MySQL

Relational Database

Introduction to Databases

- Provides a Structured Persistent Data Store
- Stores data in "Tables"
- Usually "Row Oriented"
  - Records are rows and values in the record are columns in the table
- Columns are usually typed
  - Often have some unique ID called a "Key"

A Simple Table

person_id	first	last
1	Nick	Palmer
2	Viola	Caretti

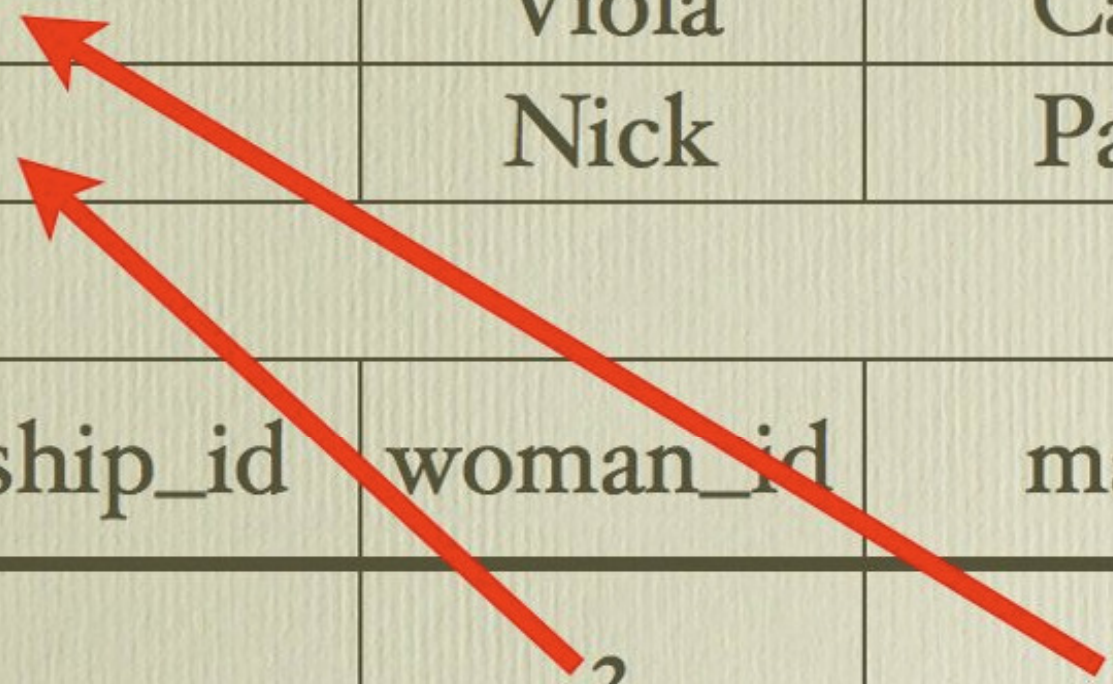
What Is Relational?

- Relational means connecting two tables
  - This makes a relationship between them
- Connection is made via Keys

