Session 2 Outline

- Review database basics
- Review E-R diagrams and db4bio
- Data types and values
- Relational algebra
- Mining/querying a database with SQL
- Querying multiple tables

Database Basics

- Databases are composed of tables (relations)
- Tables are entities that have attributes (column labels) and tuples (rows)
- Databases can be designed from E-R diagrams that are easily converted to tables
- Primary keys uniquely identify individual tuples and represent links between tables

Building an E-R Diagram

- Identify data attributes
- Conceptualize entities by grouping related attributes
- Identify relationships/links
- Draw preliminary E-R diagram
- Add cardinalities and references
- Refine E-R diagram by applying design principles

Database Normalization

- Goal: To design tables where every non-key column is dependent on the key
  - Why? To reduce redundancy and improve efficiency
- Main requirements:
  - Use primary keys
  - Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
  - Remove columns that are not dependent upon the primary key and place them in separate tables.

Connecting to MySQL

- (Optional) connect to another computer
  
  ssh hebrides.wi.mit.edu -l username

- Connect to MySQL database server
  
  mysql -u username -p -D db4bio
  
  -h hebrides.wi.mit.edu

  mysql>

  - SQL commands are case insensitive
  - Tables and attributes are case sensitive
Relational Databases for Biologists © Whitehead Institute, 2006

### Number Data Types
- **INT**
  - Signed -2147483648 to 2147483647
  - Unsigned 1844674407370551615
- **FLOAT/DOUBLE[[M,D]]**
  - Decimal values, 1.234, 1.4756439E+5
  - M is display size, D is number of decimals
- **DATE/DATETIME**
  - ‘1000-01-01 00:00:00’ to ‘9999-12-31 23:59:59’
  - ‘YYYY-MM-DD HH:MM:SS’
- **TIMESTAMP**
  - YYYYMMDDHHMMSS

### Character Data Types
- **VARCHAR(M)**
  - M characters is length, Text up to 255 characters
  - VARCHAR(5)
    - Will store Apple as ‘Apple’
    - Will store Pineapple as ‘Pineap’
- **TEXT**
  - Text up to 65535 characters
- **VARCHARs and TEXTs** must always be described inside of quotes, single or double
  - Food = “Apple”

### Data Values
- **NULL vs. NOT NULL**
  - Data can either require a value for each tuple or not need one.
- **KEY**
  - Primary keys must be NOT NULL
- **Default**
  - If an attribute was specified as NULL its default is automatically NULL (characters) or empty (numbers).
  - If an attribute was specified as NOT NULL its default value is automatically “” (characters) or zero (numbers).
  - The default value can also be specified manually.

### Using DESCRIBE

```
> DESCRIBE Data;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| affyId | varchar(30) |      | PRI |         |       |
| exptId | varchar(10) |      | PRI |         |       |
| level  | int(11)     |      |     | 0       |       |
+--------+-------------+------+-----+---------+-------+
```

```
> DESCRIBE LocusDescr;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| linkId | int(11)     | PRI  | 0   |         |       |
| description | varchar(100) | YES | NULL |         |       |
| species | varchar(20) | YES | NULL |         |       |
+--------+-------------+------+-----+---------+-------+
```

### Relational Algebra
- **Restrict**: Remove tuples that don’t fit a specific criteria.
- **Project**: Remove specific attributes
Restrict

- List all human tuples in Targets

```sql
SELECT * FROM Targets WHERE species="Hs" LIMIT 5;
```

```
+-----------+----------+---------+
<table>
<thead>
<tr>
<th>affyId</th>
<th>gbId</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>100_g_at</td>
<td>Y08200</td>
<td>Hs</td>
</tr>
<tr>
<td>1000_at</td>
<td>X60188</td>
<td>Hs</td>
</tr>
<tr>
<td>100000_at</td>
<td>AI846313</td>
<td>Mm</td>
</tr>
<tr>
<td>100001_at</td>
<td>M18228</td>
<td>Mm</td>
</tr>
<tr>
<td>100002_at</td>
<td>X70393</td>
<td>Mm</td>
</tr>
</tbody>
</table>
```

sample Targets data

Project

- List nucleotide RefSeqs in RefSeqs table

```sql
SELECT ntRefSeq FROM RefSeqs LIMIT 5;
```

```
+-----------+
<table>
<thead>
<tr>
<th>ntRefSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM_130786</td>
</tr>
<tr>
<td>NM_000014</td>
</tr>
<tr>
<td>NG_001067</td>
</tr>
<tr>
<td>NM_000662</td>
</tr>
<tr>
<td>NM_000015</td>
</tr>
</tbody>
</table>
```

