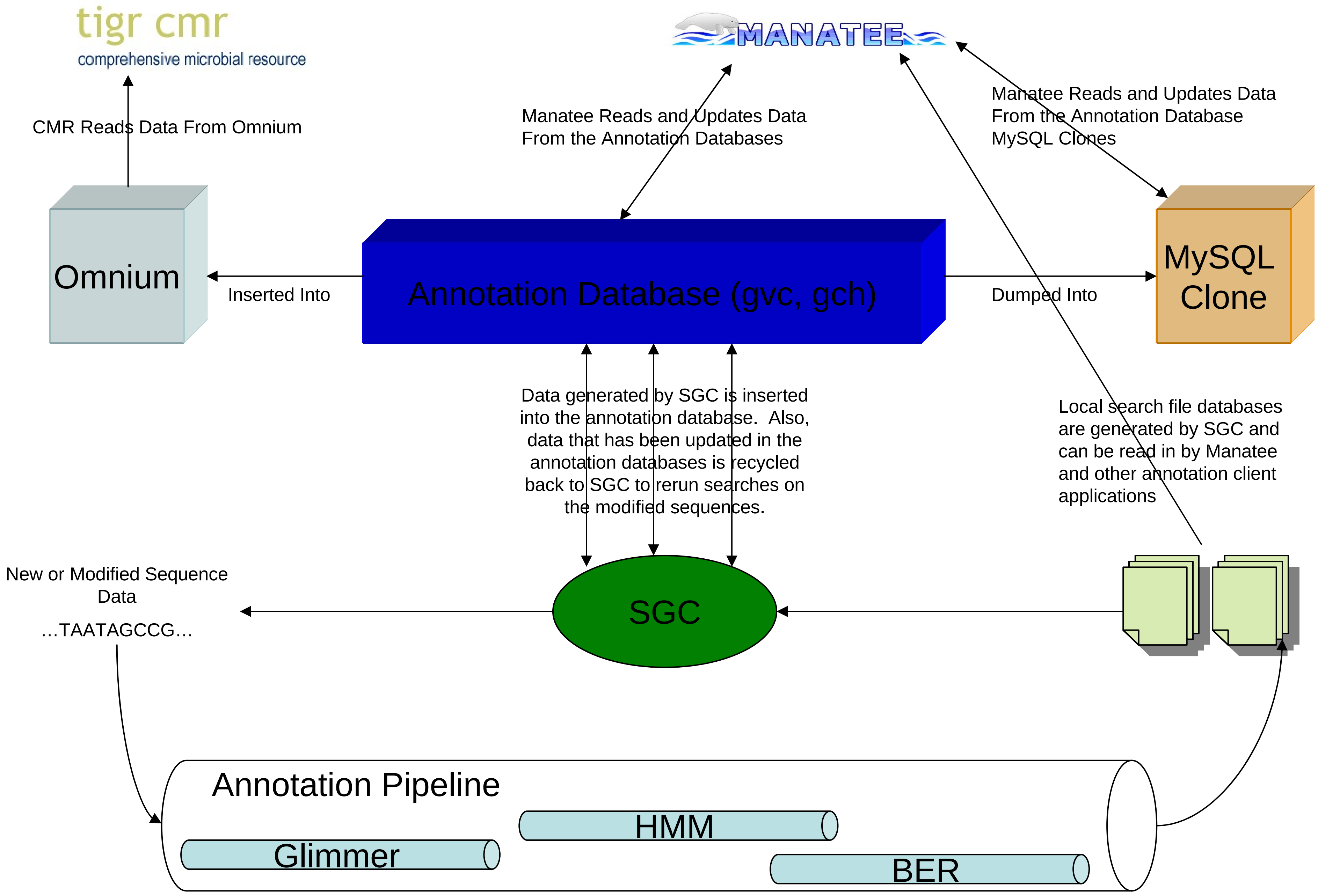
Manatee and the Annotation System Architecture

An In-depth Look Inside Manatee Development and the Annotation Process

Annotation Architecture Overview

- Manatee is only a small part of a network of annotation tools and processes that make up the annotation architecture
- The Small Genome Control (SGC) is the command center for all data management
- SGC controls results from searches from the annotation pipeline and feeds them into the database
- Client scripts like Manatee and the CMR allow manipulation and curation of this data


tigr cmr
comprehensive microbial resource
MANATEE
CMR Reads Data From Omnium
Manatee Reads and Updates Data From the Annotation Databases
Manatee Reads and Updates Data From the Annotation Database MySQL Clones
Omnium
Inserted Into
Annotation Database (gvc, gch)
Dumped Into
MySQL Clone
Data generated by SGC is inserted into the annotation database. Also, data that has been updated in the annotation databases is recycled back to SGC to rerun searches on the modified sequences.
Local search file databases are generated by SGC and can be read in by Manatee and other annotation client applications
New or Modified Sequence Data
…TAATAGCCG…
SGC
Annotation Pipeline
HMM
Glimmer
BER

