Databases 101

Matthew J. Graham
CACR

Methods of Computational Science
Caltech, 3 May 2011
DIGITAL FISH
Center of Excellence in Genomic Science

- FlipTrap Database
- Digital Fish Atlas

<table>
<thead>
<tr>
<th>Overview</th>
<th>Technology</th>
<th>Resources</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEGS</td>
<td>FlipTrap</td>
<td>FlipTrap Database</td>
<td>Molecular Instruments</td>
</tr>
<tr>
<td>People</td>
<td>HCR</td>
<td>Digital Fish Atlas</td>
<td>NUPACK</td>
</tr>
<tr>
<td></td>
<td>Light Sheet Microscopy</td>
<td></td>
<td>Zfin</td>
</tr>
<tr>
<td>Overview</td>
<td>Technology</td>
<td>Resources</td>
<td>Links</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------</td>
<td>----------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>CEGS</td>
<td>FlipTrap</td>
<td>FlipTrap Database</td>
<td>Molecular Instruments</td>
</tr>
<tr>
<td>People</td>
<td>HCR</td>
<td></td>
<td>NUPACK</td>
</tr>
<tr>
<td></td>
<td>Light Sheet Microscopy</td>
<td>Digital Fish Atlas</td>
<td>Zfin</td>
</tr>
<tr>
<td>name</td>
<td>description</td>
<td>ensemble id</td>
<td>alleles</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>syn2b</td>
<td>synapsin lb</td>
<td>ENSDARG00000078971</td>
<td>ctt22c</td>
</tr>
<tr>
<td>seratd2</td>
<td>SERTA domain containing 2</td>
<td>ENSDARG00000005530</td>
<td>ctt22a</td>
</tr>
<tr>
<td>ab215b</td>
<td>aB11binteractor 1b</td>
<td>ctt44a</td>
<td>hindbrain</td>
</tr>
<tr>
<td>prkca</td>
<td>protein kinase C, alpha</td>
<td>ENSDARG00000093244</td>
<td>ctt24a</td>
</tr>
<tr>
<td>rap1a</td>
<td>RAS related protein 1b</td>
<td>ENSDARG00000009867</td>
<td>ctt25a</td>
</tr>
<tr>
<td>rns3</td>
<td>RNA binding motif, single stranded interacting protein</td>
<td>ENSDARG00000046924</td>
<td>ctt26a</td>
</tr>
<tr>
<td>syn2b</td>
<td>synapsin lb</td>
<td>ENSDARG00000079871</td>
<td>ctt27b</td>
</tr>
<tr>
<td>prkca</td>
<td>protein kinase C, alpha</td>
<td>ENSDARG00000093244</td>
<td>ctt28a</td>
</tr>
<tr>
<td>aloc1b</td>
<td>aloc 1b</td>
<td>ENSDARG00000057661</td>
<td>ctt29a</td>
</tr>
<tr>
<td>pnp27h</td>
<td>PNP27 periodic cryptophycean protein homolog (yeast)</td>
<td>ENSDARG00000037109</td>
<td>ctt30a</td>
</tr>
<tr>
<td>casl1a</td>
<td>calmodulin kinase 1, delta a</td>
<td>ENSDARG00000083370</td>
<td>ctt31a</td>
</tr>
<tr>
<td>ctt32a</td>
<td>cerebellum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nps1996</td>
<td>nps2896</td>
<td>ENSDARG00000078586</td>
<td>ctt33b</td>
</tr>
<tr>
<td>pepl1</td>
<td>poly(A) binding protein interacting protein 1</td>
<td>ENSDARG00000042882</td>
<td>ctt34a</td>
</tr>
<tr>
<td>magt1</td>
<td>magnesium transporter 1</td>
<td>ENSDARG00000086085</td>
<td>ctt35a</td>
</tr>
<tr>
<td>slt7</td>
<td>SLT Phiping factor homolog (S. cerevisiae)</td>
<td>ENSDARG00000038870</td>
<td>ctt36a</td>
</tr>
<tr>
<td>idp42pbo</td>
<td>IDP42 binding protein kinase beta (DMPK-like)</td>
<td>ENSDARG00000019363</td>
<td>ctt37a</td>
</tr>
<tr>
<td>hcm3</td>
<td>nucleosidephosphonucleosmin, 3</td>
<td>ENSDARG00000067984</td>
<td>ctt38a</td>
</tr>
</tbody>
</table>
what is a database?

A structured collection of data residing on a computer system that can be easily accessed, managed and updated

Data is organised according to a database model

A Database Management System (DBMS) is a software package designed to store and manage databases
why use a dbms?

