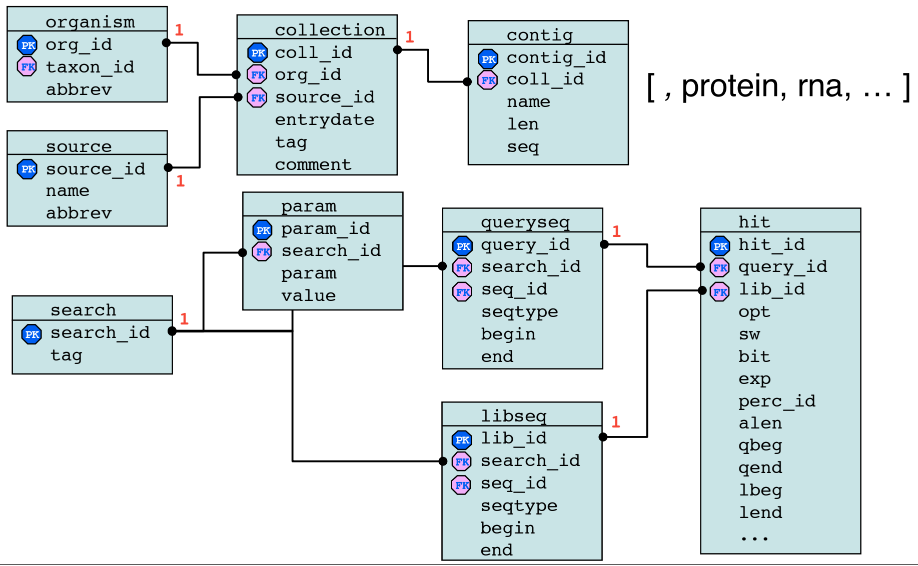
Using Relational Databases

A database for experimental results: EGADS

- A more complicated sequence database:
  - Sequences from bacterial genomes, “proteomes”, and “rnaomes”
  - mappings (ORFs) between the entities (no introns).
- Results from sequence similarity searches between collections of database sequences.
- Sequence analyses (codon bias, dinucleotide frequencies, etc.)
- Evolutionary analyses (clusters and trees).

Designing Relational Databases

egads Entity Relationship Diagram



Creating the Database in MySQL

- DDL SQL kept in schema.sql file:

```
[...]
CREATE TABLE collection (
  coll_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  org_id INT UNSIGNED NOT NULL FOREIGN KEY REFERENCES organism(org_id),
  source_id INT UNSIGNED NOT NULL FOREIGN KEY REFERENCES source(source_id),
  entry_date DATE,
  tag CHAR(20) NOT NULL DEFAULT '',
  comment TEXT
);

CREATE TABLE contig (
  contig_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  coll_id INT UNSIGNED NOT NULL FOREIGN KEY REFERENCES collection(coll_id),
  name TEXT DEFAULT NULL,
  len INT UNSIGNED NOT NULL DEFAULT 0,
  seq LONGTEXT DEFAULT NULL
);
[...]
```

- Run in command line “batch” mode:

```
% mysql egads < schema.sql
```

Programming with SQL:

- Embedded SQL: run SQL statements from within another program, using the data directly
  – Data collection, management and extraction using Perl and the Perl DBI
  – Extending existing C programs (e.g. FASTA) to become database “aware”
  – Statistical data analysis using R and RMySQL

Putting sequences into EGADS

- Perl DBI-based programs: `addgenome`, `addproteome`, and others:

```
#!/usr/bin/perl
use DBI;
my $dbh = DBI->connect(“dbi:mysql:egads”, “myusername”, “pw”);
my $sth = $dbh->prepare(<<SQL);
INSERT INTO contig (seq, name, len) VALUES (?, ?, ?)
SQL

# [parse input FASTA-formatted file and build array of @sequences]

foreach my $seq (@sequences) {
    # [extract $seq and $name, calculate $len]
    $sth->execute($seq, $name, $len);
    my $id = $sth->{mysql_insertid};
    # etc.
}
$dbh->disconnect();
```

Running Similarity Searches

- SQL query against EGADS database (`proteome.sql`, `genome.sql`)

```
SELECT contig_id, CONCAT("CONTIG_ID:", contig_id, " ", name), seq
FROM   contig INNER JOIN collection USING (coll_id)
WHERE  collection.tag = 'YPE';
```

- FASTA extended to use SQL directly
  (using the C library `libmysql`):

```
% tfastx34 -q “proteome.sql 16” “genome.sql 16”
```

