Bioinformatics Data Management

Lecture 3

Structured Databases

MARC: Developing Bioinformatics Programs
July 2009

Alex Ropelewski
PSC-NRBSC
Bienvenido Vélez
UPR Mayaguez

Bioinformatics Data Management

- The following material is the result of a curriculum development effort to provide a set of courses to support bioinformatics efforts involving students from the biological sciences, computer science, and mathematics departments. They have been developed as a part of the NIH funded project “Assisting Bioinformatics Efforts at Minority Schools” (2T36 GM008789). The people involved with the curriculum development effort include:

- Dr. Hugh B. Nicholas, Dr. Troy Wymore, Mr. Alexander Ropelewski and Dr. David Deerfield II, National Resource for Biomedical Supercomputing, Pittsburgh Supercomputing Center, Carnegie Mellon University.
- Dr. Ricardo González Méndez, University of Puerto Rico Medical Sciences Campus.
- Dr. Alade Tokuta, North Carolina Central University.
- Dr. Jaime Seguel and Dr. Bienvenido Vélez, University of Puerto Rico at Mayagüez.
- Dr. Satish Bhalla, Johnson C. Smith University.

Structured Databases: Outline

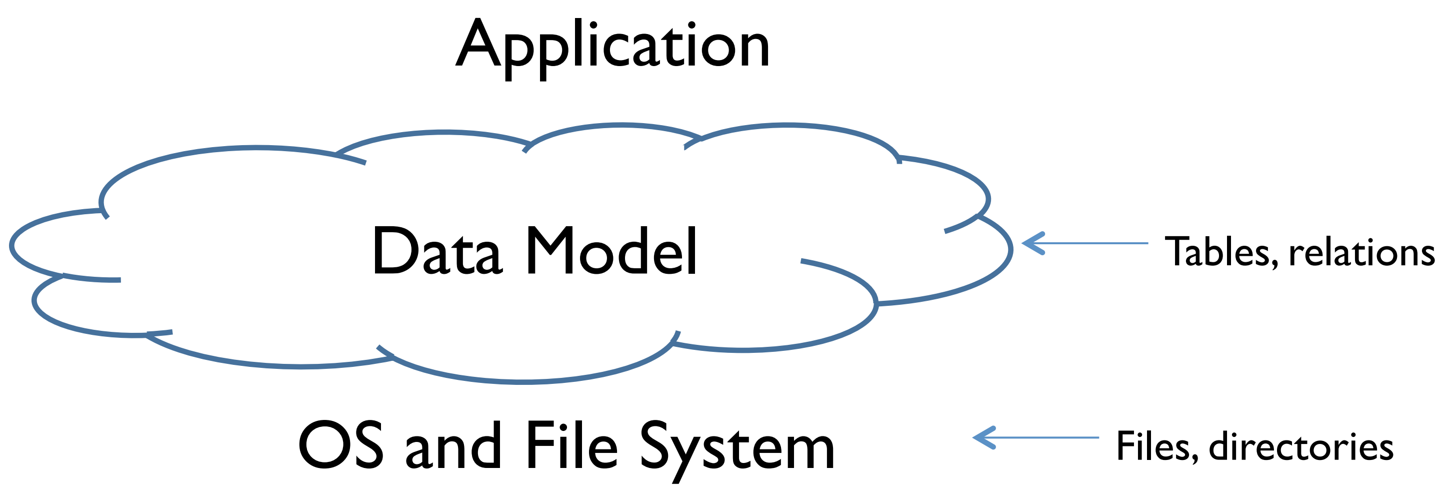
- Structured Databases at a Glance - Characteristics
- Advantages of Structured Databases
- Data Independence
- Disadvantages of Structured Databases
- Examples of Structured Databases
  - Hierarchical Databases
  - Networked Databases
  - Relational Databases
  - XML Databases

Structured Databases at a Glance

- All information organized in same way (Data Model)
- Language available to
  - describe (create) the database
  - insert data
  - manipulate data
  - update
- Language establishes an abstract data model: Data Independence
- Programs using language can work across systems
- Facilitates communication and sharing data

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Data Independence



Objective:

Shield apps from changes in “physical” platform specific layers

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Disadvantages of Structured Databases

Model may not fit data needs

Example: How to represent proteins in table format

Approach #1 – Store one residue per column

AccNum	Name	AA1	AA2	AA3	…	AA517
AAA16331	G-gamma globin	M	G	H		
AAA51693	Aldehyde Dehydrogenase	M	L	R	…	S

Wasted space

Must make number of columns = max length of any possible sequence

What is this number?

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Disadvantages of Structured Databases

Model may not fit data needs

Approach #2 – Store one residue per rows

AccNum	AAPos	AA	Name
AAA16331	1	M	G-gamma globin
AAA16331	2	M	G-gamma globin
…	…	…	…
AAA51693	1	M	Aldehyde Dehydrogenase
…	…	…	…

Difficult to recover sequence in string form using DML

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Disadvantages of Structured Databases

Model may not fit data needs

Approach #3 – Put aminoacid sequence in one attribute

AccNum	Name	Sequence
AAA16331	G-gamma globin	"MGHFTEEDKA…."
AAA51693	Aldehyde Dehydrogenase	"MLRAAARFPGP…."
…	…	

Must analyze sequence using program outside SQL
Loose some benefits of Data Model

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

A Simple Relational Example

- Intuitively model consists of tables
  - Rows are objects or "entities"
  - Columns are "attributes" of entities
  - Attributes cross reference other tables

