Databases 101

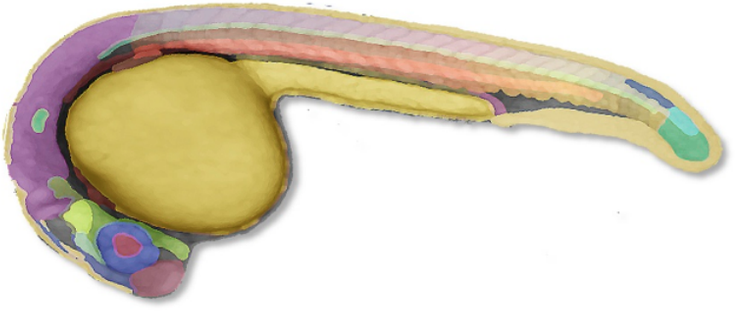
Matthew J. Graham
CACR

Methods of Computational Science
Caltech, 3 May 2011

matthew graham

Digital Fish

Center of Excellence in Genomic Science



- FlipTrap Database
- Digital Fish Atlas

Overview	Technology	Resources	Links
CEGS People	FlipTrap HCR Light Sheet Microscopy	FlipTrap Database Digital Fish Atlas	Molecular Instruments NUPACK Zfin

BROWSE BY:

- Gene Name
- Anatomy/Region
- Stage
- Allele
- Multiple Parameters

Browse by anatomy/region

hindbrain

☑ Include related terms

Overview	Technology	Resources	Links
CEGS	FlipTrap	FlipTrap Database	Molecular Instruments
People	HCR	Digital Fish Atlas	NUPACK
	Light Sheet Microscopy		Zfin

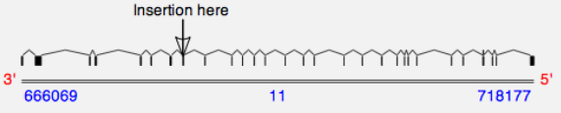
BROWSE BY:
- Gene Name
- Anatomy/Region
- Stage
- Allele
- Multiple Parameters

Search criteria

anatomy: hindbrain and related terms

48 ensembl genes

name	description	ensembl id	alleles	matched term	score ▲	
			ct168a	hindbrain	1.0	view
syn2b	synapsin IIb	ENSDARG00000078971	ct122c	hindbrain	1.0	view
sertad2	SERTA domain containing 2	ENSDARG00000055530	ct75a	hindbrain	1.0	view
			ct74a	hindbrain	1.0	view
abi1b	abl-interactor 1b	ENSDARG00000062991	ct67a, ct79a	hindbrain	1.0	view
prkca	protein kinase C, alpha	ENSDARG00000039241	ct54a	hindbrain	1.0	view
			ct48b	hindbrain	1.0	view
rap1b	RAS related protein 1b	ENSDARG00000008867	ct40a	hindbrain	1.0	view
			ct32a	hindbrain	1.0	view
rbms3	RNA binding motif, single stranded interacting protein	ENSDARG00000044574	ct30a	hindbrain	1.0	view
syn2b	synapsin IIb	ENSDARG00000078971	ct122c	hindbrain commissure	0.875	view
			ct48b	hindbrain commissure	0.875	view
prkca	protein kinase C, alpha	ENSDARG00000039241	ct7a	hindbrain commissure	0.875	view
			ct132a	cerebellum	0.875	view
aldoca	aldolase C, fructose-bisphosphate, a	ENSDARG00000057661	ct126a	cerebellum	0.875	view
pwp2h	PWP2 periodic tryptophan protein homolog (yeast)	ENSDARG00000037109	ct143a	whole organism	0.75	view
csnk1da	casein kinase 1, delta a	ENSDARG00000008370	ct135a	whole organism	0.75	view
			ct132a	whole organism	0.75	view
zgc:65996	zgc:65996	ENSDARG00000057556	ct122b	whole organism	0.75	view
paip1	poly(A) binding protein interacting protein 1	ENSDARG00000042892	ct120a	whole organism	0.75	view
magt1	magnesium transporter 1	ENSDARG00000058062	ct104a	whole organism	0.75	view
slu7	SLU7 splicing factor homolog (S. cerevisiae)	ENSDARG00000038870	ct94a	whole organism	0.75	view
cdc42bpb	CDC42 binding protein kinase beta (DMPK-like)	ENSDARG00000019383	ct89a	whole organism	0.75	view
npm3	nucleophosmin/nucleoplasmin, 3	ENSDARG00000056794	ct86a	whole organism	0.75	view

