Database systems in 21 minutes

- Relational Database Management Systems
  - MySQL, Postgres, SQLite, Oracle, Sybase, DB2, ...
- a database is a collection of tables
- each table has a fixed number of columns
  - each column is an "attribute" common to all rows
- and a variable number of rows
  - each row is a "record" that contains data

<table>
<thead>
<tr>
<th>isbn</th>
<th>title</th>
<th>author</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>MySQL</td>
<td>DuBois</td>
<td>49.95</td>
</tr>
<tr>
<td>4321</td>
<td>TPOP</td>
<td>K &amp; P</td>
<td>24.95</td>
</tr>
<tr>
<td>2468</td>
<td>Ruby</td>
<td>Flanagan</td>
<td>79.99</td>
</tr>
<tr>
<td>2467</td>
<td>Java</td>
<td>Flanagan</td>
<td>89.99</td>
</tr>
<tr>
<td>2466</td>
<td>Javascript</td>
<td>Flanagan</td>
<td>99.99</td>
</tr>
<tr>
<td>1357</td>
<td>Networks</td>
<td>Peterson</td>
<td>105.00</td>
</tr>
<tr>
<td>1111</td>
<td>Practical Ethics</td>
<td>Singer</td>
<td>25.00</td>
</tr>
<tr>
<td>4320</td>
<td>C Prog Lang</td>
<td>K &amp; R</td>
<td>40.00</td>
</tr>
</tbody>
</table>

Relational model

- simplest database has one table holding all data
  - e.g., Excel spreadsheet
- relational model: data in separate tables "related" by common attributes
  - e.g., id in custs matches custid in sales
- schema: content and structure of the tables
  - books
    - isbn title author price
  - custs
    - id name adr
  - sales
    - isbn custid date price qty
  - stock
    - isbn count
- extract desired info by queries
- query processing figures out what info comes from what tables, extracts it efficiently
Sample database

- **books**
  - 1234 MySQL DuBois 49.95
  - 4321 TPOP K & P 24.95
  - 2468 Ruby Flanagan 79.99
  - 2467 Java Flanagan 89.99

- **custs**
  - 11 Brian Princeton
  - 22 Bob Princeton
  - 33 Bill Redmond
  - 44 Bob Palo Alto

- **sales**
  - 4321 11 2010-02-28 45.00 1
  - 2467 22 2010-01-01 60.00 10
  - 2467 11 2010-03-05 57.00 3
  - 4321 33 2010-03-05 45.00 1

- **stock**
  - 1234 100
  - 4321 20
  - 2468 5
  - 2467 0

Retrieving data from a single table

- **SQL ("Structured Query Language") is the standard language for expressing queries**
  - all major database systems support it

- **general format**
  - `select column-names from tables where condition ;`

  - `select * from books;`
  - `select name, adr from custs;`
  - `select title, price from books where price > 50;`
  - `select * from books where author = "Flanagan";`
  - `select author, title from books where author like "F%";`
  - `select author, title from books order by author;`
  - `select author, count(*) from books group by author;`

- **result is a table**
Multiple tables and joins

- if desired info comes from multiple tables, this implies a "join" operator to relate data in different tables
  - in effect join makes a big table for later selection

```sql
select title, count from books, stock
  where books.isbn = stock.isbn;

select * from books, sales
  where books.isbn = sales.isbn
  and books.author like "F%";

select custs.name, books.title
  from books, custs, sales
  where custs.id = sales.custid
  and sales.isbn = books.isbn;

select price, count(*) as count from books
  where author like 'F%' 
    group by author order by count desc;
```

Database system organization

![Diagram of database system organization]

- browser
- DB client
- DB server
- network connection
- HTTP
- SQL query
- response (table)
ACID

• the central properties of a database system:

• Atomicity
  – all or nothing: all steps of a transaction are completed
  – no partially completed transactions

• Consistency
  – each transaction maintains consistency of whole database

• Isolation
  – effects of a transaction not visible to other transactions until committed

• Durability
  – changes are permanent, survive system failure
  – consistency guaranteed

MySQL

• open source (?) relational database system
  www.mysql.com

• "LAMP"
  – Linux
  – Apache
  – MySQL
  – P*: Perl, Python, PHP

• command-line interface:
  – connect to server using command interface
    mysql -h studentdb -u bwk -p
  – type commands, read responses
    show databases;
    use bwk;
    show tables;
    select now(), version(), user();
    source cmdfile;
Creating and loading a table

- create table

```sql
create table books (
    isbn varchar(15) primary key,
    title varchar(35), author varchar(20),
    price decimal(10,2)
);
```

- load table from file (tab-separated text)

```sql
load data local infile "books" into table books
    fields terminated by "\t"
    ignore 1 lines;
```

- fields have to be left justified.
- terminated clause must be single character
  - not whitespace: multiple blanks are NOT treated as single separator

- can also insert one record at a time
  ```sql
  insert into books values('2464','AWK','Flanagan','89.99');
  ```

Item types

- INT
  - of several sizes
- FLOAT, DOUBLE, DECIMAL
- CHAR, VARCHAR
- BLOB (binary large object)
  - of several sizes
- TEXT
  - of several sizes
- ENUM
  - e.g., 'M', 'F'
- SET
- DATE, TIME, …
Other statements

• generic SQL
  - ought to be the same for all db systems
  - (though they are not always)

  insert into sales
  values('1234','44','2008-03-06','27.95');
  update books set price = 99.99
  where author = "Flanagan";
  delete from books where author = "Singer";