- data independence
- efficient and concurrent access
- data integrity, security and safety
- uniform data administration
- reduced application development time
- data analysis tools
scale of databases

"DBs own the sweet spot of 1GB to 100TB" (Gray & Hey, 2006)

SQLite

MySQL, PostgreSQL

SQLServer, Oracle

*Hive, HadoopDB
data models

A collection of concepts describing how structured data is represented and accessed.

Within a data model, the **schema** is a set of descriptions of a particular collection of data.

The schema is stored in a **data dictionary** and can be represented in SQL, XML, RDF, etc.

In semantics a data model is equivalent to an ontology - "a formal, explicit specification of a shared conceptualisation"
flat (file) model

- Data files that contain records with no structural relationships
- Additional information is required to interpret these files such as the file format properties
- Hollerith 1889 patent "Art of Compiling Statistics" describes how every US resident can be represented by a string of 80 characters and numbers
- Examples: delimited-separated data, HTML table
Data is organized as relations, attributes and domains

A relation is a table with columns (attributes) and rows (tuples)

The domain is the set of values that the attributes are allowed to take

Within the relation, each row is unique, the column order is immaterial and each row contains a single value for each of its attributes

Proposed by E. F. Codd in 1969/70
An atomic sequence of actions (read/write) in the database

Each transaction has to be executed **completely** and must leave the database in a consistent state

If the transaction fails or aborts midway, the database is "rolled back" to its initial consistent state

**Example:**
Authorise Paypal to pay $100 for my eBay purchase:
- Debit my account $100
- Credit the seller's account $100
By definition, a database transaction must be:

- **Atomic**: all or nothing
- **Consistent**: no integrity constraints violated
- **Isolated**: does not interfere with any other transaction
- **Durable**: committed transaction effects persist
DBMS ensures that interleaved transactions coming from different clients do not cause inconsistencies in the data.

It converts the concurrent transaction set into a new set that can be executed sequentially.

Before reading/writing an object, each transaction waits for a **lock** on the object.

Each transaction releases all its locks when finished.
DMBS can set and hold multiple locks simultaneously on different levels of the physical data structure.

Granularity: at a row level, page (a basic data block), extent (multiple array of pages) or even an entire table.

Exclusive vs. shared

Optimistic vs. pessimistic
Ensures atomicity of transactions

Recovering after a crash, effects of partially executed transactions are undone using the log

Log record:

-- Header (transaction ID, timestamp, ...)
-- Item ID
-- Type
-- Old and new value
partitions

- Horizontal: different rows in different tables
- Vertical: different columns in different tables (normalisation)
- Range: rows where values in a particular column are inside a certain range
- List: rows where values in a particular column match a list of values
- Hash: rows where a hash function returns a particular value
Structured Query Language

- Appeared in 1974 from IBM
- First standard published in 1986; most recent in 2008
- SQL92 is taken to be default standard
- Different flavours:
  - Microsoft/Sybase: Transact-SQL
  - MySQL: MySQL
  - Oracle: PL/SQL
  - PostgreSQL: PL/pgSQL
CREATE DATABASE databaseName
CREATE TABLE tableName (name1 type1, name2 type2, ...)

CREATE TABLE star (name varchar(20), ra float, dec float, vmag float)

Data types:
- boolean, bit, tinyint, smallint, int, bigint;
- real/float, double, decimal;
- char, varchar, text, binary, blob, longblob;
- date, time, datetime, timestamp

CREATE TABLE star (name varchar(20) not null, ra float default 0, ...)
CREATE TABLE star (name varchar(20), ra float, dec float, vmag float,
  CONSTRAINT PRIMARY KEY (name))

A primary key is a unique identifier for a row and is automatically not null

CREATE TABLE star (name varchar(20), ..., stellarType varchar(8),
  CONSTRAINT stellarType_fk FOREIGN KEY (stellarType)
  REFERENCES stellarTypes(id))

A foreign key is a referential constraint between two tables identifying a column in one table that refers to a column in another table.
INSERT INTO `tableName` VALUES(val1, val2, ...)