- Or using BLAST:

```
% mysql -N < proteome.sql | perl -pe ‘s/^\S+\s+/>/;s/\S+$/\n$&/’ > proteome.fa
% mysql -N < genome.sql | perl -pe ‘s/^\S+\s+/>/;s/\S+$/\n$&/’ > genome.fa
% formatdb -p T -i proteome.fa; formatdb -p F -i genome.fa
% blastall -p T tblastx -i proteome.fa -d genome.fa
```

Using Relational Databases

Loading/Retrieving Search Results

```
Query library YPE-proteome.sql vs YPE-genome.sql library
searching YPE-genome.sql 16 library
  1>>>PROT_ID:40537 putative flavoprotein 146 aa
TFASTX (3.43 Dec 2001) function [optimized, BL62 matrix (o=11:-4:-1)xS] ktup: 2
 join: 36, opt: 32, open/ext: -7/-1 shift: -20, width:  16
The best scores are: opt bits E(1)      %id    sw  an0  ax0       an1     ax1
CONTIG_ID:16593      749  277 4.1e-78  1.000  749    1  146       711     274
CONTIG_ID:16593      245   96 6.8e-24  0.382  254    1  146   1179347 1179789
CONTIG_ID:16593      166   68 2.2e-15  0.319  166    4  138   3761512 3761111
CONTIG_ID:16593       78   37 6.3e-06  0.242  78    13  132   4291766 4291398
...
```

```
mysql> select lib.lib_id, opt, bits, exp, percid, sw, qbeg, qend, lbeg, lend
    -> from hit join search using (search_id)
    ->          join query using (query_id)
    ->          join lib using (lib_id)
    -> where search.tag = "YPE-vs-YPE-BL62"
    -> and query.seq_id = 40537
    -> order by exp asc
    -> limit 4;
+--------+-----+------+---------+--------+-----+------+------+---------+---------+
| lib_id | opt | bit  | exp     | percid | sw  | qbeg | qend | lbeg    | lend    |
+--------+-----+------+---------+--------+-----+------+------+---------+---------+
|  16593 | 749 |  277 | 4.1e-78 |      1 | 749 |    1 |  146 |     711 |     274 |
|  16593 | 245 |   96 | 6.8e-24 |  0.438 | 254 |    1 |  146 | 1179346 | 1179789 |
|  16593 | 166 |   68 | 2.2e-15 |  0.336 | 166 |    4 |  138 | 3761512 | 3761111 |
|  16593 |  78 |   37 | 6.3e-06 |  0.252 |  78 |   13 |  132 | 4291766 | 4291398 |
+--------+-----+------+---------+--------+-----+------+------+---------+---------+
```

Using Relational Databases

Analyzing Data from EGADS

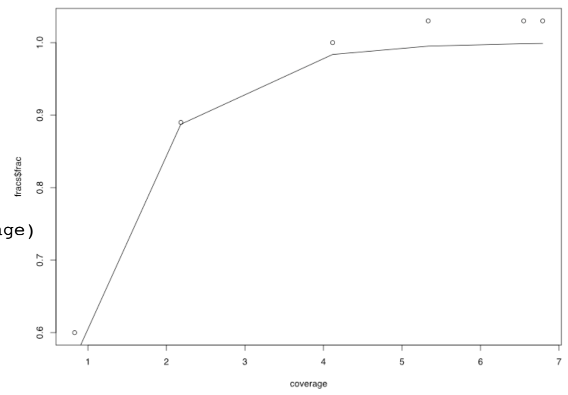
- R - a free statistical programming environment using the S programming language
- Directly access the database from within R (RMySQL, RPgSQL, RODBC)
- Using R functions from within the database (R as a PostgreSQL Procedural Language - the OmegaHat project for statistical computing)
- Using R from within Perl (+ DBI): the best of all worlds - procedural language, relational data and statistical programming.

