Databases

Learning to talk database

Introducing Databases

-
- A database is a collection of one or more related tables
-
- A table is a collection of one or more rows of data
-
- A row is a collection of one or more data items, arranged in columns

Structured Data

```
1960-12-21          P. Barry
1954-6-14           M. Moorhouse

--------------      ---------
Discovery_Date      Scientist
--------------      ---------
1960-12-21          P. Barry
1954-6-14           M. Moorhouse
1970-3-4            J. Blow
2001-12-27          J. Doe

-----------         -----------------
Column name         Type restriction
-----------         -----------------
Discovery_Date      a valid Date
Scientist           a String no longer than 64 characters
```

Relating tables

Discovery_Date	Scientist	Discovery
1960-12-21	P. Barry	Flying car
1954-6-14	M. Moorhouse	Telepathic sunglasses
1970-3-4	J. Blow	Self cleaning child
2001-12-27	J. Doe	Time travel

Column name	Type restriction
Discovery_Date	a valid Date
Scientist	a String no longer than 64 characters
Discovery	a String no longer than 128 characters

Relating tables, cont.

Column name	Type restriction
Discovery_Date	a valid Date
Scientist	a String no longer than 64 characters
Discovery	a String no longer than 128 characters
Date_of_birth	a valid Date
Telephone_number	a String no longer than 16 characters

Discovery_Date	Scientist	Discovery	Date_of_birth	Telephone_number
1960-12-21	P. Barry	Flying car	1966-11-18	353-503-555-91910
1954-6-14	M. Moorhouse	Telepathic sunglasses	1970-3-24	00-44-81-555-3232
1970-3-4	J. Blow	Self cleaning child	1955-8-17	555-2837
2001-12-27	J. Doe	Time travel	1962-12-1	-
1974-3-17	M. Moorhouse	Memory swapping toupee	1970-3-24	00-44-81-555-3232
1999-12-31	M. Moorhouse	Twenty six hour clock	1958-7-12	416-555-2000

Solving the one table problem

Column name	Type restriction
Discovery_Date	a valid Date
Scientist_ID	a String no longer than 8 characters
Discovery	a String no longer than 128 characters

Column name	Type restriction
Scientist_ID	a String no longer than 8 characters
Scientist	a String no longer than 64 characters
Date_of_birth	a valid Date
Address	a String no longer than 256 characters
Telephone_number	a String no longer than 16 characters

Solving the one table problem, cont.

Discovery_Date	Scientist_ID	Discovery
1954-6-14	MM	Telepathic sunglasses
1960-12-21	PB	Flying car
1969-8-1	PB	A cure for bad jokes
1970-3-4	JB	Self cleaning child
1974-3-17	MM	Memory swapping toupee
1999-12-31	MM2	Twenty six hour clock
2001-12-27	JD	Time travel

Scientist_ID	Scientist	Date_of_birth	Address	Telephone_number
JB	J. Blow	1955-8-17	Belfast, NI	555-2837
JD	J. Doe	1962-12-1	Syndey, AUS	-
MM	M. Moorhouse	1970-3-24	England, UK	00-44-81-555-3232
MM2	M. Moorhouse	1958-7-12	Toronto, CA	416-555-2000
PB	P. Barry	1966-11-18	Carlow, IRL	353-503-555-91910

Maxim 12.1

A little database design goes a long way.

Database system: a definition

-
- A database system is a computer program (or group of programs) that provides a mechanism to define and manipulate one or more databases

Available Database Systems

-
- Personal database systems
-
- Enterprise database systems
-
- Open source database systems

SQL: The Language of Databases

-
- Defining data with SQL
-
- Manipulating data with SQL

A Database Case Study: MER

http://www.expasy.org/sprot/userman.html

-

http://www.ebi.ac.uk/embl/Documentation/User_manual/home.html

Extracted Sample Data

```
ID MERT_ACICA STANDARD; PRT; 116 AA.
AC Q52106;
DT 01-NOV-1997 (Rel. 35, Created)
DT 01-NOV-1997 (Rel. 35, Last sequence update)
DT 15-JUN-2002 (Rel. 41, Last annotation update)
DE Mercuric transport protein (Mercury ion transport protein).
GN MERT.
OS Acinetobacter calcoaceticus.
OG Plasmid pKLH2.
OC Bacteria; Proteobacteria; Gammaproteobacteria;
   Pseudomonadales;
OC Moraxellaceae; Acinetobacter.

    .
    .
    .
```

Installing a database system

http://www.mysql.com

```
$ chkconfig --add mysqld
$ chkconfig mysqld on

$ mysqladmin -u root password 'passwordhere'

$ mysql -u root -p
```

Creating the MER database

```
mysql> create database MER;
Query OK, 1 row affected (0.36 sec)

mysql> show databases;
+------------+
| Databases  |
+------------+
| MER        |
| test       |
| mysql      |
+------------+
3 rows in set (0.00 sec)

mysql> use mysql;
Database changed

mysql> grant all on MER.* to bbp identified by 'passwordhere';
Query OK. 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Adding tables to the MER database

```
create table proteins
(
    accession_number   varchar (6) not null,
    code               varchar (4) not null,
    species            varchar (5) not null,
    last_date          date not null,
    description        text not null,
    sequence_header    varchar (75) not null,
    sequence_length    int not null,
    sequence_data      text not null
)
```