INSERT INTO `star` VALUES('Sirius', 101.287, -16.716, -1.47)

INSERT INTO `star`(name, vmag) VALUES('Canopus', -0.72)

INSERT INTO `star` SELECT ...
DELETE FROM tableName WHERE condition
TRUNCATE TABLE tableName
DROP TABLE tableName

DELETE FROM star WHERE name = 'Canopus'

DELETE FROM star WHERE name LIKE 'C_n%'

DELETE FROM star WHERE vmag > 0 OR dec < 0

DELETE FROM star WHERE vmag BETWEEN 0 and 5
UPDATE tableName SET columnName = val1 WHERE condition

UPDATE star SET vmag = vmag + 0.5

UPDATE star SET vmag = -1.47 WHERE name LIKE 'Sirius'
SELECT `selectList` FROM `tableList` WHERE `condition`  
ORDER BY `criteria`

SELECT `name`, `constellation` FROM `star` WHERE `dec` > 0  
ORDER BY `vmag`

SELECT * FROM `star` WHERE `ra` BETWEEN 0 AND 90

SELECT DISTINCT `constellation` FROM `star`

SELECT `name` FROM `star` LIMIT 5  
ORDER BY `vmag`

matthew graham
Inner join: combining related rows

SELECT * FROM star s INNER JOIN stellarTypes t ON s.stellarType = t.id

SELECT * FROM star s, stellarTypes t WHERE s.stellarType = t.id

Outer join: each row does not need a matching row

SELECT * FROM star s LEFT OUTER JOIN stellarTypes t ON s.stellarType = t.id

SELECT * FROM star s RIGHT OUTER JOIN stellarTypes t ON s.stellarType = t.id

SELECT * FROM star s FULL OUTER JOIN stellarTypes t ON s.stellarType = t.id
aggregate functions

COUNT, AVG, MIN, MAX, SUM

SELECT COUNT(*) FROM star

SELECT AVG(vmag) FROM star

SELECT stellarType, MIN(vmag), MAX(vmag) FROM star
    GROUP BY stellarType

SELECT stellarType, AVG(vmag), COUNT(id) FROM star
    GROUP BY stellarType
    HAVING vmag > 14
CREATE VIEW viewName AS ...

CREATE VIEW region1View AS
  SELECT * FROM star WHERE ra BETWEEN 150 AND 170
  AND dec BETWEEN -10 AND 10

SELECT id FROM region1View WHERE vmag < 10

CREATE VIEW region2View AS
  SELECT * FROM star s, stellarTypes t WHERE s.stellarType = t.id
  AND ra BETWEEN 150 AND 170 AND dec BETWEEN -10 AND 10

SELECT id FROM regionView2 WHERE vmag < 10 and stellarType LIKE 'A%'

matthew graham
A clustered index is one in which the ordering of data entries is the same as the ordering of data records.

Only one clustered index per table but multiple unclustered indexes.

Typically implemented as B+ trees but alternate types such as bitmap index for high frequency repeated data.
DECLARE cursorName CURSOR FOR SELECT ...
OPEN cursorName
FETCH cursorName INTO ...
CLOSE cursorName

A cursor is a control structure for successive traversal of records in a result set

Slowest way of accessing data
For each row in the result set, update the relevant stellar model

DECLARE @name varchar(20)
DECLARE @mag float
DECLARE starCursor CURSOR FOR
  SELECT name, AVG(vmag) FROM star
  GROUP BY stellarType
OPEN starCursor
  FETCH starCursor INTO @name, @mag
  EXEC updateStellarModel @name, @mag / CALL updateStellarModel(@name, @mag)
CLOSE starCursor

matthew graham
CREATE TRIGGER triggerName ON tableName ...

A trigger is procedural code that is automatically executed in response to certain events on a particular table:

- INSERT
- UPDATE
- DELETE

CREATE TRIGGER starTrigger ON star FOR UPDATE AS
    IF @@ROWCOUNT = 0 RETURN
    IF UPDATE (vmag) EXEC refreshModels
GO
CREATE PROCEDURE findNearestNeighbour @starName varchar(20) AS
BEGIN
    DECLARE @ra, @dec float
    DECLARE @name varchar(20)
    SELECT @ra = ra, @dec = dec FROM star WHERE name LIKE @starName
    SELECT name FROM getNearestNeighbour(@ra, @dec)
END

EXEC findNearestNeighbour 'Sirius'
First normal form: no repeating elements or groups of elements table has a unique key (and no nullable columns)

Second normal form: no columns dependent on only part of the key

Star Name | Constellation | Area

Third normal form: no columns dependent on other non-key columns

Star Name | Magnitude | Flux
import java.sql.*
...
String dbURL = "jdbc:mysql://127.0.0.1:1234/test";
Connection conn = DriverManager.getConnection(dbUrl, "mjg", "mjg");
Statement stmt = conn.createStatement();
ResultSet res = stmt.executeQuery("SELECT * FROM star");
...
conn.close();
import MySQLdb
Con = MySQLdb.connect(host="127.0.0.1", port=1234, user="mjg",
                      passwd="mjg", db="test")
Cursor = Con.cursor()
sql = "SELECT * FROM star"
Cursor.execute(sql)
Results = Cursor.fetchall()
...
Con.close()