Proteins

AccessionNum	Name	DiscBy	SeqBy
P201		2	3
Q303		2	1
R32		3	NULL

Primary key

Scientists

Num	Name	Last	Country
1	John	Doe	England
2	Jane	Doe	England
3	Hugh	Nicholas	USA

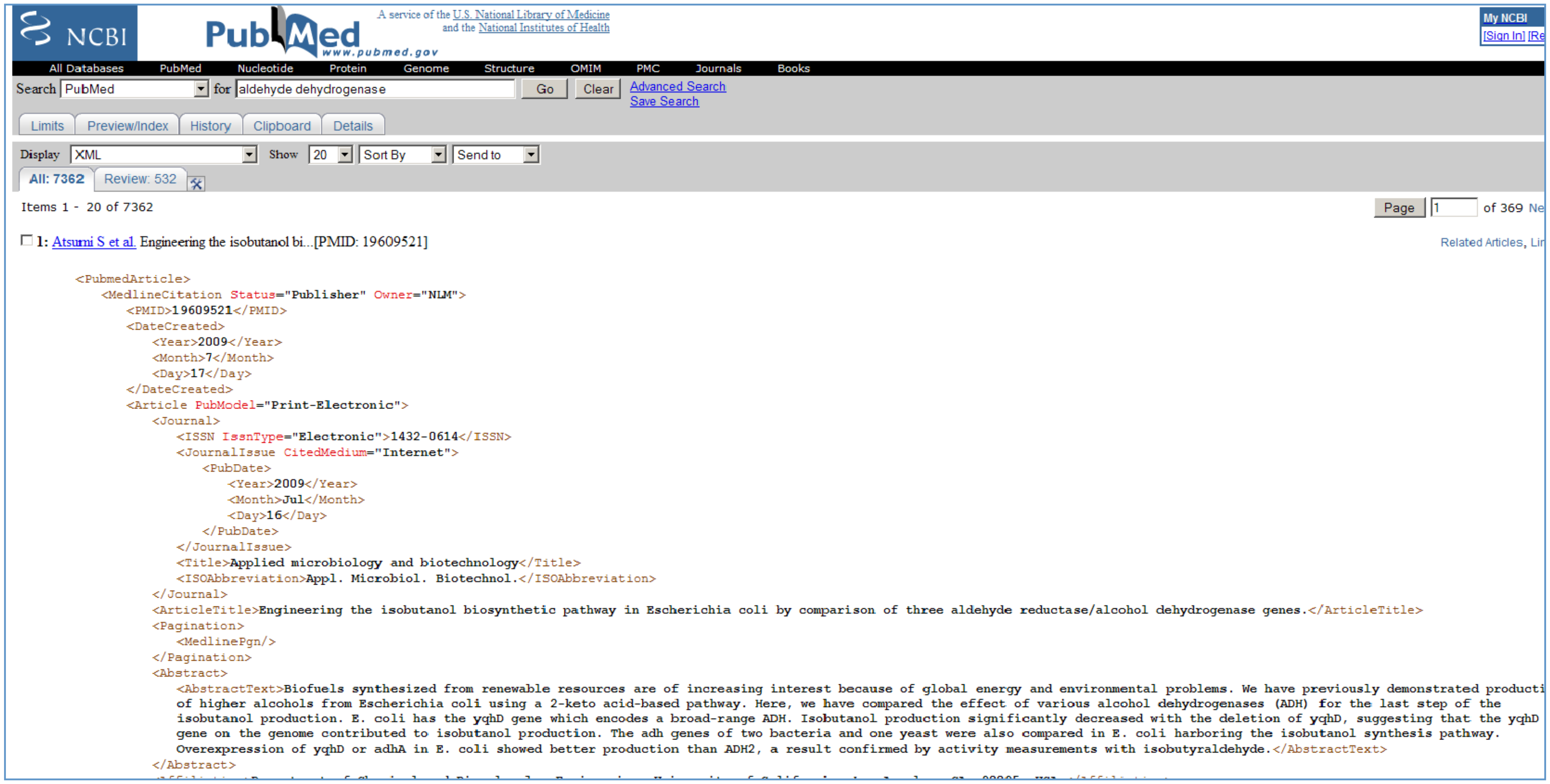
Primary key

Intuitively:
- ➢Protein P201 discovered by Jane Doe
- ➢Hugh discovered R2 and sequenced P201

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Structured Databases: Other examples

