A non-violent introduction to Relational Databases

(part 3)

1. Introduction to Relational Databases
2. Fundamentals of Relational Databases
3. Using Relational Databases (SQL)
4. Using Relational Databases (DBI)
5. Designing Relational Databases

mystartup

1. Departments
2. Employees
3. Clients
4. Meetings

mystartup

Departments

Department name	location
Sales	Suite 101
Research	Suite 102
Front Office	Suite 100
IT	Suite 002
Finance	Suite 103

Employees

Name	Job	Department
A. McPhereson	Salesman	Sales
C. Jones	Salesman	Sales
Z. McDaniels	Salesman	Sales
F. Pineda	Scientist	Research
D. Smith	Accountant	Finance
J. Rosen	Scientist	Research
L. Varonne	CEO	Front Office
F. Jones	Scientist	Research
S. Sowers	SysAdmin	Information Services
L. Francis	Receptionist	Front Office

Clients

Name	Address	Contact	Telephone
IBM	510 N. Wabash, Chicago IL	Ralph Cramden	773-189-0000
Apple	1 Infinite Loop, Coopertino CA	Steven Jobs	510-100-1000
WorldDom	516 Charles St. , Baltimore MD	Fernando Pineda	443-287-0000
Art House	4311 Loco Dr., Venice Beach, CA	Pumpkin Man	401-999-8100

Creating the database

1. Design the database
   - In general designing a database schema can be hard
   - For simple databases it's not too bad
   - More about this in subsequent lectures
2. Write an SQL script that creates the database
   - Statements that create the database
   - Statements that create the tables
   - Statements that load tables (only the ones that have no foreign keys!)
3. Write script (perl) that loads tables that have foreign key
   - Tables wth foreign keys need queries of other tables to get the values of the foreign keys

Create the database

(in the bash shell)

```
rm mystartup
sqlite3 mystartup mystartup.sql
```

creating tables (sqlite)

```
create table departments (
    departmentID integer primary key,
    name varchar(30),
    location varchar(30)
);

create table clients (
    clientID integer primary key,
    name varchar(40),
    address varchar(100),
    contactPerson varchar(80),
    contactNumber char(12)
);

create table employees (
    employeeID integer primary key,
    name varchar(80),
    job varchar(30),
    departmentID int not null references department(departmentID)
);
```

creating tables (sqlite)

```
create table meeting (
   clientID int not null references client(clientID),
   employeeID int not null references employee(employeeID),
   workdate date not null,
   hours float,
   primary key(clientID,employeeID, workdate)
);
```

A script to create the database and bulk load data

mystartup.sql

```
sqlite3 mystartup < mystartup.sql
```

bulk load data

```
.import departments departments.txt
.import clients clients.txt
```

But we can’t bulk load tables with foreign keys!

scripting with DBI

- The standard database interface module for Perl
- Object oriented interface
- Defines methods, variables and conventions
- Independent of actual database being used

DBI example

- DBI_test.pl

```
#!/usr/bin/perl -w
use strict;
use DBI;

# connect to the sqlite
my $dsn = 'dbi:SQLite:dbname=mystartup';
my $dbh = DBI->connect($dsn, '','',{RaiseError=>1,PrintError=>0});

# execute a query
my $stmt = "select clientID, contactNumber, contactPerson, name from clients;";
my $sth = $dbh->prepare($stmt);
$sth->execute;

# print each row of the result
while (my @row = $sth->fetchrow_array) {
        printf ("%d\t%s\t%s\t%s\n", @row);
}

$dbh->disconnect;
```

Simplified DBI usage

```
# load DBI module and connect to the RDBMS
use DBI;
$dbh = DBI->connect(...);

# example: execute SQL statements with no result
$statement = “start transaction”;
$dbh->do($statement)

# example: easy select and return one row
$statement= “select ... ”;
$row_ref = $dbh->selectrow_arrayref($statement);

# example: easy select and return multiple rows as array references
# (also exist methods for returning hash references)
$statement= “select ... ”;
$rows_ref = $dbh->selectall_arrayref($statement);
foreach $row_ref (@$rows_ref){
        print(“@$row_ref\n”)
}

# disconnect:
$dbh->disconnect

# More details http://search.cpan.org/~timb/DBI-1.637/DBI.pm
```