An Annotation Database Overview

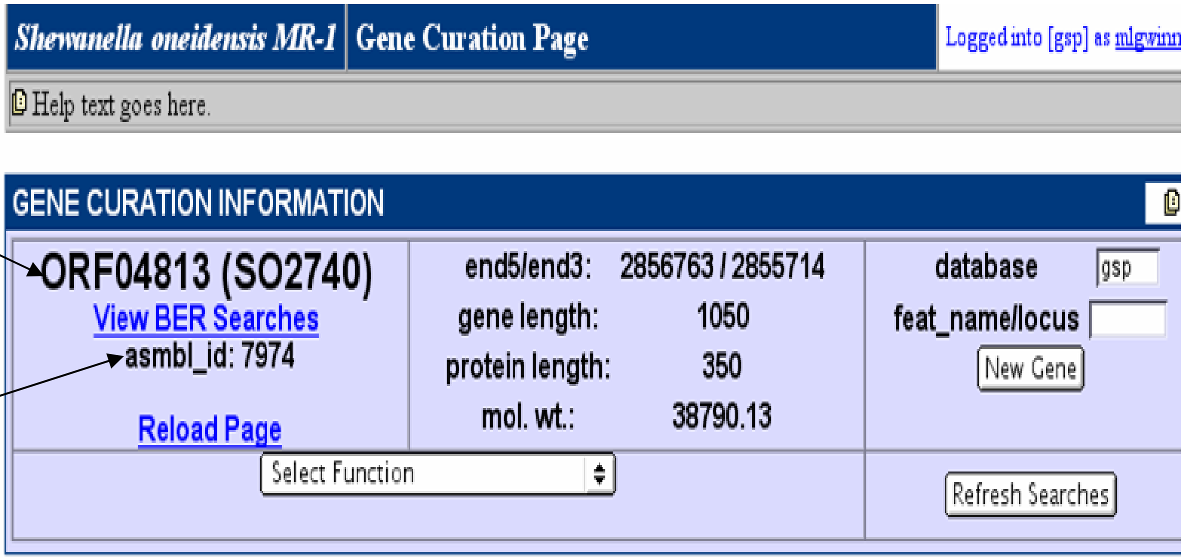
MANATEE

Annotation Database

- Manatee draws data from numerous microbial relational databases whose schemas are identical
- Certain primary identifiers are used throughout the database to identify specific genes and sequences
- See examples below:

feat_name: Primary identifier for a gene

asmbl_id: Primary identifier for a genomic sequence



Shewanella oneidensis MR-1 | Gene Curation Page | Logged into [gsp] as mlgwinn

Help text goes here.

GENE CURATION INFORMATION

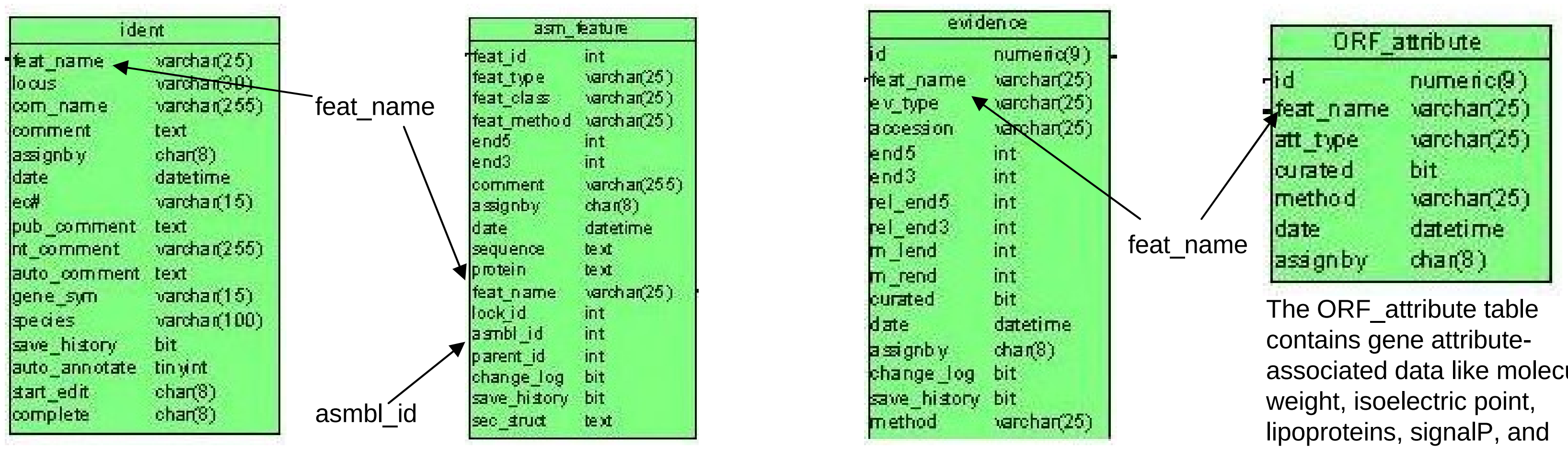
ORF04813 (SO2740)
View BER Searches
asmbl_id: 7974
Reload Page

