### Relational Databases for Biologists

**Session 2**
SQL To Data Mine
A Database

Robert Latek, Ph.D.
Sr. Bioinformatics Scientist
Whitehead Institute for Biomedical Research

---

### Session 2 Outline

- Database Basics
- Review E-R Diagrams And db4bio
- Data Types And Values
- Connecting To MySQL
- Relational Algebra
- Data Mining SQL

---

### Database Basics

- Databases Are Composed Of Tables (Relations)
- Relations Are Entities That Have Attributes (Column Labels) And Tuples (Records)
- 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

---

### db4bio E-R Diagram II

---

### Number Data Types

- **INT**
  - Signed -2147483648 to 2147483647
  - Unsigned 1844674407370551615
- **FLOAT/DOUBLE[(M,D)]**
  - Decimal values, 1.234, 1.47564839E+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
  - Will store Apple as ‘Apple’
  - Will store Pineapple as ‘Pineapple’

- 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.

Connecting To MySQL

- If No Local MySQL, In Terminal Window
  - % ssh hebrides.wi.mit.edu -l username

- Connect to MySQL Database Server
  - % mysql -u username -p -D db4bio
  - mysql>

- SQL Commands Are Case-Insensitive
- Tables And Attributes Are Case-Sensitive

Relational Algebra

- Restrict: Remove Tuples That Don’t Fit a Specific Criteria.

- Project: Remove Specific Attributes

Table Product And Divide

- Product: Merge Tuples From Two Tables In Every Possible Way

- Divide: Separate Tuples That Have Every Tuple In Another Table
Table Join

- Similar To Product Except That Merged Tuples Must Satisfy A Specific Requirement

<table>
<thead>
<tr>
<th>A1</th>
<th>B1</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>B2</td>
<td>C2</td>
</tr>
<tr>
<td>A3</td>
<td>B3</td>
<td>C3</td>
</tr>
</tbody>
</table>

\[ \text{Join} \]

Table Algebra

- **Union:** Combine Tuples From Both Tables Without Duplicates
- **Intersection:** Remove Tuples That Are Not Found In Both Tables
- **Difference:** Remove Tuples That Are Not Shared In One Of The Tables

Aggregates

- Aggregates Act On An Attribute
  - AVG()
  - COUNT()
  - MAX()
  - MIN()
  - SUM()

Project

- List Nucleotide RefSeqs In RefSeqs Table

\[ \text{SELECT ntRefSeq FROM RefSeqs LIMIT 5;} \]

Project With Math

- List Expression Levels And Twice Level In Data Table

\[ \text{SELECT level, level*2 FROM Data LIMIT 5;} \]

Restrict

- List All Human Tuples in Targets

\[ \text{SELECT * FROM Targets WHERE species="Hs" LIMIT 5;} \]
Using WHERE

- Restricts Queries Having Lists, Ranges, Inequalities, Patterns

> SELECT nRefSeq, aRefSeq FROM RefSeqs WHERE linkId = 10;

> SELECT * FROM Data WHERE level BETWEEN 80 AND 100;

Using ORDER BY

- Lists Results In Numerical/Alphabetical Order According To Specified Tuples

> SELECT nRefSeq, aRefSeq FROM RefSeqs WHERE level LIKE "105%" ORDER BY level DESC;

Using WHERE

> SELECT * FROM GO_Descr WHERE description = "collagen";

> SELECT * FROM Data WHERE linkId = 1000_at;

Using GROUP BY

- Operates Only On The Tuples That Were Not Removed By a Where

> SELECT * FROM Data WHERE level < 100 GROUP BY level, aRefSeq LIMIT 5;

Mining With WHERE

> SELECT * FROM Data WHERE level BETWEEN 80 AND 100 LIMIT 5;

> SELECT * FROM Data WHERE level BETWEEN 100 AND 200 LIMIT 5;
### Using HAVING

- Sets the conditions for the GROUP BY clause like WHERE sets conditions for SELECT.
- CAN use aggregates.

```sql
> SELECT description AS gene_description,
FROM GO_Descr
GROUP BY description
HAVING COUNT(description) > 1
LIMIT 5;
```

<table>
<thead>
<tr>
<th>description</th>
<th>species</th>
<th>gbId</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-phosphatidylinositol-4-phosphate kinase, class I</td>
<td>Homo sapiens</td>
<td>HSU30872</td>
</tr>
<tr>
<td>lipase, gastric</td>
<td>Homo sapiens</td>
<td>HSLFBPS7</td>
</tr>
<tr>
<td>serine (or cysteine) protein kinase inhibitor</td>
<td>Homo sapiens</td>
<td>HSLFBPS7</td>
</tr>
<tr>
<td>albumin</td>
<td>Homo sapiens</td>
<td>HSLFBPS7</td>
</tr>
<tr>
<td>S100 calcium binding protein A8</td>
<td>Homo sapiens</td>
<td>HSLFBPS7</td>
</tr>
</tbody>
</table>