```
$ mysql -u bbp -p MER < create_proteins.sql
```

Maxim 12.2

Understand the data before designing the tables

Example SWISS-PROT data-files

acica_ADPT.swp.txt

serma_abdprt.swp.txt

shilf_seq_ACDP.swp.txt

Preparing SWISS-PROT data for importation

```
$ ./get_proteins *swp* > proteins.input
```

Importing tab-delimited data into proteins

```
$ mysql -u bbp -p MER

mysql> load data local infile "proteins.input" into table proteins;

Query OK, 14 rows affected (0.07sec)
Records: 14 Deleted: 0, Skipped: 0, Warnings: 0
```

Working with the data in proteins

```
mysql> select accession_number, sequence_length
-> from proteins;
```

```
mysql> select accession_number, sequence_length
-> from proteins
-> order by accession_number;
```

```
mysql> select accession_number, sequence_length
-> from proteins
-> where sequence_length > 200
-> order by sequence_length;
```

Adding another table to the MER database

```
create table dnas
(
    accession_number    varchar (8) not null,
    entry_name          varchar (9) not null,
    sequence_version    varchar (16) not null,
    last_date           date not null,
    description         text not null,
    sequence_header     varchar (75) not null,
    sequence_length     int not null,
    sequence_data       text not null
)
```

```
$ mysql -u bbp -p MER < create_dnas.sql
```

Preparing EMBL data for importation

Example EMBL data-files

```
AF213017.EMBL.txt

J01730.embl.txt

M15049.embl.txt

M24940.embl.txt
```

```
$ ./get_dnas *EMBL* *embl* > dnas.input
```

Importing tab-delimited data into dnas

```
mysql> load data local infile "dnas.input" into table dnas;

Query OK, 4 rows affected (0.01sec)
Records: 4 Deleted: 0, Skipped: 0, Warnings: 0
```

Working with the data in dnas

```
mysql> select accession_number, sequence_length
-> from dnas
-> where sequence_length > 4000
-> order by sequence_length;
```

```
mysql> select accession_number, sequence_length
-> from dnas
-> where sequence_length > 4000
-> order by sequence_length desc;
```

```
mysql> select accession_number, entry_name, sequence_length
-> from dnas
-> order by sequence_length desc
-> limit 1;
```

Adding the crossrefs table to the MER database

```
create table crossrefs (
    ac_protein      varchar (6) not null,
    ac_dna          varchar (8) not null
)
```

```
$ mysql -u bbp -p MER < create_crossrefs.sql
```

Preparing cross-references for importation

```
$ ./get_protein_crossrefs *swp* > protein_crossrefs

$ ./get_dna_crossrefs *embl* *EMBL* > dna_crossrefs

$ ./unique_crossrefs protein_crossrefs dna_crossrefs > unique.input
```

Importing tab-delimited data into crossrefs

```
mysql> load data local infile "unique.input" into table crossrefs;
Query OK, 22 rows affected (0.04 sec)
Records: 22 Deleted: 0 Skipped: 0 Warnings: 0
```

Working with the data in crossrefs

```
mysql> select * from crossrefs;
```

```
mysql> select proteins.sequence_header, dnas.sequence_header
-> from proteins, dnas, crossrefs
-> where proteins.accession_number = crossrefs.ac_protein
-> and dnas.accession_number = crossrefs.ac_dna
-> order by proteins.sequence_header;
```

```
mysql> select proteins.code, proteins.species, dnas.entry_name
-> from proteins, dnas, crossrefs
-> where proteins.accession_number = crossrefs.ac_protein
-> and dnas.accession_number = crossrefs.ac_dna;
```

Working with the data in crossrefs, cont.

```
mysql> select
-> proteins.code as 'Protein Code',
-> proteins.species as 'Protein Species',
-> dnas.entry_name as 'DNA Entry Name'
-> from proteins, dnas, crossrefs
-> where proteins.accession_number = crossrefs.ac_protein
-> and dnas.accession_number = crossrefs.ac_dna
-> order by proteins.code;
```

Adding the citations table to the MER database

```
create table citations (
    accession_number        varchar (8) not null,
    number                  int not null,
    author                  text not null,
    title                   text not null,
    location                text not null,
    annotation              text
)
```

```
$ mysql -u bbp -p MER < create_citations.sql
```

Preparing citation information for importation

```
$ ./get_citations * > citations.input
```

Importing tab-delimited data into citations

```
mysql> load data local infile "citations.input" into table citations;
Query OK, 34 rows affected (0.08 sec)
Records: 34 Deleted: 0 Skipped: 0 Warnings: 0
```

Working with the data in citations

```
mysql> select
-> proteins.code as 'Protein Code',
-> proteins.species as 'Protein Species',
-> dnas.entry_name as 'DNA Entry Name',
-> citations.location as 'Citation Location'
-> from proteins, dnas, crossrefs, citations
-> where proteins.accession_number = crossrefs.ac_protein
-> and dnas.accession_number = crossrefs.ac_dna
-> and dnas.accession_number = citations.accession_number
-> order by proteins.code;
```

Maxim 12.3

The SELECT query can do no harm.

Where To From Here