end5/end3:	2856763 / 2855714
gene length:	1050
protein length:	350
mol. wt.:	38790.13

database: gsp
feat_name/locus:

Annotation Database

- Manatee runs queries that pull data from several tables each of which contain specific groups of data.
- The entire schema can be found at: http://manatee.sourceforge.net/images/prok_annotation_schema.jpg
- Examples of important tables from the annotation database are below:



ident	
feat_name	varchar(25)
locus	varchar(30)
com_name	varchar(255)
comment	text
assignby	char(8)
date	datetime
ec#	varchar(15)
pub_comment	text
nt_comment	varchar(255)
auto_comment	text
gene_sym	varchar(15)
species	varchar(100)
save_history	bit
auto_annotate	tinyint
start_edit	char(8)
complete	char(8)

The ident table contains data related to gene identification. You will find information on the locus, gene name, gene symbols, EC #'s and annotation comments

asm_feature	
feat_id	int
feat_type	varchar(25)
feat_class	varchar(25)
feat_method	varchar(25)
end5	int
end3	int
comment	varchar(255)
assignby	char(8)
date	datetime
sequence	text
protein	text
feat_name	varchar(25)
lock_id	int
asmbl_id	int
parent_id	int
change_log	bit
save_history	bit
sec_struct	text

The asm_feature table contains data related to gene coordinates, mRNA, and protein sequences

evidence	
id	numeric(9)
feat_name	varchar(25)
ev_type	varchar(25)
accession	varchar(25)
end5	int
end3	int
rel_end5	int
rel_end3	int
m_lend	int
m_rend	int
curated	bit
date	datetime
assignby	char(8)
change_log	bit
save_history	bit
method	varchar(25)

The evidence table contains all coordinate based "hit" data like HMMs, BERs, Interpro

ORF_attribute	
id	numeric(9)
feat_name	varchar(25)
att_type	varchar(25)
curated	bit
method	varchar(25)
date	datetime
assignby	char(8)

The ORF_attribute table contains gene attribute-associated data like molecular weight, isoelectric point, lipoproteins, signalP, and transmembrane proteins

Other Required Databases

- EGAD and COMMON are two required supporting databases that contain important information for annotation

- EGAD contains data related to HMM annotation and TIGR role identification

- COMMON contains data for GO annotation and identification

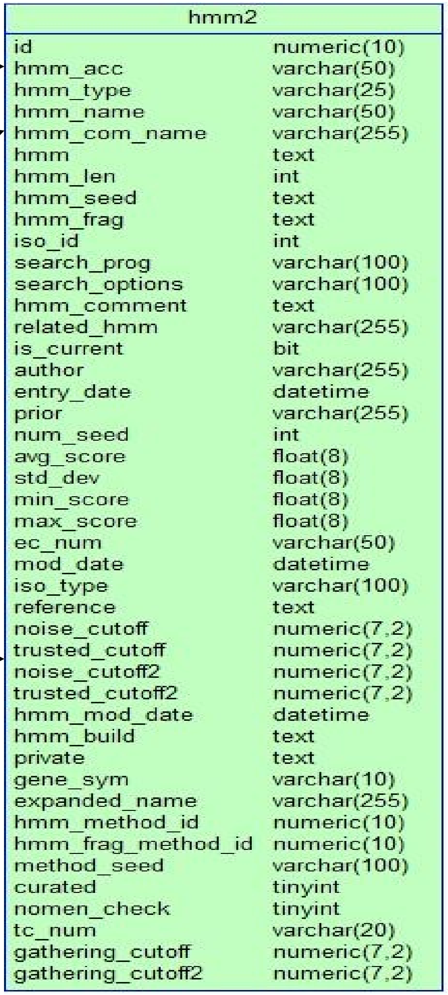
EGAD Tables

- hmm2 contains HMM annotation data like names, cutoffs, and other scores
- roles contains TIGR role data like role names and sub-names