- XML Databases



1: Atsumi S et al. Engineering the isobutanol bi...[PMID: 19609521]

```
<PubmedArticle>
    <MedlineCitation Status="Publisher" Owner="NLM">
        <PMID>19609521</PMID>
        <DateCreated>
            <Year>2009</Year>
            <Month>7</Month>
            <Day>17</Day>
        </DateCreated>
        <Article PubModel="Print-Electronic">
            <Journal>
                <ISSN IssnType="Electronic">1432-0614</ISSN>
                <JournalIssue CitedMedium="Internet">
                    <PubDate>
                        <Year>2009</Year>
                        <Month>Jul</Month>
                        <Day>16</Day>
                    </PubDate>
                </JournalIssue>
                <Title>Applied microbiology and biotechnology</Title>
                <ISOAbbreviation>Appl. Microbiol. Biotechnol.</ISOAbbreviation>
            </Journal>
            <ArticleTitle>Engineering the isobutanol biosynthetic pathway in Escherichia coli by comparison of three aldehyde reductase/alcohol dehydrogenase genes.</ArticleTitle>
            <Pagination>
                <MedlinePgn/>
            </Pagination>
            <Abstract>
                <AbstractText>Biofuels synthesized from renewable resources are of increasing interest because of global energy and environmental problems. We have previously demonstrated producti
                of higher alcohols from Escherichia coli using a 2-keto acid-based pathway. Here, we have compared the effect of various alcohol dehydrogenases (ADH) for the last step of the
                isobutanol production. E. coli has the yqhD gene which encodes a broad-range ADH. Isobutanol production significantly decreased with the deletion of yqhD, suggesting that the yqhD
                gene on the genome contributed to isobutanol production. The adh genes of two bacteria and one yeast were also compared in E. coli harboring the isobutanol synthesis pathway.
                Overexpression of yqhD or adhA in E. coli showed better production than ADH2, a result confirmed by activity measurements with isobutyraldehyde.</AbstractText>
            </Abstract>
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

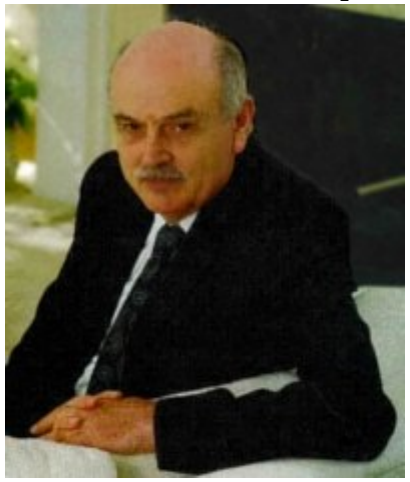
- Query language = XPATH/XQUERY

Relational Databases: Outline

- Introduction and Examples
- Relational Database Design by Example
    - entities and relational diagrams
    - normal forms
- SQL (Sequel) Language
- SQL Data Manipulation
    – Select
    – Joins
    – Updates and deletes
    – Inserts

Relational Databases: Timeline

- Originally proposed by E.F. Codd in 1970
- First research prototypes in early 80's:
  - Ingres @ UC Berkeley
  - System R @ IBM
- Today the market exceeds $20B annually


Edgar F. Codd

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Databases Products

- Commercial
  - Oracle
  - MS SQL Server
  - IBM DB2
- Open Source
  - MySQL
  - Postgres
  - SQLite

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Example Relational Database Design

Goal: Store results from multiple sequence search attempts

Leverage SQL to analyze large result set

Entities to be stored: Matching sequences with scores for each search matrix

Acc#	Definition	Source	Matrix	eValue	SearchDate
P14555	Group IIA Phospholipase A2	Human	Pam70	4.18 E-32	7/21/07
P81479	Phospholipase A2 isozyme IV	Indian Green Tree Viper	Pam70	2.68 -E52	7/21/07
P14555	Group IIA Phospholipase A2	Human	Blosom80	3.47 E-33	7/20/07
P81479	Phospholipase A2 isozyme IV	Indian Green Tree Viper	Blosom80	1.20 E-54	7/20/07

Problems: Lots of redundant information

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Dealing with Redundancy

Normalization

Sequences

AccNum	Definition	Source
P14555	Group IIA Phospholipase A2	Human
P81479	Phospholipase A2 isozyme IV	Indian Green Tree Viper

Matches

Acc#	Date	Matrix	eValue
P14555	7/21/07	Pam70	4.18 E-32
P81479	7/21/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

Foreign key

Still redundant

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Dealing with Redundancy

Normalization

Sequences

Acc#	Definition	Source
P14555	Group IIA Phospholipase A2	Human
P81479	Phospholipase A2 isozyme IV	Indian Green Tree Viper

Matches

Acc#	Run#	eValue
P14555	1	4.18 E-32
P81479	1	2.68 -E52
P14555	2	3.47 E-33
P81479	2	1.20 E-54

Runs

Run#	Matrix	Date
1	7/21/07	Pam70
2	7/20/07	Blosom80

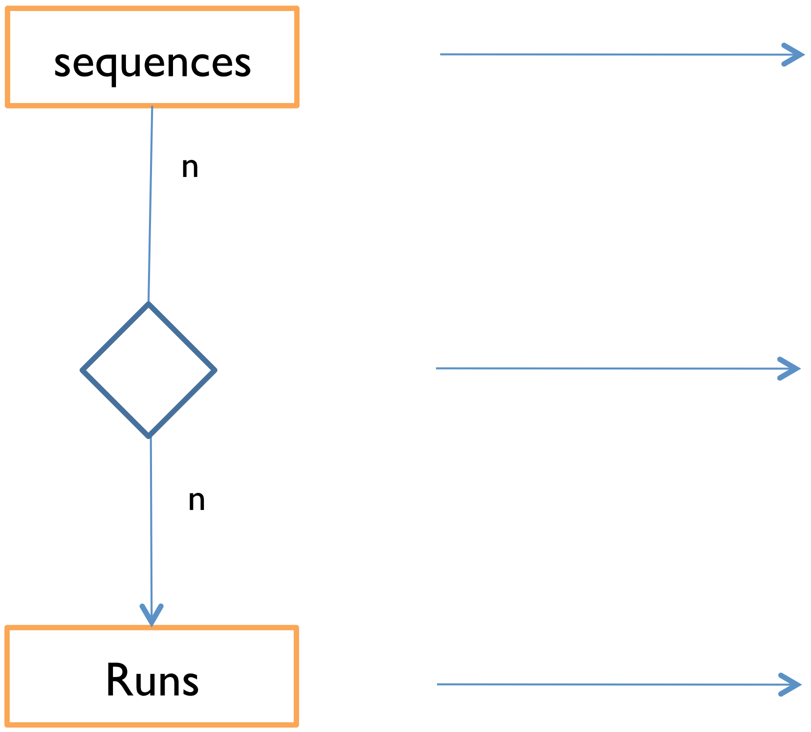
These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Basic Normal Forms

- First Normal Form
  - Table must be flat; no multi-valued attributes
- Second Normal Form
  - All non-key attributes determined by whole primary key
- Third Normal Form
  - All non-key attributes can ONLY depend on whole primary key directly
- Others…

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Entity Relationship Diagrams



Sequences

Acc#	Definition	Source

Matches

Acc#	Run#	eValue

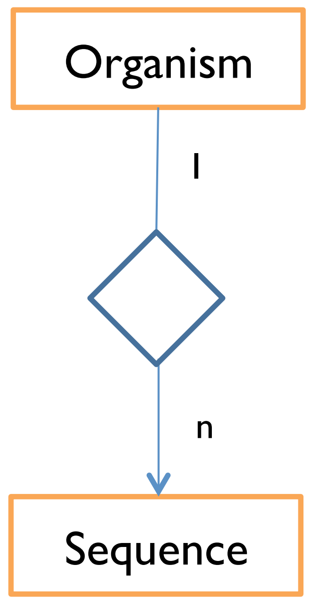
Runs

Run#	Matrix	Date

N-to-N relations model as table

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Entity Relationship Diagrams



Organism

ID	Name	Description
1	Cod worm	Pseudoterranova decipiens
2	Human	Homo sapiens

No table necessary for one-to-many rel's

Sequence

Acc#	SrcID	Name
CAA77743	1	Hemoglobin
AB647031	2	Hemoglobin

one-to-many relationships

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Simple Query Language (SQL)

- Organization
  - Data definition/schema creation
  - Data manipulation
    - Insertion
    - Manipulation
    - Updates
    - Removals
  - A standard (ISO) since 1987

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Data Definition Language (DDL)

- The CREATE statement

Sequences

Acc#	Definition	Source

Runs

Run#	Matrix	Date

Matches

Acc#	Run#	eValue