### Multiple Table Join

```sql
> SELECT Descriptions.description AS description,
GO_Descr.goAcc AS goDescription,
LocusLinks.gbId AS gbId
FROM Descriptions, GO_Descr, LocusLinks
WHERE Descriptions.gbId = Targets.gbId
AND Targets.gbId = LocusLinks.gbId
AND LocusLinks.linkId = Ontologies.linkId
AND Ontologies.goAcc = GO_Descr.goAcc
LIMIT 5;
```

<table>
<thead>
<tr>
<th>Descriptions</th>
<th>Targets</th>
<th>LocusLinks</th>
<th>Ontologies</th>
<th>GO_Descr</th>
</tr>
</thead>
<tbody>
<tr>
<td>gbId</td>
<td>gbId</td>
<td>gbId</td>
<td>goAcc</td>
<td>goDescription</td>
</tr>
</tbody>
</table>

### Binary Table Join

```sql
> SELECT LocusDescr.description,
LocusDescr.species,
LocusLinks.gbId
FROM LocusDescr, LocusLinks
WHERE LocusDescr.linkId = LocusLinks.linkId
GROUP BY LocusLinks.gbId
LIMIT 5;
```

### Binary Table Join

```sql
> SELECT GO_Descr.description,
Ontologies.linkId
FROM GO_Descr, Ontologies
WHERE Ontologies.goAcc = GO_Descr.goAcc
LIMIT 5;
```

### Mega Table Join

```sql
> SELECT Descriptions.description,
Sources.source,
RefSeqs.ntRefSeq
FROM Descriptions,
Sources,
RefSeqs,
Targets,
LocusLinks,
Data
WHERE Descriptions.gbId = Targets.gbId
AND Targets.gbId = LocusLinks.gbId
AND LocusLinks.linkId = RefSeqs.linkId
AND Targets.affyId = Data.affyId
AND Data.expAnd = Sources.expAnd
LIMIT 5;
```

### 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 theta operators: =, <, >, !=, >=, <=
- Traverse from descriptions to sources.

### Using AND AND Formulas

- Use AND AND (AND) expressions to join tables.
- Aggregates can be used in the GROUP BY clause.

### Using OR OR Formulas

- Use OR OR (OR) expressions to join tables.
- Aggregates can be used in the GROUP BY clause.
Table Self Join

- Identify Relationships Between Data Within A Single Table

> 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>affyId</th>
<th>exptId1</th>
<th>exptId2</th>
<th>level1</th>
<th>level2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-MurIL10_at</td>
<td>hs-cer-1</td>
<td>hs-hrt-1</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>AFFX-MurIL4_at</td>
<td>hs-cer-1</td>
<td>ha-hrt-1</td>
<td>77</td>
<td>20</td>
</tr>
<tr>
<td>AFFX-BioB-M_at</td>
<td>ha-cer-1</td>
<td>mm-cer-1</td>
<td>214</td>
<td>20</td>
</tr>
<tr>
<td>AFFX-BioB-M_at</td>
<td>ha-cer-1</td>
<td>mm-hrt-1</td>
<td>214</td>
<td>48</td>
</tr>
<tr>
<td>AFFX-BioB-M_at</td>
<td>ha-cer-1</td>
<td>mm-liv-1</td>
<td>214</td>
<td>20</td>
</tr>
</tbody>
</table>

Master Table Self Join

> 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 BETWEEN Data2.level*2 AND Data2.level*3
ORDER BY Data1.affyId
LIMIT 5;

<table>
<thead>
<tr>
<th>affyId</th>
<th>exptId1</th>
<th>exptId2</th>
<th>level1</th>
<th>level2</th>
</tr>
</thead>
<tbody>
<tr>
<td>100014_at</td>
<td>mm-hrt-1</td>
<td>mm-liv-1</td>
<td>52</td>
<td>20</td>
</tr>
<tr>
<td>100014_at</td>
<td>mm-cer-1</td>
<td>mm-liv-1</td>
<td>55</td>
<td>20</td>
</tr>
<tr>
<td>100015_at</td>
<td>mm-cer-1</td>
<td>mm-hrt-1</td>
<td>943</td>
<td>396</td>
</tr>
<tr>
<td>100015_at</td>
<td>mm-cer-1</td>
<td>mm-liv-1</td>
<td>943</td>
<td>468</td>
</tr>
<tr>
<td>100024_at</td>
<td>mm-hrt-1</td>
<td>mm-liv-1</td>
<td>306</td>
<td>111</td>
</tr>
</tbody>
</table>

Summary

- Tables Store Data Of Specific Types
- Data Can Have Default Values And Be NOT NULL Restricted
- Restrict And Project
- Use WHERE Or HAVING To Constrain SELECT
- Table Joins Highlight The Relationships Between Data In A Database

Next Week

- 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