HMM accession

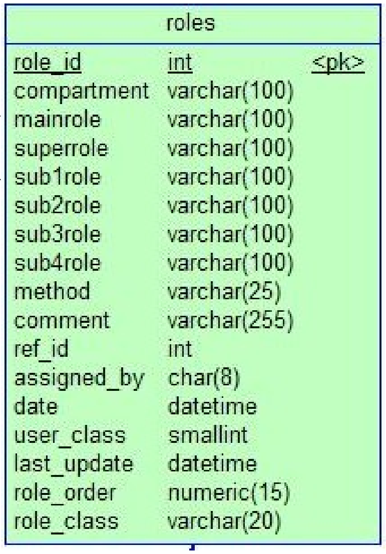
HMM name

Cutoff Scores



hmm2	
id	numeric(10)
hmm_acc	varchar(50)
hmm_type	varchar(25)
hmm_name	varchar(50)
hmm_com_name	varchar(255)
hmm	text
hmm_len	int
hmm_seed	text
hmm_frag	text
iso_id	int
search_prog	varchar(100)
search_options	varchar(100)
hmm_comment	text
related_hmm	varchar(255)
is_current	bit
author	varchar(255)
entry_date	datetime
prior	varchar(255)
num_seed	int
avg_score	float(8)
std_dev	float(8)
min_score	float(8)
max_score	float(8)
ec_num	varchar(50)
mod_date	datetime
iso_type	varchar(100)
reference	text
noise_cutoff	numeric(7,2)
trusted_cutoff	numeric(7,2)
noise_cutoff2	numeric(7,2)
trusted_cutoff2	numeric(7,2)
hmm_mod_date	datetime
hmm_build	text
private	text
gene_sym	varchar(10)
expanded_name	varchar(255)
hmm_method_id	numeric(10)
hmm_frag_method_id	numeric(10)
method_seed	varchar(100)
curated	tinyint
nomen_check	tinyint
tc_num	varchar(20)
gathering_cutoff	numeric(7,2)
gathering_cutoff2	numeric(7,2)

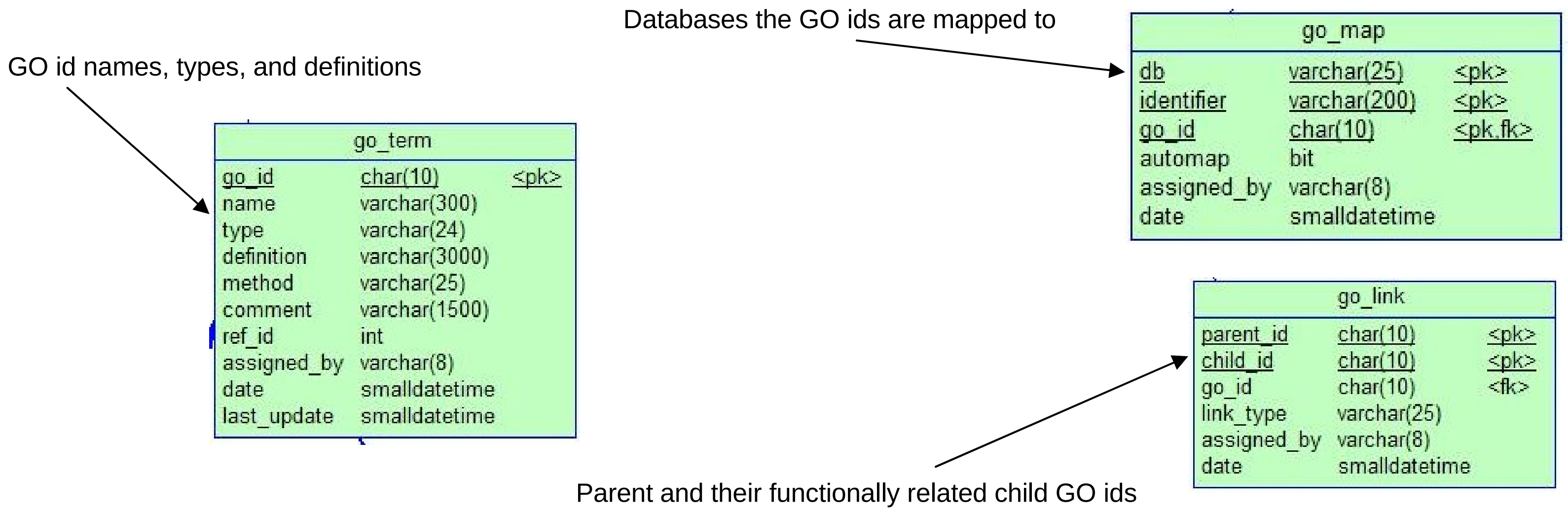
Role names



roles		
role_id	int	<pk>
compartment	varchar(100)	
mainrole	varchar(100)	
superrole	varchar(100)	
sub1role	varchar(100)	
sub2role	varchar(100)	
sub3role	varchar(100)	
sub4role	varchar(100)	
method	varchar(25)	
comment	varchar(255)	
ref_id	int	
assigned_by	char(8)	
date	datetime	
user_class	smallint	
last_update	datetime	
role_order	numeric(15)	
role_class	varchar(20)	

