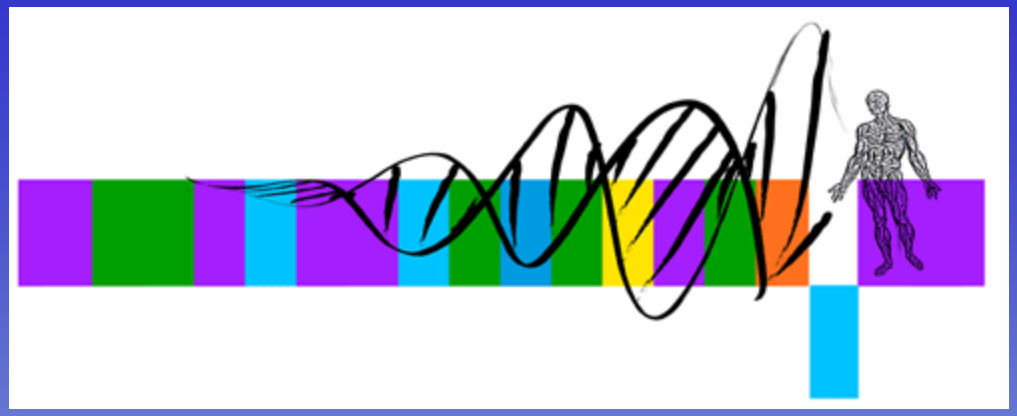


Relational Databases for Biologists: Efficiently Managing and Manipulating Your Data