```
CREATE TABLE Sequences(
            AccNum int,
            Definition varchar (255),
            Source varchar(255) )
CREATE TABLE Runs(
            RunNum int,
            Matrix varchar(255),
            Date date)
CREATE TABLE Matches(
            AccNum int,
            RunNum int,
            eValue int)
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Data Manipulation Language (DML)

- Foundation is relational algebra
- An algebra on relations with three basic operations:

Operation	Description
Projection	Select subset of attributes of a table
Selection	Select subset of entities in a table
Join	Merge two tables into one

All three operations yield a new table as the result

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra - Projection

- Selects a subset of attributes or columns

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

$\pi_{\text{Acc\#, Source}}(\text{Sequences})$

Acc#	Source
CAM22514	Mus Musculus
CAO91797	Mus Musculus
ABG47031	Homo Sapiens

Intuitively:
Find organism that each sequence belongs to

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra - Selection

- Selects a subset of entities or rows

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

$$\downarrow$$

$\sigma_{\text{Source=Mus Musculus}}$ (Sequences)

$$\downarrow$$

Intuitively:
Find all sequences from Mus Musculus

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus

No reason to keep organism is result

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra: Projection and Selection

$\pi_{\text{Acc\#, Definition}}(\sigma_{\text{Source= Mus Musculus}}(\text{Sequences}))$

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

$\sigma_{\text{Source= Mus Musculus}}$ (Sequences)

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus

$\pi_{\text{Acc\#, Definition}}$ (Sequences)

Acc#	Definition
CAM22514	Hemoglobin alpha
CAO91797	Hemoglobin X

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra - Natural Join

Sequences

AccNum	Definition	SourceID
CAM22514	Hemoglobin alpha	1
CAO91797	Hemoglobin X	1
ABG47031	Hemoglobin	2

Organisms

SourceID	Name
1	Mus Musculus
1	Mus Musculus
2	Homo Sapiens

Sequences $\bowtie$ Organisms

AccNum	Definition	SourceID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
CAO91797	Hemoglobin X	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens

Requires attributes for join to have same name

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra - Inner Join

Sequences

AccNum	Definition	SrcID
CAM22514	Hemoglobin alpha	1
ABG47031	Hemoglobin	2
NPO34535	Hemoglobin zeta	3

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens

$$\text{Sequences} \bowtie_{\text{sequences.SrcID = organisms.SrcID}} \text{Organisms}$$

AccNum	Name	SrcID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens

Rows without matching attributes excluded

Can name attribute explicitly

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Relational Algebra - Left Join

Sequences

AccNum	Definition	SrcID
CAM22514	Hemoglobin alpha	1
ABG47031	Hemoglobin	2
NPO34535	Hemoglobin zeta	3

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens

$\text{Sequences} \bowtie_{\text{sequences.SrcID = organisms.SrcID}} \text{Organisms}$

AccNum	Name	SrcID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens
NPO34535	Hemoglobin zeta	3	NULL

Tuples (rows) on left table without matches have NULL values on attributes of right table

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select - Projection

- Selects a subset of attributes or columns

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

$\pi_{\text{Acc\#, Source}}(\text{Sequences})$

Acc#	Source
CAM22514	Mus Musculus
CAO91797	Mus Musculus
ABG47031	Homo Sapiens

SELECT Acc#, Source
FROM Sequences

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select - Selection

- Selects a subset of entities or rows

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

$\sigma_{\text{Source=Mus Musculus}}$ (Sequences)

```
SELECT *
  FROM Sequences
   WHERE Source= Mus Musculus
```