COMMON Tables

- go_term contains GO annotation like GO names, types, and their definitions
- go_map maps GO annotation to other annotation data types and databases like EC numbers and Interpro accessions
- go_link maintains the GO id hierarchy between “parent” GO ids and the parent’s functionally related “child” GO ids



GO id names, types, and definitions

Databases the GO ids are mapped to

Parent and their functionally related child GO ids

go_term

go_id	char(10)	<pk>
name	varchar(300)	
type	varchar(24)	
definition	varchar(3000)	
method	varchar(25)	
comment	varchar(1500)	
ref_id	int	
assigned_by	varchar(8)	
date	smalldatetime	
last_update	smalldatetime	

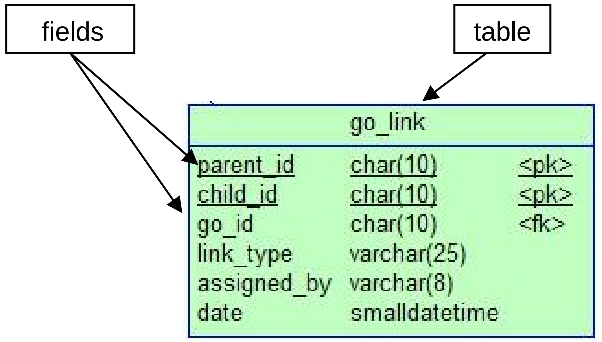
go_map

db	varchar(25)	<pk>
identifier	varchar(200)	<pk>
go_id	char(10)	<pk,fk>
automap	bit	
assigned_by	varchar(8)	
date	smalldatetime	

go_link

parent_id	char(10)	<pk>
child_id	char(10)	<pk>
go_id	char(10)	<fk>
link_type	varchar(25)	
assigned_by	varchar(8)	
date	smalldatetime	

Basic Query Structure



go_link		
parent_id	char(10)	<pk>
child_id	char(10)	<pk>
go_id	char(10)	<fk>
link_type	varchar(25)	
assigned_by	varchar(8)	
date	smalldatetime	

- SELECT [fields name(s) in the table]
- FROM [the table name(s)]
- WHERE [constraints to pull out only the types of data you want]
- AND [additional constrains]
- Final Basic Structure:

SELECT “a field” FROM “a table” WHERE “the field data looks like something” AND “the field data ALSO looks like something else”

Simple Query Examples

Get the com_name, locus, and gene symbol for a particular gene:

```
SELECT feat_name, locus, gene_sym, com_name
FROM ident
WHERE feat_name = “ORF04813”
```

feat_name	locus	gene_sym	com_name
ORF04813	SO2740	bioB	biotin synthase

Get the TIGR role ids for a particular gene:

```
SELECT feat_name, role_id
FROM role_link
WHERE feat_name = “ORF04813”
```

feat_name	role_id
ORF04813	77

Get the GO ids for a particular gene:

```
SELECT feat_name, go_id
FROM go_role_link
WHERE feat_name = “ORF04813”
```

feat_name	go_id
ORF04813	GO:0004076
ORF04813	GO:0009102

Complex Query Examples

Get the TIGR role ids for a particular gene:

```
SELECT feat_name, role_id
FROM role_link
WHERE feat_name = "ORF04813"
```

feat_name	role_id
ORF04813	77

Now get the role names as well

```
SELECT r.feat_name, r.role_id, e.mainrole, e.sub1role
FROM role_link r, egad..roles e
WHERE r.feat_name = "ORF04813"
AND r.role_id = e.role_id
```

feat_name	role_id	mainrole	sub1role
ORF04813	77	Biosyntesis of cofactors, prosthetic groups, and carriers	Biotin

Get the GO ids for a particular gene:

```
SELECT feat_name, go_id
FROM go_role_link
WHERE feat_name = "ORF04813"
```

feat_name	go_id
ORF04813	GO:0004076
ORF04813	GO:0009102

Now get the GO annotation as well

```
SELECT g.feat_name, g.go_id, t.type, t.name
FROM go_role_link g, common..go_term t
WHERE g.feat_name = "ORF04813"
AND g.go_id = t.go_id
```

feat_name	go_id	type	name
ORF04813	GO:0004076	function	biotin synthase activity
ORF04813	GO:0009102	process	biotin biosynthesis