Aggregates

- Aggregates act on an attribute (column)
  - AVG()
    - AVG(level)
  - COUNT()
    - COUNT(affyId)
  - MAX()
    - MAX(affyId)
  - MIN()
    - MIN(species)
  - SUM()
    - SUM(email)

```sql
SELECT COUNT(affyId) FROM Data WHERE level > 5000;
```

```
+---------------+
<table>
<thead>
<tr>
<th>count(affyId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>789</td>
</tr>
</tbody>
</table>
```

```sql
SELECT COUNT(DISTINCT affyId) FROM Data WHERE level > 5000;
```

```
+------------------------+
<table>
<thead>
<tr>
<th>count(distinct affyId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
</tr>
</tbody>
</table>
```

Basic Arithmetic

- List expression levels and twice level in data table

```sql
SELECT level, level*2 FROM Data LIMIT 5;
```

```
+-----------------+----------+-------+
<table>
<thead>
<tr>
<th>affyId</th>
<th>exptId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-MurIL2_at</td>
<td>hs-cer-1</td>
<td>20</td>
</tr>
<tr>
<td>AFFX-MurIL10_at</td>
<td>hs-cer-1</td>
<td>8</td>
</tr>
<tr>
<td>AFFX-MurIL4_at</td>
<td>hs-cer-1</td>
<td>77</td>
</tr>
<tr>
<td>AFFX-BioB-5_at</td>
<td>hs-cer-1</td>
<td>30</td>
</tr>
<tr>
<td>AFFX-MurFAS_at</td>
<td>hs-cer-1</td>
<td>258</td>
</tr>
</tbody>
</table>
```

```sql
SELECT level, level*2 FROM Data LIMIT 5;
```

```
+-------+---------+
<table>
<thead>
<tr>
<th>level</th>
<th>level*2</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>77</td>
<td>154</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>258</td>
<td>516</td>
</tr>
</tbody>
</table>
```

Using DISTINCT

```sql
SELECT count(affyId) FROM Data;
```

```
+------------------------+
<table>
<thead>
<tr>
<th>count(affyId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>789</td>
</tr>
</tbody>
</table>
```

```sql
SELECT count(DISTINCT affyId) FROM Data;
```

```
+--------------------------+
<table>
<thead>
<tr>
<th>count(DISTINCT affyId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
</tr>
</tbody>
</table>
```

Using WHERE

- Restricts queries based on text, numerical value, including inequalities and patterns
- Not equal: !=

```sql
SELECT * FROM GO_Descr WHERE description = "collagen";
```

```sql
SELECT * FROM Data WHERE affyId != "1000_at" LIMIT 3;
```

```
+------------+-------------+
<table>
<thead>
<tr>
<th>goAcc</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0005202</td>
<td>collagen</td>
</tr>
<tr>
<td>GO:0005581</td>
<td>collagen</td>
</tr>
</tbody>
</table>
```

```
+-----------+----------+-------+
<table>
<thead>
<tr>
<th>affyId</th>
<th>exptId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-MurIL2_at</td>
<td>mm-cer-1</td>
<td>20</td>
</tr>
<tr>
<td>AFFX-MurIL10_at</td>
<td>mm-hrt-1</td>
<td>5</td>
</tr>
<tr>
<td>AFFX-MurIL4_at</td>
<td>mm-liv-1</td>
<td>20</td>
</tr>
</tbody>
</table>
```
Using **WHERE**

- LIKE, NOT LIKE: for patterns
- Wildcard: %

```sql
> SELECT ntRefSeq, aaRefSeq
    FROM RefSeqs
    WHERE linkId = 10;

> SELECT *
    FROM RefSeqs
    WHERE linkId LIKE "105%"
    LIMIT 5;
```