	NCBI	Ensembl
Gene Name	syn2b	LOC100330853
Aliases	MGC123199 | zgc:123199 | synapsin IIb	zgc:123199
Gene Description	synapsin IIb	synapsin IIb (syn2b), mRNA [Source:RefSeq DNA;Acc:NM_001037576]
Gene Ontology	show	show
NCBI Nucleotide Hit	NM_001037576 show details	
Ensembl Genome Map	 contig view	
Comments		

mRNA sequence

hide

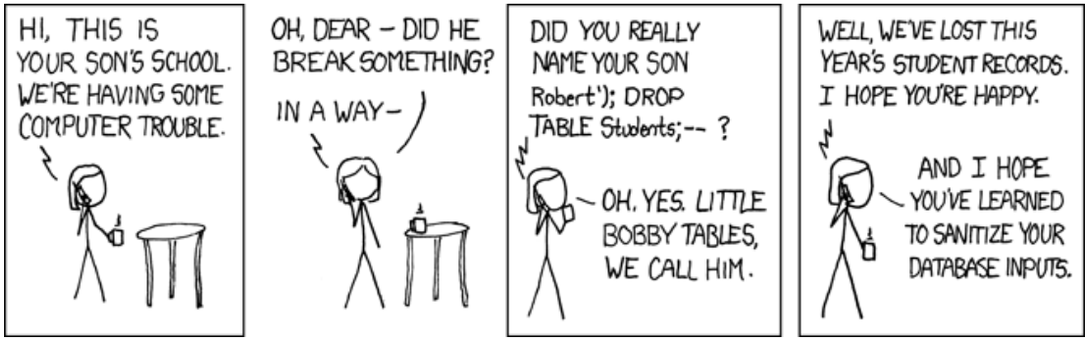
```
TGGCCTATGCATTGGAGCTGTCAGTGCATGAAGACGGAGAAATTCAGCAGCAGCATCAGCACACAGTCTTAAAGCGGA
GATTCCTGGAAAGCCAGAGCTGATTTTCCCCAGTCGCTGACGCTCACCGTTGAAGATCTGAGAGTTTAACCTTCACAG
CCATGAGCTTCCTGCGTCGTCGTCTCTCTGACAGCAGCTTCATCGCTAATCTGCCCAATGGCTACATGTCGGACCTC
CAGCGTCCGGACCCGCCGCCGCCCCCCTCCAGCCGACGCCAAACCCCCATCCGGATCCCCTGCCGCCCCCATTGGCCCCT
GCAGAAAACAGGTCTCCGTCTTCATCTCCGGCTCCAGAGCGGCGTCCGCAGCCGGCTCCATCCAGCGGCTCCGGCTTC
TTCAGCTCCATCACTAACGTAGTCAAGCAGACGGCGGCATCTGCGGGACTCGTGGAGCAAAGCCCTGCCACCGCTTCC
CGGAAGTTCAAGATCCTGCTGGTCATCGACGAACCCCAGAATGATTGGACAAAGCTTTTTCGTGGGAAGAAGGTGCAAG
GAGATTATGACATCAAAGTGGAGCAGGCTGAGTTCAGTGAGATTAATCTGGTGGCTCATGCTAACGGCACGTGCAGTG
TGGACATGCAGGTCATCAGAAACGGCACCAAAGTGGTTCGCTCCTTCAAGCCTGATTTTGTCCTGGTTCGACAGCATG
CCTACAGTATGGCACAGAATGAAGACTTCAGGAACATCATCATCGGCTTACAGTACGCCGGCATTCCCAGCGTCAACT
CACTGGAGTCCATCTACAACCTCTGTGACAAACCCTGGGCTTTCGCTCAGCTGATCAGCGTCTACAAGAAAATGGGTC
CAGAAAAGTTCCCTCTGGTTGATCAGACCTTTTACTCCAACTACAGGGACAT GATCTCAATGCCAACATTCCCAGT
GGTGGTGAAGATTGGACATGCTCATTCTGGGATGGGGAAGGTGAAGGTTGACAATCACAGTGATTTCCAGGACATTGC
AAGCGTTGTGGCCATCACTCAGACATACACCACCACTGAGCCTTTCATTGATGCCAAATACGACATCAGGGTGCAGAA
GATCGGCTTTGATTATAAAGCCTACATGAGAACATCCATCTCAGGCAACTGGAAATCAAACACTGGATCTGCCATGCTG
GTGTTTTAGCGACATCCAGCTTGCATCCAGCACTTATCAACCCGTCCGGATATTCATGAGTGCCATATTTATAGTATC
TCCGCATATTGTATCGTACTGTACGTTAAACATCTCAGGAGTTTGATTTGGGTGTGTGTTTATTCCGGCGTCACTCTG
TTCATGTTGTGTTTTCTTTGTGCGCAGTGATATTGCTTTGCATGACCCGGTGATAATTGTTTTCTCTGTACAAGTGTC
AATAAACATGTGGCGCATCATTCCACTGAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Stage	Image (click thumbnail for full size image)	Comments (click to highlight)

whole embryo | ventral forebrain | head | hindbrain | trunk