Session 2:
Mining a database with SQL

George Bell, Ph.D.
WIBR Bioinformatics and Research Computing

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

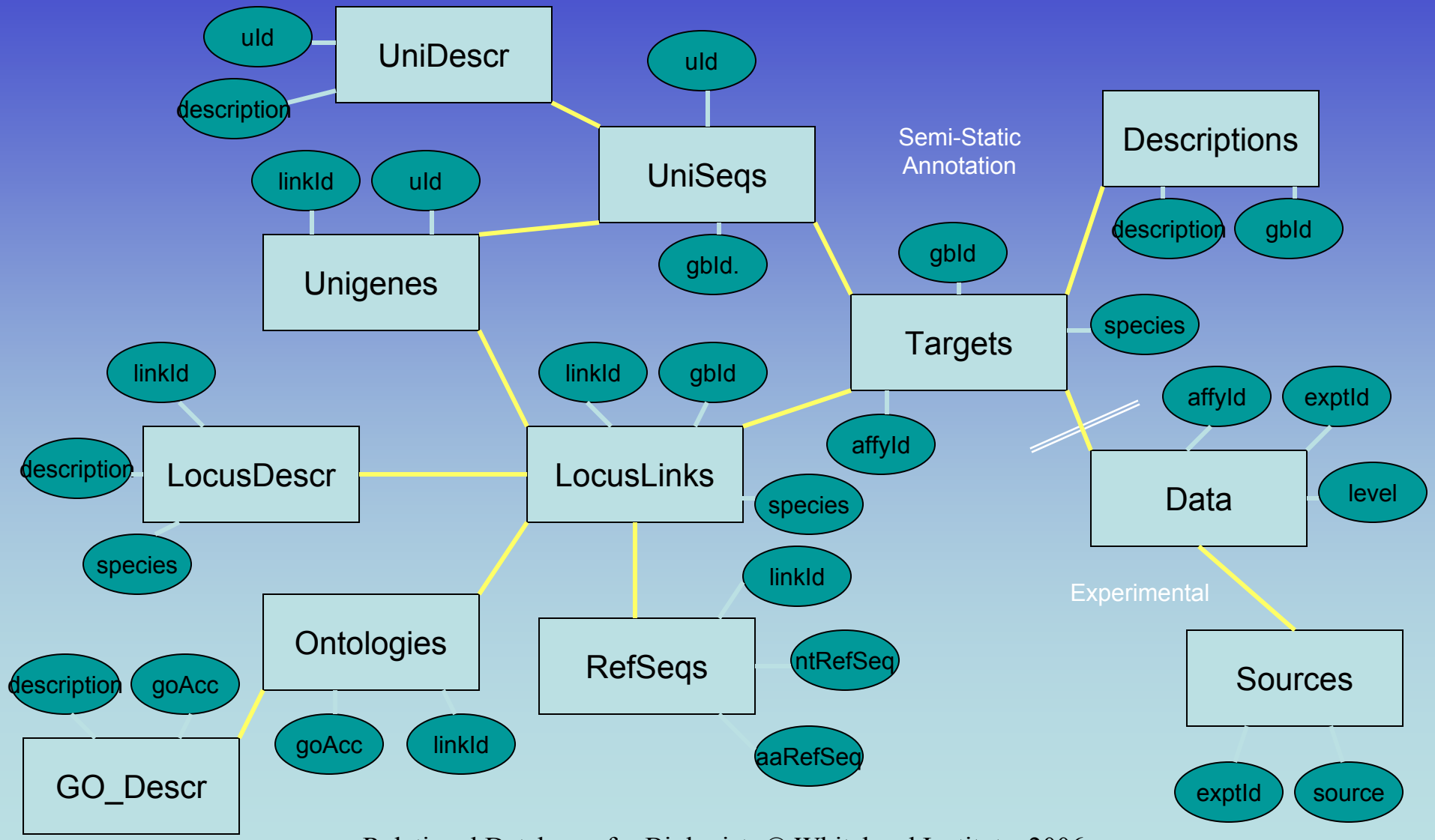
  mysql -u username -p -D db4bio

  –h hebrides.wi.mit.edu

  mysql>

- SQL commands are case-insensitive
- Tables and attributes are case-sensitive

db4bio E-R Diagram



Number Data Types

- **INT**
  - Signed -2147483648 to 2147483647
  - Unsigned 18446744073705516l5
- **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
  - VARCHAR(5)
    - Will store Apple as ‘Apple’
    - Will store Pineapple as ‘Pinea’
- **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;

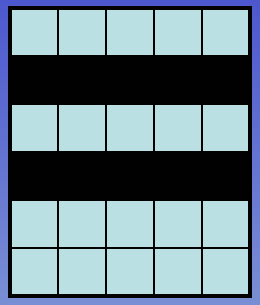
```
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| affyId | varchar(30) |      | PRI |         |       |
| exptId | varchar(10) |      | PRI |         |       |
| level  | int(11) 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

> DESCRIBE LocusDescr;

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

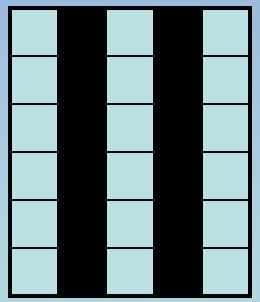
Relational Algebra

- Restrict: Remove tuples that don't fit a specific criteria.



Restrict

- Project: Remove specific attributes



Project

Restrict

- List all human tuples in Targets

affyId	gbId	species
100_g_at	Y08200	Hs
1000_at	X60188	Hs
100000_at	AI846313	Mm
100001_at	M18228	Mm
100002_at	X70393	Mm

sample Targets data

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

affyId	gbId	species
100_g_at	Y08200	Hs
1000_at	X60188	Hs
1001_at	X60957	Hs
1002_f_at	X65962	Hs
1003_s_at	X68149	Hs

Project

- List nucleotide RefSeqs in RefSeqs table

linkId	ntRefSeq	aaRefSeq
1	NM_130786	NP_570602
2	NM_000014	NP_000005
3	NG_001067	
9	NM_000662	NP_000653
10	NM_000015	NP_000006

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

ntRefSeq
NM_130786
NM_000014
NG_001067
NM_000662
NM_000015

Aggregates

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

Using DISTINCT

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

```
+---------------+
| count(affyId) |
+---------------+
|           789 |
+---------------+
```

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

```
+------------------------+
| count(distinct affyId) |
+------------------------+
|                    600 |
+------------------------+
```

Basic Arithmetic

- List expression levels and twice level in data table

affyId	exptId	level
AFFX-MurIL2_at	hs-cer-1	20
AFFX-MurIL10_at	hs-cer-1	8
AFFX-MurIL4_at	hs-cer-1	77
AFFX-MurFAS_at	hs-cer-1	30
AFFX-BioB-5_at	hs-cer-1	258

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

level	level*2
20	40
8	16
77	154
30	60
258	516

Using WHERE

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

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