<table>
<thead>
<tr>
<th>ntRefSeq</th>
<th>aaRefSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM_000015</td>
<td>NP_000006</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using **ORDER BY**

- Lists results in numerical/alphabetical order according to specified tuples

```sql
> SELECT *
    FROM RefSeqs
    WHERE linkId LIKE "105%"
    ORDER BY linkId DESC;

> SELECT *
    FROM RefSeqs
    WHERE linkId LIKE "105%"
    ORDER BY aaRefSeq ASC;
```

Advanced **WHERE**

```sql
> SELECT affyId, level
    FROM Data
    WHERE level BETWEEN 80 AND 100
    LIMIT 5;

> SELECT *
    FROM UniSeqs
    WHERE gbId
    NOT LIKE "NM_%" LIMIT 5;
```

<table>
<thead>
<tr>
<th>affyId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-BioB-3_at</td>
<td>97</td>
</tr>
<tr>
<td>AFFX-HUMTFRR/M11507_3_at</td>
<td>90</td>
</tr>
<tr>
<td>31324_at</td>
<td>91</td>
</tr>
<tr>
<td>31356_at</td>
<td>95</td>
</tr>
</tbody>
</table>

Mining with **WHERE**

```sql
> SELECT affyId, level
    FROM Data
    WHERE level BETWEEN 80 AND 100
    OR level < 21
    LIMIT 5;

> SELECT affyId, level
    FROM Data
    WHERE exptId != "hs-cer-1"
    AND level BETWEEN 250 AND 300
    LIMIT 5;
```

<table>
<thead>
<tr>
<th>affyId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-M27830_3_at</td>
<td>271</td>
</tr>
<tr>
<td>AFFX-HUMGAPDH/M33197_3_st</td>
<td>277</td>
</tr>
<tr>
<td>31315_at</td>
<td>250</td>
</tr>
<tr>
<td>31362_at</td>
<td>256</td>
</tr>
</tbody>
</table>

### Table Join

- Taking the product of two matrices where merged tuples (rows) must satisfy a specific requirement

```
+-------+-------+-------+
| A1    | B1    | Join  |
| A2    | B2    |      |
| A3    | B3    |      |
|       |       |      |
+-------+-------+-------+
```

```
Join Based on B Column
```

Natural Joins

- Table joining links tables together through their relationships and allows you to traverse your schema/database
- Use SELECT and FROM to join tables
- Join through common attributes with **WHERE** and **AND** using operators: =, <, >, !=, >=, <=
- Traverse from descriptions to sources
Table Self Join

- Identify relationships between data within a single table

```sql
> SELECT Data1.affyId, Data1.exptId as exptId1, Data2.exptId as exptId2,
   Data1.level as level1, Data2.level as level2
FROM Data Data1, Data Data2
WHERE Data1.affyId=Data2.affyId
AND Data1.level >= Data2.level*2
LIMIT 5;
```

<table>
<thead>
<tr>
<th>Data</th>
<th>affyId</th>
<th>exptId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data1</td>
<td>AFFX-MurIL10_at</td>
<td>hs-cer-1</td>
<td>8</td>
</tr>
<tr>
<td>Data2</td>
<td>AFFX-MurIL4_at</td>
<td>hs-cer-1</td>
<td>77</td>
</tr>
<tr>
<td>Data1</td>
<td>AFFX-BioB-M_at</td>
<td>hs-cer-1</td>
<td>214</td>
</tr>
<tr>
<td>Data2</td>
<td>AFFX-BioB-M_at</td>
<td>hs-cer-1</td>
<td>214</td>
</tr>
<tr>
<td>Data1</td>
<td>AFFX-BioB-M_at</td>
<td>hs-cer-1</td>
<td>214</td>
</tr>
</tbody>
</table>

Summary

- Tables store data of specific types
- Restrict and project
- Mining/querying a database with SQL
- Querying multiple tables
- Table joins highlight the relationships between data in a database

Next Session

- Build your own database!
- Use SQL to create tables and specify their structure
- Use SQL to INSERT and DELETE data into your database
- Use SQL to UPDATE/modify your database
- Input data files directly into your database