“Really” Complex Query Examples

- These are very complex queries that use 3 or more tables to get information

Get all data for each HMM belonging to a particular gene including all coordinates, cutoff scores, HMM accessions/names, and other scores. **Note** that this query makes use of tables in the egad and common databases. These databases are required for Manatee to run properly:

```
SELECT e.id, e.accession, e.ev_type, s.score, t.score_type,
e.curated, e.end5, e.end3, e.rel_end5, e.rel_end3,
h.trusted_cutoff, h.noise_cutoff, e.assignby, e.date,
e.m_lend, e.m_rend, h.hmm_com_name, h.iso_type,
h.hmm_len, h.ec_num, h.gene_sym, h.trusted_cutoff2,
h.noise_cutoff2, h.gathering_cutoff, h.gathering_cutoff2
FROM evidence e, feat_score s, egad..hmm2 h, common..score_type t
WHERE e.feat_name = "ORF04813"
AND h.is_current = 1
AND e.id = s.input_id
AND t.input_type = "HMM2"
AND t.id = s.score_id
AND e.accession = h.hmm_acc
```

The **score_type table** in the common database contains a controlled vocabulary describing all the scores found in the feat_score table

The **feat_score table** contains multiple scores for certain types of hits like HMMs

The **stan table** is used to pull out the latest genomic sequence identifier (the **asmbl_id**)

Get all HMM genomic coordinates and relative gene coordinates for a particular gene:

```
SELECT e.end5, e.end3, e.rel_end5, e.rel_end3
FROM evidence e, asm_feature f, stan s
WHERE e.feat_name = "ORF04813"
AND e.ev_type = "HMM2“
AND e.feat_name = f.feat_name
AND f.asmbl_id = s.asmbl_id
AND s.iscurrent = 1
```

Get all genes and their respective annotation for a particular **role_id** (not feat_name). In this example, we use role_id 91 – Cell Envelope, Surface Structures:

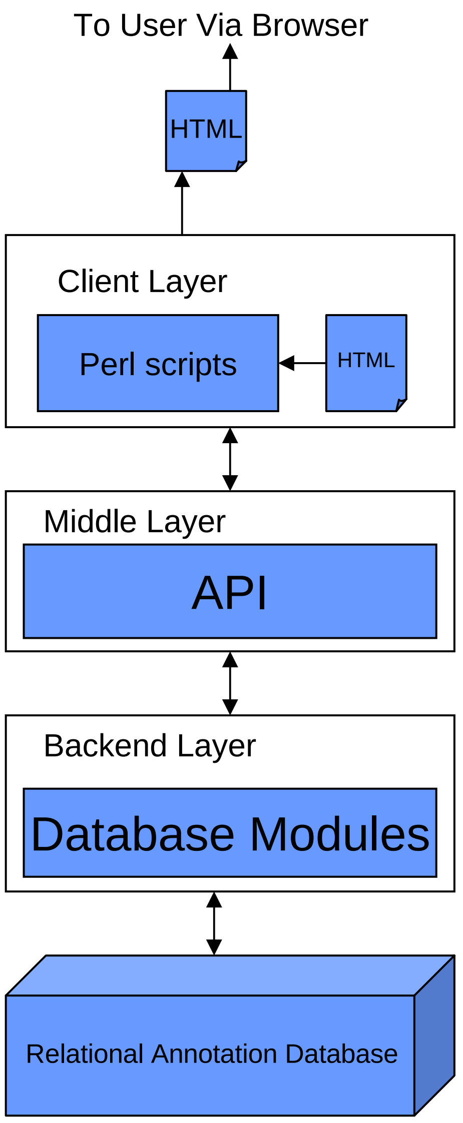
```
SELECT i.feat_name, i.locus, i.com_name, i.gene_sym,
i.ec#, a.asmbl_id, a.end5, a.end3
FROM ident i, role_link r, asm_feature a, stan s
WHERE r.role_id = 91
AND r.feat_name = a.feat_name
AND a.feat_name = i.feat_name
AND a.asmbl_id = s.asmbl_id
AND s.iscurrent = 1
```