```
+------------+-------------+
| goAcc      | description |
+------------+-------------+
| GO:0005202 | collagen    |
| GO:0005581 | collagen    |
+------------+-------------+
```

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

```
+-----------+----------+-------+
| affyId    | exptId   | level |
+-----------+----------+-------+
| 100001_at | mm-cer-1 |    20 |
| 100001_at | mm-hrt-1 |     5 |
| 100001_at | mm-liv-1 |    20 |
+-----------+----------+-------+
```

Using WHERE

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

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

ntRefSeq	aaRefSeq
NM_000015	NP_000006

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

linkId	ntRefSeq	aaRefSeq
1050	NM_004364	NP_004355
1051	NM_005194	NP_005185
1052	NM_005195	NP_005186
1053	NM_001805	NP_001796
1054	NM_001806	NP_001797

Using ORDER BY

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

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

```
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
| 105910 | NM_134094 | NP_598855 |
| 105892 | XM_128276 | XP_128276 |
| 105887 | XM_127943 | XP_127943 |
| 105870 | XM_128254 | XP_128254 |
| 105866 | XM_128271 | XP_128271 |
+--------+-----------+-----------+
```

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

```
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
|   1057 | NR_001275 |           |
|   1053 | NM_001805 | NP_001796 |
|   1054 | NM_001806 | NP_001797 |
|   1056 | NM_001807 | NP_001798 |
|   1058 | NM_001809 | NP_001800 |
+--------+-----------+-----------+
```

Advanced WHERE

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

affyId	level
AFFX-BioB-3_at	97
AFFX-HUMTFRR/M11507_3_at	90
AFFX-HSAC07/X00351_M_st	86
31324_at	91
31356_at	91

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

uId	gbId
Hs.2	D90042
Hs.4	X03350
Hs.11	D90278
Hs.11	L00693
Hs.21	M16652

Mining with WHERE

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

→

affyId	exptId	level
AFFX-MurIL2_at	hs-cer-1	20
AFFX-MurIL10_at	hs-cer-1	8
AFFX-BioB-3_at	hs-cer-1	97
AFFX-BioB-5_st	hs-cer-1	20
AFFX-BioB-M_st	hs-cer-1	20

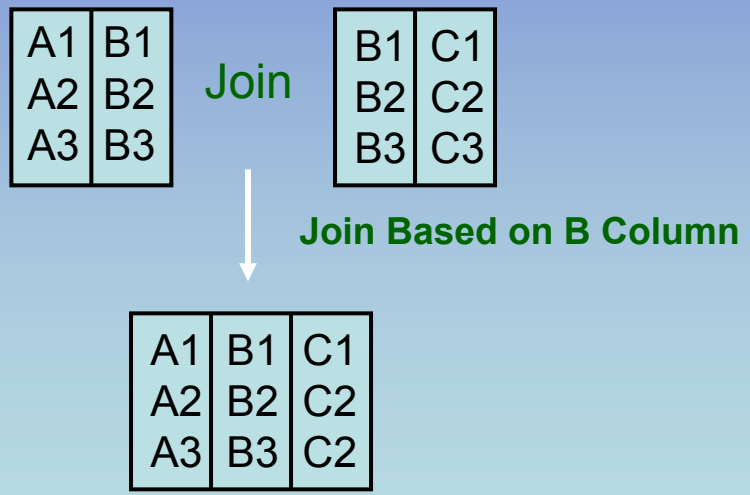
```
> SELECT affyId, level
  FROM Data
  WHERE exptId != "hs-cer-1"
  AND level BETWEEN
  250 AND 300
  LIMIT 5;
```

→

affyId	level
AFFX-M27830_3_at	271
AFFX-HUMGAPDH/M33197_3_st	277
31315_at	250
31362_at	256
31510_s_at	257

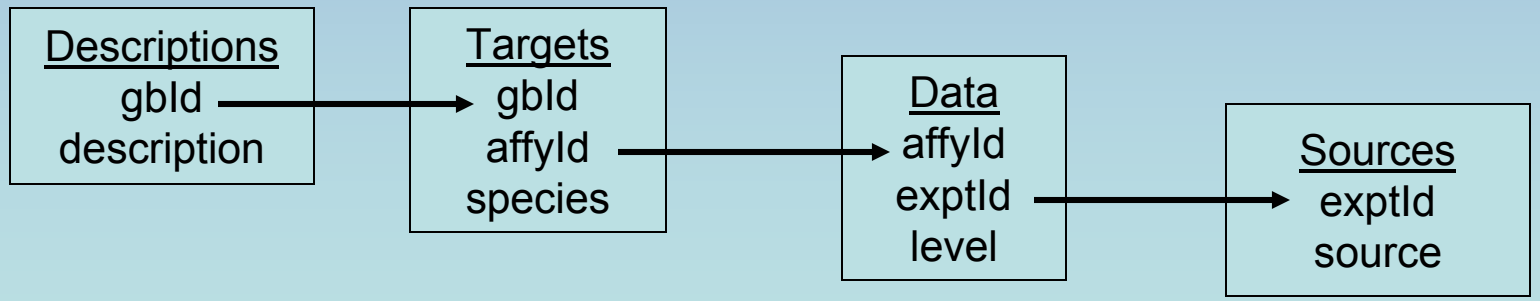
Table Join

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



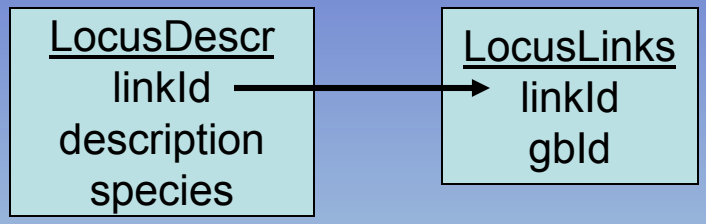
Natural Joins