• MySQL-specific
  - other db's have analogous but different statements

  use bwk;
  show tables;
  describe books;
  drop tables if exists books, custs;

SQLite: an alternative (www.sqlite.org)

• small, fast, simple, embeddable
  - no configuration
  - no server
  - single cross-platform database file

• most suitable for
  - embedded devices (cellphones)
  - web sites with modest traffic & rapid processing
    <100K hits/day, 10 msec transaction times
  - ad hoc file system or format replacement
  - internal or temporary databases

• probably not right for
  - large scale client server
  - high volume web sites
  - gigabyte databases
  - high concurrency

• "SQLite is not designed to replace Oracle.
  It is designed to replace fopen()."
Program interfaces to MySQL

• original and basic interface is in C
  - about 50 functions
  - other interfaces build on this
  - most efficient access though query complexity is where the time goes
  - significant complexity in managing storage for query results

• API's exist for most other languages
  - Perl, Python, PHP, Ruby, ...
  - C++, Java, ...
  - can use MySQL from Excel, etc., with ODBC module

• basic structure for all API's is

  db_handle = connect to database
  repeat {
    stmt_handle = prepare an SQL statement
    execute (stmt_handle)
    fetch result
  } until tired
  disconnect (db_handle)

Simple standalone Perl example

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

my $dsn = "DBI:mysql:bwk:studentdb.cs.princeton.edu";
my $dbh = DBI->connect( $dsn, "bwk", "xxx", {RaiseError=>1});
print "Enter query: ";
while (<>){
  chomp;
  next if $_ eq "";
  $sth = $dbh->prepare("$_");
  $sth->execute();
  while (my @ary = $sth->fetchrow_array()) {
    print join ("\t", @ary), "\n";
  }
  $sth->finish();
  print "Enter query: ";
}

$dbh->disconnect();
Perl CGI version  (part 1: get query, access db)

#!/usr/local/bin/perl -w
use strict;
use DBI;
use CGI;
my $query = new CGI;
my $ret = "";
my $passwd = $query->param("password");
if (defined($query->param("sql"))) {
  my $dsn = "DBI:mysql:bwk:studentdb.cs.princeton.edu";
  my $dbh = DBI->connect($dsn, "bwk", $passwd, {RaiseError=>1});
  my $q = $query->param("sql");
  my $sth = $dbh->prepare($q);
  my $nchg = $sth->execute();
  my @ary;
  if ($nchg > 0) {
    while (@ary = $sth->fetchrow_array()) {
      $ret .= join ("\t", @ary), "\n";
    }
  }
  $sth->finish();
  $dbh->disconnect();
}

Perl CGI version  (part 2: generate HTML)

print $query->header;
print $query->start_html(-title=>'MySQL test', -
  bgcolor=>'white');

print qq(
  <P><form METHOD=POST enctype="multipart/form-data"
      ACTION="http://www.cs.princeton.edu/~bwk/mysql.cgi">
    Password: <input type="password" name=password text="" size="30">
    <br><textarea name=sql rows=5 cols=65 wrap>$s</textarea>
    <br><input type="submit" value="Submit"> <input type=reset>
    <br><textarea name=results rows=15 cols=60 wrap>$ret</textarea>
  </form>
);
PHP version  (just enough to demonstrate connectivity)

```php
<?php
$con = mysql_connect("studentdb.cs.princeton.edu", "bwk", "xx");
if (!$con) {
    echo "Error: couldn't connect<br>\n";
    $er = mysql_error($con);
    echo "<li> $er\n";
    exit;
}
mysql_select_db("bwk", $con);
$result = mysql_query("select * from books");
while ($row = mysql_fetch_array($result)) {
    for ($i = 0; $i < mysql_num_fields($result); $i++) {
        printf("%s ", $row[$i]);
    }
    printf("<br>\n");
}
?>
</body></html>
```

### ODBC, JDBC, and all that

- **ODBC** ("open database connectivity")
  - Microsoft standard interface between applications and databases
  - API provides basic SQL interface
  - driver does whatever work is needed to convert
  - underlying database has to provide basic services
  - used for applications like Excel, Visual Basic, C/C++, ...
  - drivers exist for all major databases
  - makes applications relatively independent of specific database being used

- **JDBC is the same thing for Java**
  - passes calls through to ODBC drivers or other database software
MySQL access from Java (Connector/J JDBC package)

```java
import java.sql.*;

public class mysql {
    public static void main(String args[]) {
        String url = "jdbc:mysql://studentdb.cs.princeton.edu/bwk";
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: "+ e.getMessage());
        }
        try {
            Connection con = DriverManager.getConnection(url, "bwk", "xxx");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("select * from books");
            while (rs.next())
                System.out.println(rs.getString("title") + " "+ rs.getString("author"));
            stmt.close();
            con.close();
        } catch(SQLException ex) {
            System.err.println("SQLException: "+ ex.getMessage());
        }
    }
}
```

Interface design

- **two different possible table structures:**
  - books
    - isbn title author price
  - booktitle, bookauthor, bookprice
  - isbn title
  - isbn author
  - isbn price

- **they need different SQL queries:**
  - select title, author, price from books;
  - select title, author, price from booktitle, bookauthor, bookprice
    - where booktitle.isbn = bookauthor.isbn
    - and bookauthor.isbn = bookprice.isbn;

- **most of the program should be independent of the specific table organization**
  - shouldn’t know or care which one is being used
    - `getList(title, author, price)`