Using Relational Databases

R with EGADS: Lander/Waterman coverage

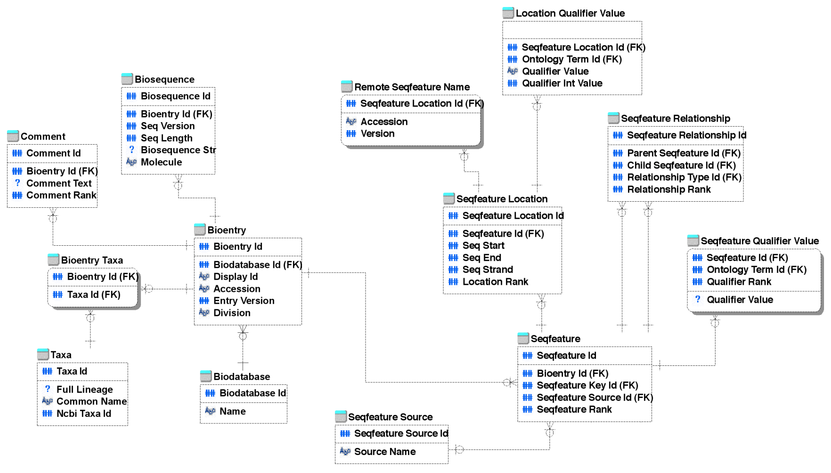
```
library("RMySQL")
dbi <- dbManager("MySQL")
dbh <- dbConnect(dbi, group = "egads")
sth <- dbExecStatement(dbh,
  statement = paste(
"SELECT SUM(contig.len) / 4857432 AS frac",
"FROM   contig INNER JOIN collection USING (coll_id)",
"WHERE  collection.tag LIKE 'STM-%x'",
"GROUP BY contig.coll_id",
"ORDER BY collection.tag",
  sep = " ")
)
fracs <- fetch(sth, n = -1)
close(sth)
close(dbh)


plot(coverage, fracs$frac)
lines(coverage, 1-exp(-coverage)
```



Using Relational Databases

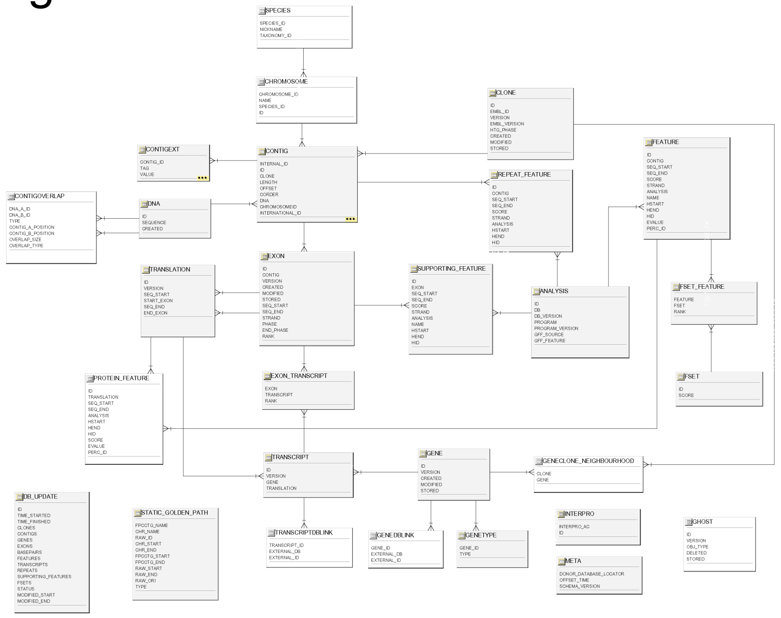
BioSQL - a full-featured biosequence database



http://cvs.bioperl.org/cgi-bin/viewcvs/viewcvs.cgi/*checkout*/biosql-schema/doc/biosql-ERD.pdf?rev=1.2&cvsroot=biosql&content-type=application/pdf

Using Relational Databases

A genome-centric solution: ensembl.org



http://cvsweb.sanger.ac.uk/cgi-bin/cvsweb.cgi/~checkout~/ensembl/sql/ensembl_0_7_4.pdf?rev=1.1&content-type=text/plain&cvsroot=Ensembl

Using Relational Databases

Online Resources

- RDBM Products
  - LEAP: http://leap.sourceforge.net/
  - MySQL: http://www.mysql.com/
  - PostgreSQL: http://www.postgresql.org/
- Relational Biological Databases:
  - Pearson Lab databases (seqdb, egads): ftp://ftp.virginia.edu/fasta/rdb/
  - bioSQL: http://bioteam.net/dag/BioTeam-HOWTO-1-BIOSQL.html
  - OBDA: http://obda.open-bio.org
  - ensembl: http://www.ensembl.org/
- Software Tools:
  - Tangram: http://www.soundobjectlogic.com/tangram/
  - Perl: http://www.perl.com, http://www.perl.org
  - R Statistical Environment: http://www.r-project.org/
  - The OmegaHat Project: http:www.omegahat.org

Final Exam:

Take a Genbank Flat File:

1. What are the entities?
2. What are the attributes?
3. Design a database that captures:
   a. Locus
   b. Accession
   c. Sequence
   d. Species
   e. Authors/ref.
   f. Features

```
LOCUS       GTM1_HUMAN               218 aa            linear   PRI 16-OCT-2001
DEFINITION  Glutathione S-transferase Mu 1 (GSTM1-1) (HB subunit 4) (GTH4)
            (GSTM1A-1A) (GSTM1B-1B) (GST class-Mu 1).
ACCESSION   P09488
VERSION     P09488  GI:121735
DBSOURCE    swissprot: locus GTM1_HUMAN, accession P09488;
            created: Mar 1, 1989.
            xrefs: gi: gi: 31923, gi: gi: 31924, gi: gi: 183668, gi: gi:
            xrefs (non-sequence databases): MIM 138350, InterPro IPR004046,
            InterPro IPR004045, InterPro IPR003081, Pfam PF00043, Pfam PF02798,
            PRINTS PR01267
KEYWORDS    Transferase; Multigene family; Polymorphism; 3D-structure.
SOURCE      human.
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
REFERENCE   2  (residues 1 to 218)
  AUTHORS   Seidegard,J., Vorachek,W.R., Pero,R.W. and Pearson,W.R.
  TITLE     Hereditary differences in the expression of the human glutathione
            transferase active on trans-stilbene oxide are due to a gene deletion
  JOURNAL   Proc. Natl. Acad. Sci. U.S.A. 85 (19), 7293-7297 (1988)
  MEDLINE   89017184
FEATURES             Location/Qualifiers
     source          1..218
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606”
     Protein         1..218
                     /product="Glutathione S-transferase Mu 1"
                     /EC_number="2.5.1.18"
     Region          173
                     /region_name="Variant"
                     /note="K -> N (IN ALLELE B). /FTId=VAR_003617."
ORIGIN
        1 mpmilgywdi rglahairll leytdssyee kkytmgdapd ydrsqwlnek fklgldfpnl
//
```

Further Reading

- *Access Database Design and Programming* (Steven Roman): excellent simple introduction to relational theory, normalization and SQL
- *An Introduction to Database Systems* (C.J. Date): undergraduate CS text
- *Data Modelling Essentials* (Graeme Simsion and Graham Witt): Strategies for E/R modelling of complicated relationships
- *SQL For Smarties* (Joe Celko): Advanced SQL, trees, graphs, time series, etc.
- *MySQL* (Paul DuBois): A beginner's user-manual for installing, administering and using MySQL.
- *Advanced MySQL* (Jeremy Zawodny): not yet published (exp: late 2002), but a more in-depth treatment than the DuBois book.

Glossary

API	applications program interface	
COM	component object model	(MS)
CORBA	common object request broker architecture	
CPAN	Comprehensive Perl Access Network (Perl software modules)	
DDL	data description language	(SQL)
DML	data manipulation language	(SQL)
DOM	domain object model	(WWW)
foreign key	a link from a tuple (row) in one relation (table) to additional information about the entity in another relation. A foreign key in one table is a primary key in the other.	
IDL	interface design language (CORBA)	
GO	Gene Ontology	
inheritance	using the properties of one object to define the properties of another; e.g. a protein_sequence is a sequence	(OO)
JDBC	java database connectivity	(SQL)
middleware	software that provides a standard link (API) between two applications, or other computing resources (BioPerl)	
OO	object oriented	
OORDBM	object oriented relational database manager	
OQL	object query language	
ORB	object request broker (CORBA)	

Perl DBI	a general database interfacein for Perl - middleware	(SQL)
primary key	the unique identifier for each *tuple* (row) in a *relation* (table)	(SQL)
Polymorphism	the different behaviors of an entity; the ability to have different forms	(OO)
RDBMS	relational database management system	(SQL)
RDF	resource description framework - a lightweight ontology for exchanging knowledge	(WWW)
schema	the tables and links (entity relationships) in an database	(SQL)
semantics	the meaning of a term or relationship	
SOAP	simple object access protocol	(WWW)
SQL	structured query language	
syntax	the structure (grammar) of relationships	
UDDI	Universal Description, Discovery and Integration	(WWW)
UML	unified modeling language	(OO)
XML	extensible markup language	
XML database	called XDB sometimes, but not often	
XML schema	an XML specification for writing schemas in XML, not database specific	
XQL	XML query language	

http://www.webopedia.com/