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?

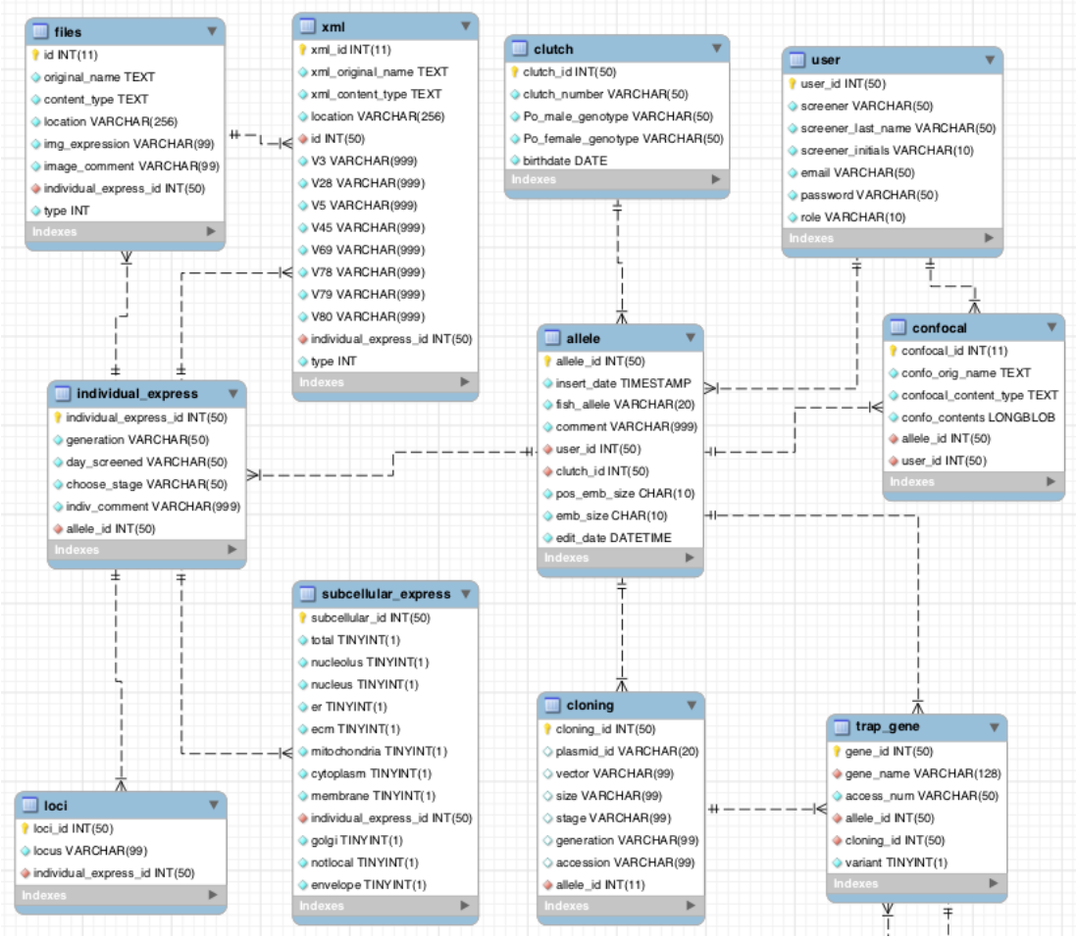
- data independence
- efficient and concurrent access
- data integrity, security and safety
- uniform data administration
- reduced application development time
- data analysis tools

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"


files
id INT(11)
original_name TEXT
content_type TEXT
location VARCHAR(256)
img_expression VARCHAR(99)
image_comment VARCHAR(99)
individual_express_id INT(50)
type INT
Indexes
xml
xml_id INT(11)
xml_original_name TEXT
xml_content_type TEXT
location VARCHAR(256)
id INT(50)
V3 VARCHAR(999)
V28 VARCHAR(999)
V5 VARCHAR(999)
V45 VARCHAR(999)
V69 VARCHAR(999)
V78 VARCHAR(999)
V79 VARCHAR(999)
V80 VARCHAR(999)
individual_express_id INT(50)
type INT
Indexes
clutch
clutch_id INT(50)
clutch_number VARCHAR(50)
Po_male_genotype VARCHAR(50)
Po_female_genotype VARCHAR(50)
birthdate DATE
Indexes
user
user_id INT(50)
screener VARCHAR(50)
screener_last_name VARCHAR(50)
screener_initials VARCHAR(10)
email VARCHAR(50)
password VARCHAR(50)
role VARCHAR(10)
Indexes
individual_express
individual_express_id INT(50)
generation VARCHAR(50)
day_screened VARCHAR(50)
choose_stage VARCHAR(50)
indiv_comment VARCHAR(999)
allele_id INT(50)
Indexes
allele
allele_id INT(50)
insert_date TIMESTAMP
fish_allele VARCHAR(20)
comment VARCHAR(999)
user_id INT(50)
clutch_id INT(50)
pos_emb_size CHAR(10)
emb_size CHAR(10)
edit_date DATETIME
Indexes
confocal
confocal_id INT(11)
confo_orig_name TEXT
confocal_content_type TEXT
confo_contents LONGBLOB
allele_id INT(50)
user_id INT(50)
Indexes
subcellular_express
subcellular_id INT(50)
total TINYINT(1)
nucleolus TINYINT(1)
nucleus TINYINT(1)
er TINYINT(1)
ecm TINYINT(1)
mitochondria TINYINT(1)
cytoplasm TINYINT(1)
membrane TINYINT(1)
individual_express_id INT(50)
golgi TINYINT(1)
notlocal TINYINT(1)
envelope TINYINT(1)
Indexes
loci
loci_id INT(50)
locus VARCHAR(99)
individual_express_id INT(50)
Indexes
cloning
cloning_id INT(50)
plasmid_id VARCHAR(20)
vector VARCHAR(99)
size VARCHAR(99)
stage VARCHAR(99)
generation VARCHAR(99)
accession VARCHAR(99)
allele_id INT(11)
Indexes
trap_gene
gene_id INT(50)
gene_name VARCHAR(128)
access_num VARCHAR(50)
allele_id INT(50)
cloning_id INT(50)
variant TINYINT(1)
Indexes

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

relational model

- 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

transactions

- 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

acid

By definition, a database transaction must be:

- **A**tomic: all or nothing
- **C**onsistent: no integrity constraints violated
- **I**solated: does not interfere with any other transaction
- **D**urable: committed transaction effects persist

concurrency

- 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

locks

- 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

logs

- 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

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, ...)
```

keys

```
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

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

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

SELECT *selectionList* 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
```

joins

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%'
```

indexes

CREATE INDEX *indexName* ON *tableName*(*columns*)

```
CREATE INDEX vmagIndex ON star(vmag)
```

- 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

cursors

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

cursors example

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
```

triggers

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
```

stored procedures

CREATE PROCEDURE *procedureName* @param1 type, ...
AS ...

```
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'
```

normalisation

- 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

Java

```
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();
```

Python:

```
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()
```