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus

"*" means "all" attributes

No reason to keep organism is result

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select: Projection and Selection

$\pi_{\text{Acc\#, Definition}}(\sigma_{\text{Source= Mus Musculus}}(\text{Sequences}))$

Sequences

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus
ABG47031	Hemoglobin	Homo Sapiens

```
SELECT SeqNum,  Org
   FROM Sequences
    WHERE Source= Mus Musculus)
```

$\sigma_{\text{Source= Mus Musculus}}$ (Sequences)

Acc#	Definition	Source
CAM22514	Hemoglobin alpha	Mus Musculus
CAO91797	Hemoglobin X	Mus Musculus

$\pi_{\text{Acc\#, Definition}}$ (Sequences)

Acc#	Definition
CAM22514	Hemoglobin alpha
CAO91797	Hemoglobin X

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select - Natural Join

Sequences

AccNum	Definition	SourceID
CAM22514	Hemoglobin alpha	1
CAO91797	Hemoglobin X	1
ABG47031	Hemoglobin	2

Organisms

SourceID	Name
1	Mus Musculus
1	Mus Musculus
2	Homo Sapiens

Sequences $\bowtie$ Organisms

```
SELECT *
   FROM Sequences
   NATURAL JOIN Organisms
```

AccNum	Definition	SourceID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
CAO91797	Hemoglobin X	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select - Inner Join

Sequences

AccNum	Definition	SrcID
CAM22514	Hemoglobin alpha	1
ABG47031	Hemoglobin	2
NPO34535	Hemoglobin zeta	3

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens

$\text{Sequences} \bowtie_{\text{sequences.SrcID = organisms.SrcID}} \text{Organisms}$

```
SELECT *
    FROM Sequences
    INNER JOIN Organisms
    on sequences.SrcID = organisms.SrcID
```

AccNum	Name	SrcID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens

Rows without matching attributes excluded

Can name attribute explicitly

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select - Left Join

Sequences

AccNum	Definition	SrcID
CAM22514	Hemoglobin alpha	1
ABG47031	Hemoglobin	2
NPO34535	Hemoglobin zeta	3

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens

Sequences $\bowtie_{\text{sequences.SrcID = organisms.SrcID}}$ Organisms

```
SELECT *
   FROM Sequences LEFT OUTER JOIN Organisms
      ON Sequences.SrcID = Organisms.SrcID;
```

AccNum	Name	SrcID	Name
CAM22514	Hemoglobin alpha	1	Mus Musculus
ABG47031	Hemoglobin	2	Homo Sapiens
NPO34535	Hemoglobin zeta	3	NULL

Tuples (rows) on left table without matches have NULL values on attributes of right table

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL UPDATE Statement

Matches

Acc#	Date	Matrix	eValue
P14555	7/21/07	Pam70	4.18 E-32
P81479	7/21/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

```
UPDATE Matches
        SET Date = '7/22/07'
       WHERE Date = '7/21/07'
```

Matches

Acc#	Date	Matrix	eValue
P14555	7/22/07	Pam70	4.18 E-32
P81479	7/22/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL DELETE Statement

Matches

Acc#	Date	Matrix	eValue
P14555	7/21/07	Pam70	4.18 E-32
P81479	7/21/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

```
DELETE FROM Matches
       WHERE Date = '7/21/07'
```

Matches

Acc#	Date	Matrix	eValue
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

ADVICE: BE CAREFUL WITH DELETE. THERE IS NO EASY UNDO

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Select with SORT BY

Matches

Acc#	Date	Matrix	eValue
P14555	7/21/07	Pam70	4.18 E-32
P81479	7/21/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P81479	7/20/07	Blosom80	1.20 E-54

```
SELECT * FROM Matches
         SORT BY eValue ASC
```

Matches

Acc#	Date	Matrix	eValue
P81479	7/20/07	Blosom80	1.20 E-54
P81479	7/22/07	Pam70	2.68 -E52
P14555	7/20/07	Blosom80	3.47 E-33
P14555	7/22/07	Pam70	4.18 E-32

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

SQL Data Manipulation

- Grouping Results and Aggregates

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Filling-Up the database - Insert

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens

```
INSERT INTO Organisms
        Values (3,'C. Elegans')
```

Organisms

SrcID	Name
1	Mus Musculus
2	Homo Sapiens
3	C. Elegans

Good when you know the attributes of all entities c-priori

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Filling-Up the database

- Approach #2: Input from file (CSV)

- Under construction

- Tricky to set data types handled correctly

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study: Analyzing Results of Multiple BLAST Runs with alternative search matrices

- Steps at a Glance
  - Install and configure tools
    - Python environment including BioPython
    - Database system (SQLite)
  - Design the relational database schema
  - Write Python functions to insert results into database
  - Analyze data using SQL queries

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 1): Install Tools

Windows version

- Install Python interpreter
  - Download installer from www.python.org
  - Run installer and verify installation
- Install BioPython
  - Download installer from www.biopython.org
  - Run installer and verify installation
- Install SQLiteMan
  - Download SQLiteMan query browser from www.sqliteman.com
  - Run installer and verify installation

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 1): Install Tools

Mac OS version

- UNDER CONSTRUCTION

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 2):Design Database Schema

Source	Accession Num	Definition	Matrix	eValue	SearchDate
Human	P14555	Group IIA Phospholipase A2	Pam70	4.18 E-32	7/21/07
Indian Green Tree Viper	P81479	Phospholipase A2 isozyme IV	Pam70	2.68 -E52	7/21/07
Human	P14555	Group IIA Phospholipase A2	Blosom80	3.47 E-33	7/20/07
Indian Green Tree Viper	P81479	Phospholipase A2 isozyme IV	Blosom80	1.20 E-54	7/20/07