A Software Architecture Overview

Manatee Software Architecture

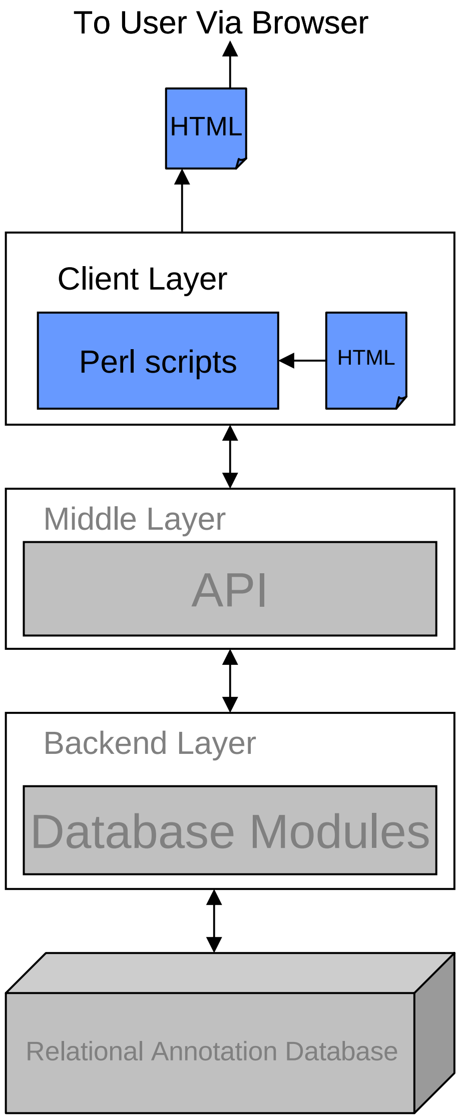
- The Manatee software is a Perl/CGI based application which is built upon a 3-tier architecture
- Portable between several different database vendors (Sybase, MySQL) and operating systems (Redhat/Fedora/Mandrake Linux, Solaris)
- 3-tier, API-based structure allows the software to be expandable and reusable by other future annotation related software products
- Project-specific customization possible through clever use of an HTML templating system

3-Tier Architecture Overview



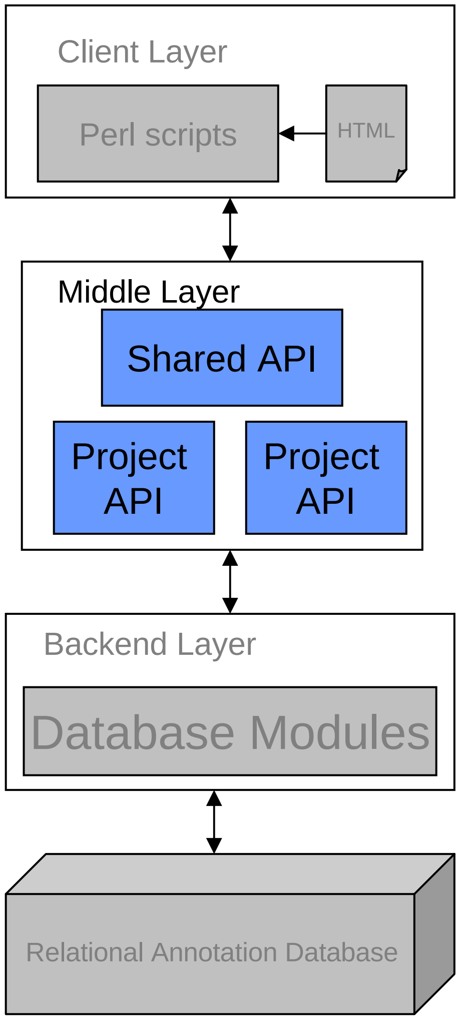
- A software application is “3-Tier” if it contains 3 standard layers each of which are independent from each other
- Each layer can be independently modified and new modules can be introduced to make use of new databases or changes in user requirements
- Client Layers are responsible for taking data and formatting it in some sort of useful user interface
- Middle Layers house a standard API, a group of “functions” that allow the Client Layer to communicate with the Backend Layer
- Backend Layers contain the queries that the Middle Layer requests and retrieve the data