- 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



Binary Table Join

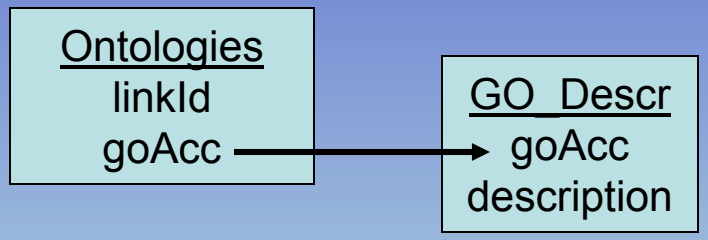
```
> SELECT LocusDescr.description, LocusDescr.species, LocusLinks.gbId
  FROM LocusDescr, LocusLinks
  WHERE LocusDescr.linkId = LocusLinks.linkId
  GROUP BY LocusLinks.gbId
  LIMIT 5;
```



description	species	gbId
granulysin	Hs	A00142
lipase, gastric	Hs	A01046
serine (or cysteine) proteinase inhibitor	Hs	A03911
albumin	Hs	A06977
S100 calcium binding protein A8	Hs	A12027

Binary Table Join

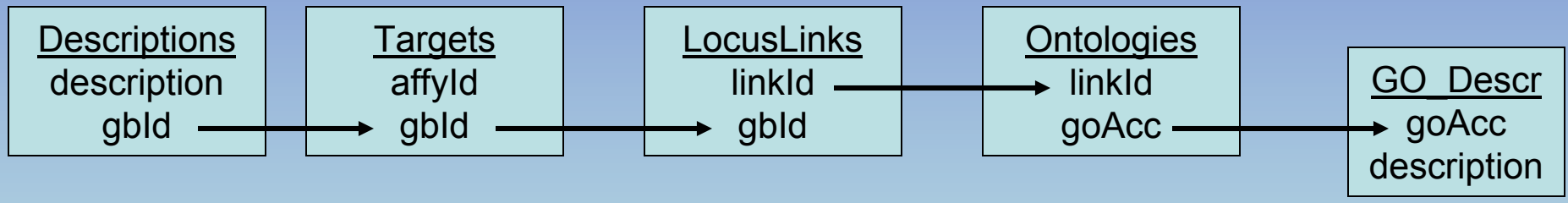
```
> SELECT GO_Descr.description, Ontologies.linkId
  FROM GO_Descr, Ontologies
  WHERE Ontologies.goAcc=GO_Descr.goAcc
  LIMIT 5;
```



description	linkId
protein carrier	2
arylamine N-acetyltransferase	9
arylamine N-acetyltransferase	10
serine protease inhibitor	12
enzyme	13

Multiple Table Join

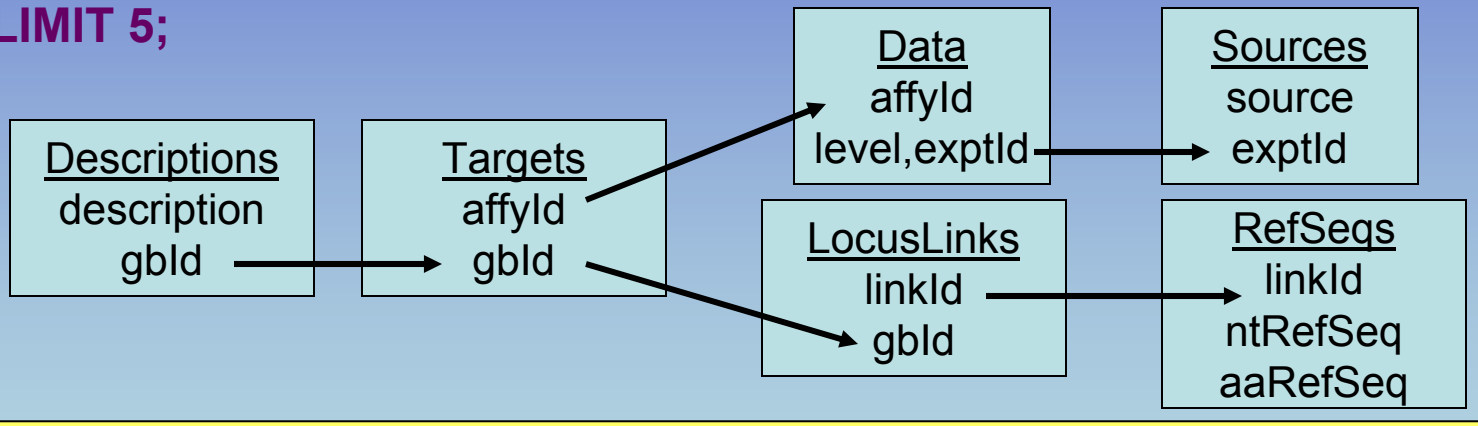
```
> SELECT Descriptions.description AS gene_description,
  GO_Descr.description AS GO_description
  FROM Descriptions, GO_Descr, LocusLinks, Ontologies, Targets
  WHERE Descriptions.gbId=Targets.gbId
  AND Targets.gbId=LocusLinks.gbId
  AND LocusLinks.linkId=Ontologies.linkId
  AND Ontologies.goAcc=GO_Descr.goAcc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