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Write Python functions to:
  - Run a BLAST search for the query sequence and a specific matrix
  - Insert results into Database
    - Run SQL insert queries to populate the database
    - Create a CSV file with all results from BLAST searches and import to DB
  - Run all functions together running one Blast for each one of a set of matrices

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Run a BLAST search for the query sequence and a specific matrix

```
From Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML
from Bio import Fasta
from sys import *
import sqlite3

def searchAndStoreBlastSearch(query, matrix_id, filename):
    # Creates handle to store the results sent by NCBI website
    results_handle = NCBIWWW.qblast("blastp", "swissprot", query, expect=10,
                descriptions=2000, alignments=2000, hitlist_size=2000,
                matrix_name=matrix_id)
    # Reads results into memory
    blast_results = results_handle.read()

    # Creates and opens a file in filesystem for writing
    save_file = open(filename, 'w')

    # Store results from memory into the filesystem
    save_file.write(blast_results)

    #Close the file handler
    save_file.close()
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Insert results into Database: Run multiple SQL insert queries

```
def xmlToDatabase(filename, matrix_id, dbCursor, dbConn):
    blast_fileptr=open(filename)
    record=NCBIXML.parse(blast_fileptr).next()

    for alignment in record.alignments:
        hsp=alignment.hsps[0]
        storeIntoDatabase(hsp, alignment, matrix_id,dbCursor,dbConn)
    blast_fileptr.close()
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Insert results into Database: Run multiple SQL *insert* queries

```
def storeIntoDatabase(hsp, alignment, matrix_id, dbCursor, dbConn):
    # Insert a row of data into the table (securely)
    t = (hsp.expect, hsp.score, matrix_id, alignment.accession, alignment.title,)
    dbCursor.execute("insert into sequences values (?,?,?,?,?)",t)

    # Save (commit) the changes
    dbConn.commit()
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Insert results into Database: Create a CSV file with all results from BLAST searches

```
def xmlToCSV(xmlFilename, csvFilename, matrix_id):
    # Create file handler
    blast_fileptr=open(xmlFilename)
    # Parse xml file into biopython object
    record=NCBIXML.parse(blast_fileptr).next()
    # Create csv file writer
    csvFileWriter = csv.writer(open(csvFilename,'wb'))
    # Iterate over record alignments
    for alignment in record.alignments:
        # Extract high score pairwise alignment object
        hsp=alignment.hsps[0]
        # Create record vector
        record = [hsp.expect, hsp.score, matrix_id, alignment.accession, alignment.title,]
        # Store record vector into csv file
        csvFileWriter.writerow( record )
    # Close file handler
    blast_fileptr.close()
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Use all functions to run one Blast for each of a sequence of search matrices

```
Matrixname=['PAM70','BLOSUM80']
FileName=['TestPAM70.xml','TestBLOSUM80.xml']
dbFileName='data.db3'
```

```
def doIt():
    print "Import fasta file"
    query_file = open('P00624.fasta')
    fasta = Fasta.Iterator(query_file)
    query = fasta.next()
    query_file.close()
    print "Creating database interface"
    dbConnection = sqlite3.connect(dbFileName)
    dbCursor = dbConnection.cursor()

    # Create tables
    dbCursor.execute("DROP TABLE IF EXISTS sequences")

    dbCursor.execute("CREATE TABLE IF NOT EXISTS sequences (expect real, score int, matrix text, \
                                                            accession text, description text)")
    print "Iterate over matrixes"
    for i in range(0,len(MatrixName)):
        print "Sending blast search"
        searchAndStoreBlastSearch(query, MatrixName[i], FileName[i])
        print "Processing blase search"
        xmlToDatabase(FileName[i], MatrixName[i], dbCursor, dbConnection)
    print "Program done"
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study Step 3: Python Programming

- Run all functions together running one Blast for each one of a set of matrices
- Import CSV version

```
MatrixName=['PAM70','BLOSUM80']
TableName=['PAM70','BLOSUM80']
FileName=['TestPAM70.xml','TestBLOSUM80.xml']
CsvFileName=['TestPAM70.csv','TestBLOSUM80.csv']

def doIt():
    query_file = open('P00624.fasta')
    fasta = Fasta.Iterator(query_file)
    query = fasta.next()
    query_file.close()
    for i in range(0,len(MatrixName)):
        searchAndStoreBlastSearch(query, MatrixName[i], FileName[i])
        xmlToCSV(FileName[i], CsvFileName[i], MatrixName[i])
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 3): Import CSV File

