

# A brief introduction to SQL

# Databases

- ▶ A database application / server administers databases
- ▶ A database contains tables
- ▶ A table consists of rows and columns (records and fields)
- ▶ Each record has a unique (primary) key

# SQL

## Structured (Standard) Query Language

- ▶ Not structured well
- ▶ Not very standard
- ▶ Not just for querying
- ▶ Not a proper programming language

# Standard SQL

- ▶ ANSI / standard core for interacting with most databases
  - ▶ Oracle, Sybase, Microsoft, Access, MySQL, SQLite
- ▶ Vendor extensions and variations (unfortunately)
- ▶ Build, manipulate and query database
  - ▶ Express database as a set of SQL statements
  - ▶ `sqldump`

# Conventions

- ▶ Keywords are case insensitive
  - ▶ Usually write as UPPER
- ▶ Identifiers *may* be case sensitive
- ▶ Linebreaks & whitespace make no difference
- ▶ May need to end in semi-colon:

```
SELECT LastName FROM Persons;
```

# Creating databases

- ▶ `CREATE DATABASE <name>;`
- ▶ `CREATE DATABASE epi;`
- ▶ `CREATE TABLE <name> (<coldef_1>, <coldef_2> ...);`
- ▶ `CREATE TABLE samples (id INTEGER NOT NULL,  
country CHAR(40));`

# Data types

- ▶ Char(x)
- ▶ Integer
- ▶ Decimal(x, y) maximum digits, maximum points.
- ▶ Date
- ▶ Logical boolean
- ▶ NOT NULL (must be filled in)

# Deleting stuff

## DROP

- ▶ DROP DATABASE <name>
- ▶ DROP TABLE <name>

# Extracting data

## ▶ SELECT

▶ `SELECT <field_1>, <field_2> ... FROM <table>;`

▶ `SELECT country, date, time FROM samples;`

▶ `SELECT * FROM sampled;`

▶ `SELECT samples.country, samples.date FROM samples;`

## ▶ Conditional selection with WHERE:

▶ `SELECT id, no_hosts FROM samples WHERE no_hosts >= 10`

# Logical operators

```
SELECT id, num_hosts FROM samples WHERE (num_hosts >=
10 AND num_hosts < 30) OR country = 'BURMA'
```

- ▶ = (numbers and strings)
- ▶ <> or !=
- ▶ <, >, <=, >=
- ▶ AND, OR, ()

## More operators

```
SELECT id, num_hosts FROM samples WHERE (num_hosts  
BETWEEN 10 AND 30) OR country IN ('BURMA', 'CHINA')
```

- ▶ ... <field> BETWEEN <low> AND <high>
- ▶ ... <field> IN (<list>)
- ▶ NOT (NOT BETWEEN, NOT IN)

# Strings and Like

```
SELECT * FROM samples WHERE country LIKE '%KOREA'
```

- ▶ We quote strings (but not names)
- ▶ % is a wildcard
- ▶ No sophisticated pattern matching (in SQL)

# Creating entries

Inserting rows (not modifying):

```
INSERT INTO <table_name> VALUES (<val_1>, <val_2>, ...)
```

```
INSERT INTO <table_name> (<col_1>, <col_2>, ...) VALUES (<val_1>, <val_2>, ...)
```

# Deleting entries

- ▶ `DELETE FROM <table> WHERE <cond>`
- ▶ Will delete all that match
  - ▶ Delete by Id?

# Updating

Updates columns in rows:

```
UPDATE samples SET country = 'Myan-  
mar' WHERE id = 'X46';
```

Can update multiple

Have to think about replacing

# Aggregate functions

- ▶ SUM ()
- ▶ AVG () (average)
- ▶ MAX ()
- ▶ MIN ()
- ▶ COUNT(\*) (how many)

```
SELECT MIN(num_hosts) FROM samples;
```

```
SELECT COUNT(*) FROM samples WHERE country = 'BURMA';
```

# MySQL

- ▶ Can interact with database from commandline
- ▶ mysqldump to capture database
- ▶ Use to set up user:

```
[home]£ mysql -u olduser -p oldpass
mysql> USE epidata;
mysql> GRANT ALL ON epi-
data.* to 'newuser'@'localhost' IDENTI-
FIED BY 'newpass';
mysql> SELECT * FROM samples WHERE sam-
ple.country IS 'BUR';
```

## Worked example: Epi data

Have a series of records in a CSV file concerning epidemiological reports:

- ▶ *An unique id, e.g. "27"*
- ▶ *A country, e.g. "Taiwan"*
- ▶ *Latitude and longitude, e.g "12.7, -34.2"*
- ▶ *Confirmed as disease, e.g. "Yes"*

15, China, 10.1, 12.4, No

## Setup database

```
[home] mysql -u rootuser -p rootpassword
mysql> CREATE DATABASE epi;
mysql> USE epi;
mysql> GRANT ALL ON epi.* to 'dbuser'@'localhost' IDENTIFIED BY 'dbpass';
mysql> CREATE TABLE reports (id INTEGER NOT NULL, country CHAR(40), lat DECIMAL(3,2), lon DECIMAL(3,2), disease LOGICAL);
```

# Open database connection

Connection details may vary depending on whether the database is local or remote:

```
import MySQLdb
```

```
conn = MySQLdb.connect (user='dbuser', password='dbpass', db='epi', host='localhost')
```

## Fill database

```
import csv
in_file = open ('epi_reports.csv')
reader = csv.reader (in_file)

for id, cntry, lat, lon, dis in reader:
    cursor = conn.cursor()
    cursor.execute ('''INSERT INTO reports
        (id, country, lat, lon, disease)
        VAL-
UES (%s, %s, %s, %s, %s);''' % (id, cn-
try, lat, lon, dis))

conn.close()
```

## Search database

Find all the entries from Mexico, print their id and whether the disease has been confirmed. Assume an open connection:

```
cursor = conn.cursor()
cursor.execute ("SELECT * FROM re-
ports WHERE country = 'Mexico';")
for row in cursor.fetchall():
    print "Id: %s, dis-
ease: %s" % (row[0], row[4])
```

# Resources

- ▶ [W3 SQL tutorial](#)
- ▶ [SQLZoo SQL tutorial](#)
- ▶ [About.com SQL fundamentals](#)