gene_description	GO_description
HSLFBPS7 Human fructose-1,6-biphosphatase	fructose-2,6-bisphosphate 2-phosphatase
HSU30872 Human mitosin mRNA, complete cds	regulation of mitosis
HSU33052 Human lipid-activated, protein kinase	protein kinase
HSU33053 Human lipid-activated protein kinase	protein kinase
HSU33920 Human clone lambda 5 semaphorin mRNA,	extracellular space

Mega Table Join

```
> 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.exptId=Sources.exptId
  LIMIT 5;
```



description	source	ntRefSeq
Homo sapiens immunoglobulin lambda gene locus DNA, clone:288A10	Human liver	NG_000002
HSIGVL009 Human rearranged immunoglobulin lambda light chain mRNA	Human heart	NG_000002
Homo sapiens immunoglobulin lambda gene locus DNA, clone:31F3	Human liver	NG_000002
Homo sapiens immunoglobulin lambda gene locus DNA, clone:288A10	Human brain	NG_000002
HSIGVL009 Human rearranged immunoglobulin lambda light chain mRNA	Human liver	NG_000002

Using GROUP BY

- To group rows by some attribute and get summary info about that group

```
> SELECT affyId,
  MAX(level) as max_level
  FROM Data
  GROUP BY affyId
  ORDER BY max_level DESC
  LIMIT 3;
```

```
+----------+------------+
| affyId   | MAX(level) |
+----------+------------+
| 36780_at |      47914 |
| 39106_at |      42001 |
| 33377_at |      41726 |
+----------+------------+
```

```
> SELECT affyId, AVG(level)
  AS mean_level
  FROM Data
  GROUP BY affyId
  ORDER BY mean_level DESC
  LIMIT 4;
```

```
+------------------+------------+
| affyId           | mean_level |
+------------------+------------+
| AFFX-MURINE_B2_at | 22705.0000 |
| 36780_at         | 22193.0000 |
| 35083_at         | 21245.0000 |
| 31962_at         | 20769.3333 |
+------------------+------------+
```

Using HAVING

- Sets the conditions for the GROUP BY clause like WHERE sets conditions for SELECT

```
> SELECT affyId, SUM(level)/count(level)
  AS mean_level
  FROM Data
  GROUP BY affyId
  HAVING mean_level > 20000;
```

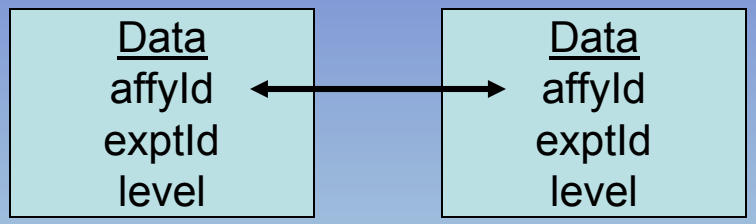
```
+------------------+------------+
| affyId           | mean_level |
+------------------+------------+
| 31962_at         | 20769.3333 |
| 35083_at         | 21245.0000 |
| 36780_at         | 22193.0000 |
| AFFX-MURINE_B2_at | 22705.0000 |
+------------------+------------+
```

```
> SELECT affyId, SUM(level)/count(level)
  AS mean_level
  FROM Data
  GROUP BY affyId
  HAVING MAX(level) > 40000
  AND MIN(level) < 4000
  ORDER BY mean_level ASC;
```

Table Self Join

- Identify relationships between data within a single table

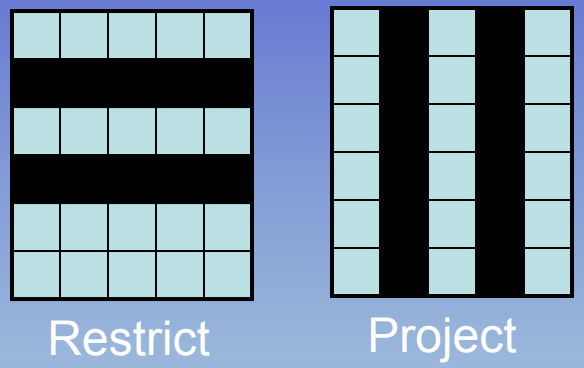
```
> SELECT Data1.affyId, Data1.exptId as exptId1, Data2.exptId as exptId2,
  Data1.level as level1, Data2.level as level2
  FROM Data Data1, Data Data2
  WHERE Data1.affyId=Data2.affyId
  AND Data1.level >= Data2.level*2
  LIMIT 5;
```



affyId	exptId1	exptId2	level1	level2
AFFX-MurIL10_at	hs-cer-1	hs-hrt-1	8	4
AFFX-MurIL4_at	hs-cer-1	hs-hrt-1	77	20
AFFX-BioB-M_at	hs-cer-1	mm-cer-1	214	20
AFFX-BioB-M_at	hs-cer-1	mm-hrt-1	214	48
AFFX-BioB-M_at	hs-cer-1	mm-liv-1	214	20

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