- UNDER CONSTRUCTION

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

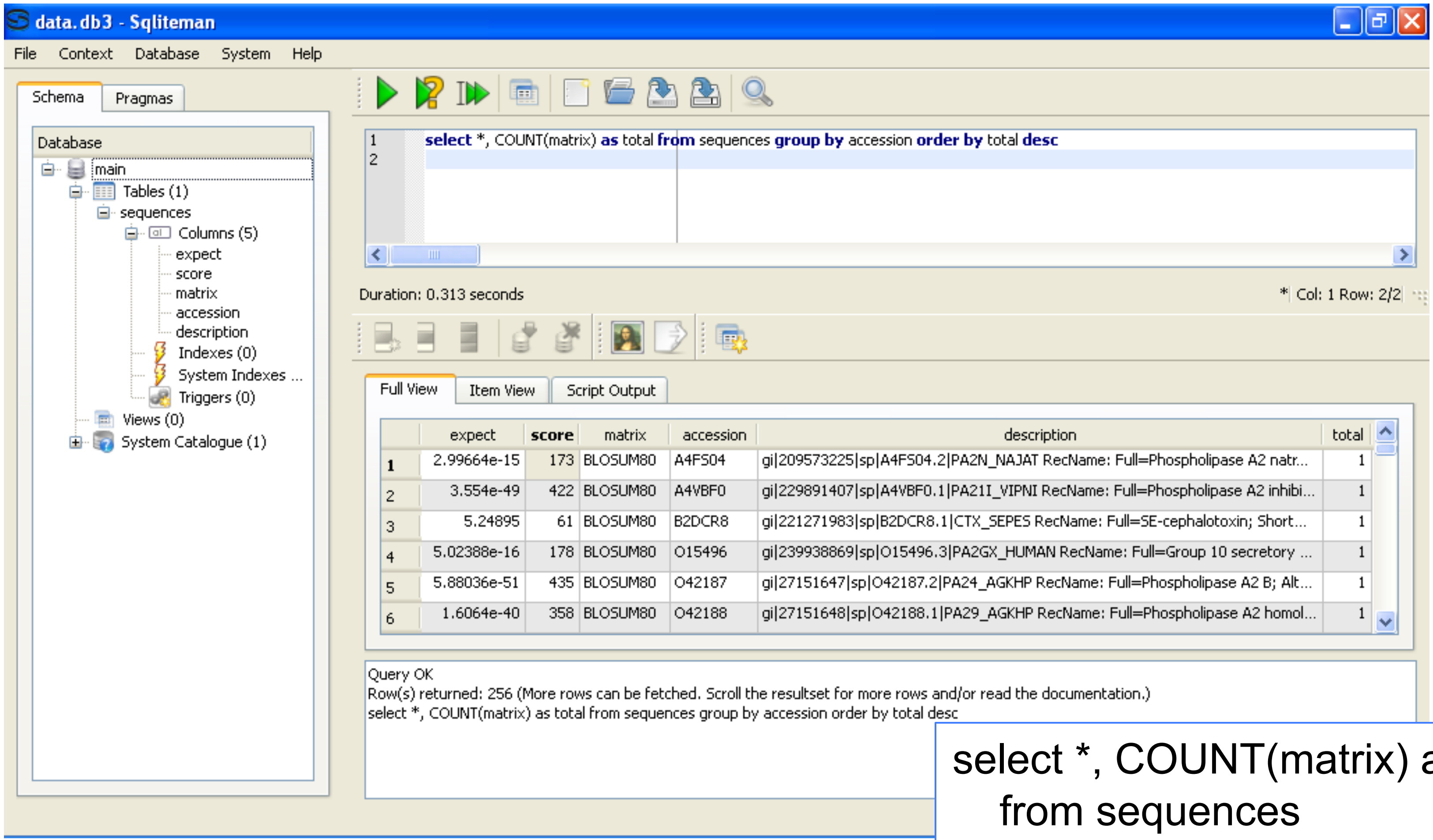
Case Study (Step 4): Analyze Data

- Analyze data using SQL
- Example 1: Finding top matches
  - Display sequences sorted by average score/run and number of runs where they appeared

```
select *, COUNT(matrix) as total
    from sequences
    group by accession
    order by total desc
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 4): Analyze Data



```
select *, COUNT(matrix) as total from sequences group by accession order by total desc
```

Duration: 0.313 seconds

* Col: 1 Row: 2/2

	expect	score	matrix	accession	description	total
1	2.99664e-15	173	BLOSUM80	A4FS04	gi|209573225|sp|A4FS04.2|PA2N_NAJAT RecName: Full=Phospholipase A2 natr...	1
2	3.554e-49	422	BLOSUM80	A4VBF0	gi|229891407|sp|A4VBF0.1|PA21I_VIPNI RecName: Full=Phospholipase A2 inhibi...	1
3	5.24895	61	BLOSUM80	B2DCR8	gi|221271983|sp|B2DCR8.1|CTX_SEPES RecName: Full=SE-cephalotoxin; Short...	1
4	5.02388e-16	178	BLOSUM80	O15496	gi|239938869|sp|O15496.3|PA2GX_HUMAN RecName: Full=Group 10 secretory ...	1
5	5.88036e-51	435	BLOSUM80	O42187	gi|27151647|sp|O42187.2|PA24_AGKHP RecName: Full=Phospholipase A2 B; Alt...	1
6	1.6064e-40	358	BLOSUM80	O42188	gi|27151648|sp|O42188.1|PA29_AGKHP RecName: Full=Phospholipase A2 homol...	1

```
Query OK
Row(s) returned: 256 (More rows can be fetched. Scroll the resultset for more rows and/or read the documentation.)
select *, COUNT(matrix) as total from sequences group by accession order by total desc
```