The Client Layer



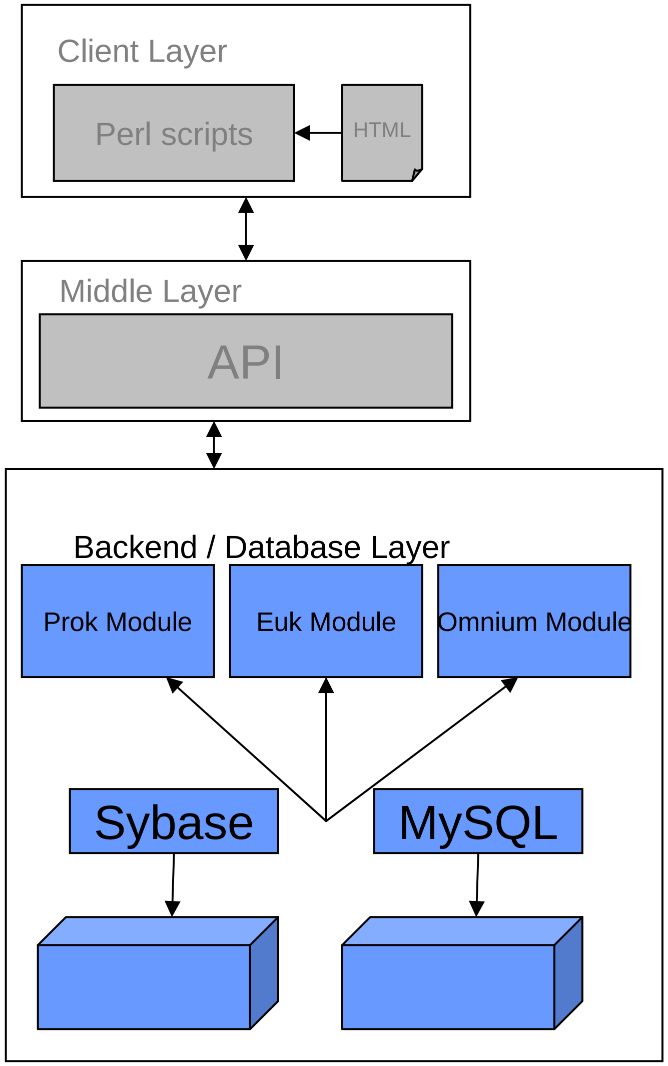
- Made up of Perl CGI scripts
- HTML Templating System allows for project-specific user interfaces
- Communicates with the Middle Layer to retrieve data from a specified database and vendor
- Output sent to web browser for viewing

The Middle Layer (API)



- Interface for clients to the Backend Layer
- Light-weight API is independent of user interfaces and databases
- Simple, static, and contains a controlled vocabulary
- Extensible API's allow for more than one type of project to use a shared API
- For example, the CMR and Manatee can share the API which will in turn call a different set of queries from the Backend layer

The Backend Layer (Database)



- SQL resides in this layer
- Multiple schema support (prok, euk, and omnium schemas)
- Multiple database vendor support (Sybase, MySQL, Postgres)
- Flexibility to use future schemas and vendors in the future (Chado, Oracle?)

Open Source Initiative

- Manatee and the Annotation Engine project are part of our Open Source Initiative
- Goal is to provide high quality software and services to the genomic community
- External involvement in development and feedback is strongly encouraged
- Creation of centralized portals that provide a knowledge base and allow for access and support for these software services
- Manatee’s Portal: http://manatee.sourceforge.net
- Community feedback drives development!

Manatee Example Review

- Manatee is an example of all that we’ve discussed so far
- Works off of our annotation databases
- Software is built on the 3-tier architecture
    - Web based user interface representing the client
    - Client “asks” to the API middle layer for certain information
    - API in turn asks the database layer to “go get it”
- Open source project with emphasis on community feedback and contributions to development

Manatee External Overview

- Installed on a Linux or Solaris machine at your location
- Other required packages will be needed: MySQL, Apache, Perl modules, and required databases
- Annotation Engine -> MySQL dump of annotation database -> Placed in private FTP area -> Downloaded by you!
- See the installation instructions for all the details on how to do this: http://manatee.sourceforge.net/installation.shtml
- I will give an overview of Manatee’s website after the presentation

Future Software and Services

- An automated flat file refreshing service
- Automatic database updates (common database, xml files, etc)
- Sybil – Comparative Genomics Software
  - Will allow data mining of a database containing comparative genomics data
  - Software will be available to the external community much like Manatee is currently
- Chado – Generic Model Organism Database
  - Chado is an open source modular schema that we are currently testing
  - Development is underway to migrate the annotation databases into this new schema (will take a while, though) ☺

MANATEE

Funding of Future Developments

- Pathema: A Bioinformatics Resource Center funded by the NIAID
- In depth curatorial analysis of six bacterial pathogens will be available:

Category A priority pathogens:	**Category B priority pathogens:**
Bacillus anthracis	Burkholderia mallei
Clostridium botulinum	Burkholderia pseudomallei
Francisella tularensis	Clostridium perfringens

- Pathema will also contain data on the genomes, genes and many functions related to pathogenicity on the complete set of NIAID category A-C bacterial pathogens
- Many of our tools will be used to drive this project
- In turn, helping us provide more services to you
- For More Info: http://www.tigr.org/pathema/

Acknowledgements

- Development:
  - Todd Creasy – Lead Developer and Support Guru for you
  - Sam Angiuoli, Anup Mahurkar, Victor Felix, Tanja Davidsen, Jeremy Peterson
  - Owen White - The Boss
- Slave Drivers (i.e. The Annotators):
  - Michelle Gwinn – Critical liaison for Manatee development
  - Ramana Madupu – Wonderful tester and catcher of my numerous bugs
  - The Whole Microbial Annotation Team – Without them, Manatee wouldn’t exist!