```
select *, COUNT(matrix) as total
    from sequences
    group by accession
    order by total desc
```

Display sequences sorted by average score/run and number of runs where they appeared

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

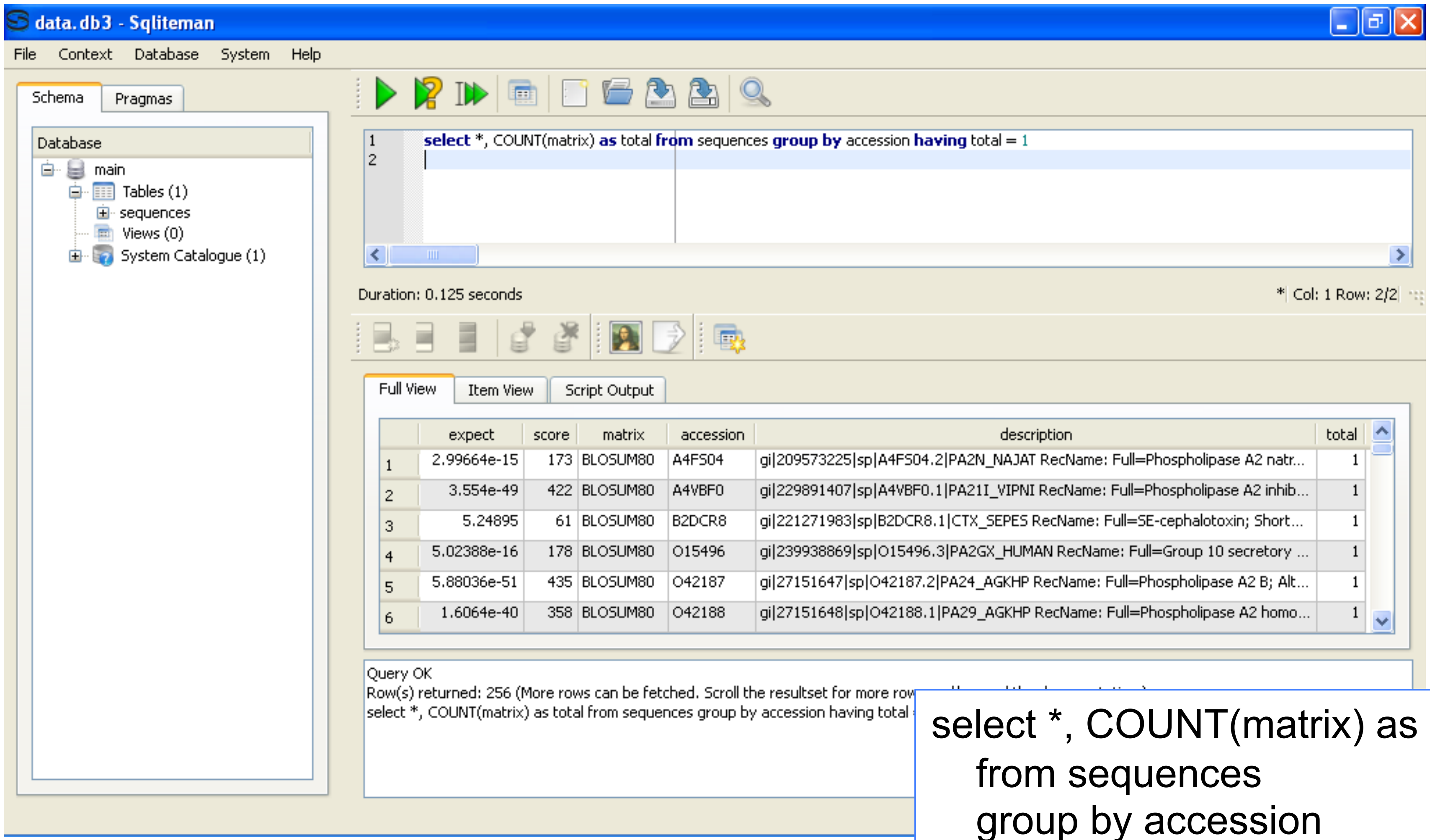
Case Study (Step 4): Analyze Data

- Analyze data using SQL
- Example 2: Finding discriminating matrices
  - Display sequences that only showed up in one matrix run

```
select *, COUNT(matrix) as total
    from sequences
    group by accession
    having total = 1
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study (Step 4): Analyze Data



```
select *, COUNT(matrix) as total from sequences group by accession having total = 1
```

Duration: 0.125 seconds   * Col: 1 Row: 2/2

	expect	score	matrix	accession	description	total
1	2.99664e-15	173	BLOSUM80	A4FS04	gi|209573225|sp|A4FS04.2|PA2N_NAJAT RecName: Full=Phospholipase A2 natr...	1
2	3.554e-49	422	BLOSUM80	A4VBF0	gi|229891407|sp|A4VBF0.1|PA21I_VIPNI RecName: Full=Phospholipase A2 inhib...	1
3	5.24895	61	BLOSUM80	B2DCR8	gi|221271983|sp|B2DCR8.1|CTX_SEPES RecName: Full=SE-cephalotoxin; Short...	1
4	5.02388e-16	178	BLOSUM80	O15496	gi|239938869|sp|O15496.3|PA2GX_HUMAN RecName: Full=Group 10 secretory ...	1
5	5.88036e-51	435	BLOSUM80	O42187	gi|27151647|sp|O42187.2|PA24_AGKHP RecName: Full=Phospholipase A2 B; Alt...	1
6	1.6064e-40	358	BLOSUM80	O42188	gi|27151648|sp|O42188.1|PA29_AGKHP RecName: Full=Phospholipase A2 homo...	1

```
Query OK
Row(s) returned: 256 (More rows can be fetched. Scroll the resultset for more row
select *, COUNT(matrix) as total from sequences group by accession having total
```

Display sequences that only showed up in one matrix run

```
select *, COUNT(matrix) as total
    from sequences
    group by accession
    having total = 1
```

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center

Case Study: Step 4

- Generate Graphic Reports using Excel

These materials were developed with funding from the US National Institutes of Health grant #2T36 GM008789 to the Pittsburgh Supercomputing Center