A Relationship

person_id	first	last
1	Viola	Caretti
2	Nick	Palmer

relationship_id	woman_id	man_id
5	2	1



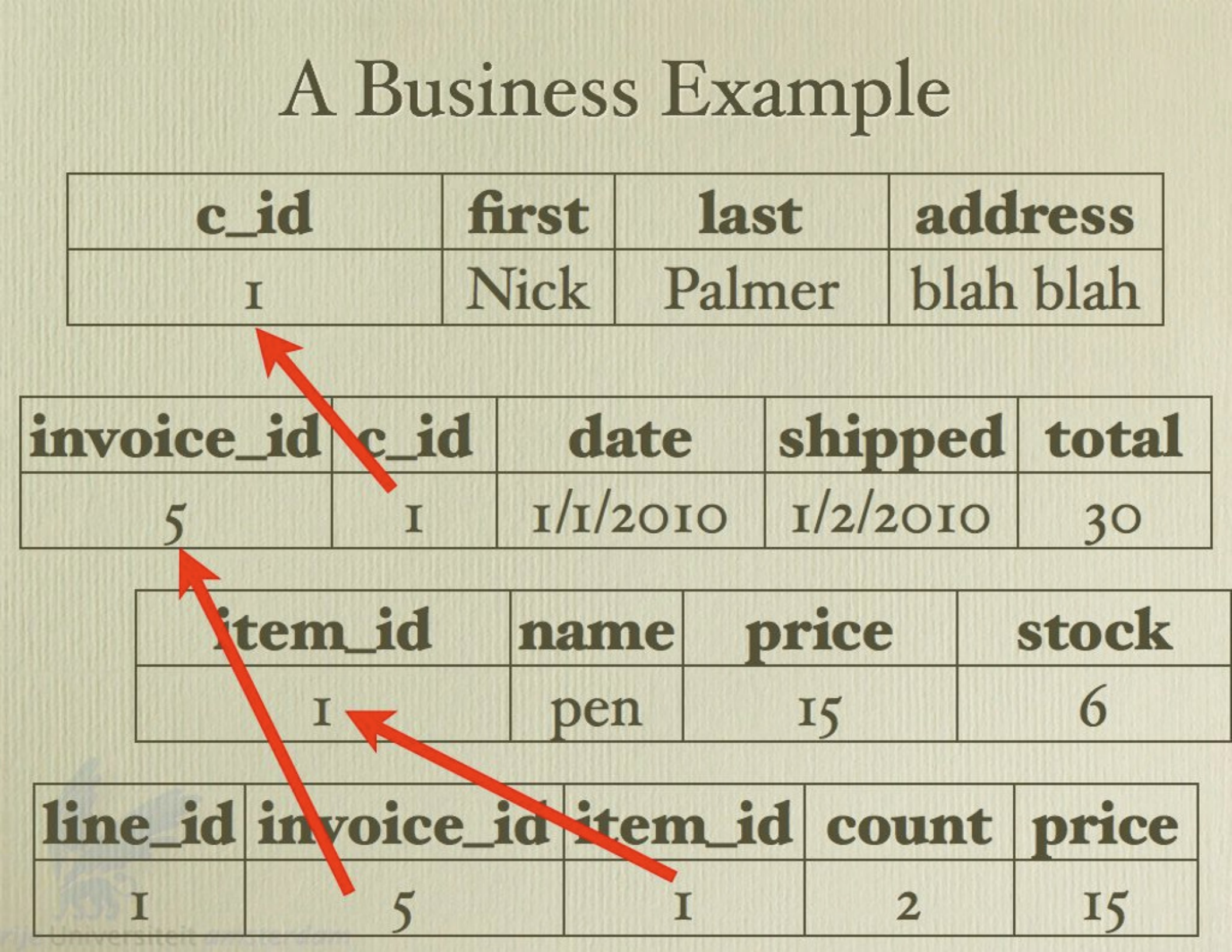
A Business Example

c_id	first	last	address
1	Nick	Palmer	blah blah

invoice_id	c_id	date	shipped	total
5	1	1/1/2010	1/2/2010	30

item_id	name	price	stock
1	pen	15	6

line_id	invoice_id	item_id	count	price
1	5	1	2	15



SQL

Structured Query Language

SQL

- Database Language (Standardized as SQL92)
  - Creation of Databases
  - Creation, Deletion, and Modification of Tables
  - Creation and Deletion of Indexes
  - Querying the Database
  - Inserting into Tables
  - Deleting Rows from Tables
  - Updating Rows in a Table

SQL

- Composed of Statments
  (Like in JavaScript and PHP)
- Statements are terminated with a ;
  (Like in JavaScript and PHP)
- Statements are executed in Order
  (Like in JavaScript and PHP)
- Strings are delimited with “
  (Like in JavaScript and PHP)

Begining SQL

- Creating a Database
  - CREATE DATABASE *name*;
  - Has been done for you already!
- Using a Database
  - USE name;
  - Scopes all tables into local scope
- Removing a Database
  - DROP DATABASE name;
  - Removes all tables and their data as well!

Beginning SQL

- Manipulating Tables
  - CREATE TABLE *table* (
    column_list
) type=*table_type*;
  - DROP TABLE table;

SQL Column Types

- Columns in MySQL are Strongly Typed!
  (Not like variables in JavaScript and PHP)
- You MUST store the right type in a column!
- Types include:
  - int, double, datetime, char, varchar, blob, etc
  - Some can be sized
    - int(11), char(5), varchar(128)

SQL Constraints

- Used to ensure data integrity
- Limits data in a column or columns
- Examples:
  - AUTO INCREMENT
    - Default to a SEQUENCE of integer values
  - NOT NULL
    - Ensures that a column is filled in
  - FOREIGN KEY
    - Ensures referential integrity

SQL Defaults

- Columns can also have a default value
- Useful particularly for row creation date

SQL Keys

- Primary Keys are a row identifier
  - They MUST be unique!
- Keys are a possibly non-unique identifier

Example Table Creation

```
mysql>CREATE TABLE joke (
    ->  id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->  joketext TEXT,
    ->  jokedate DATE NOT NULL
    ->);
```

Example Table Creation

```
CREATE TABLE payments (
        customerNumber int NOT NULL,
        checkNumber varchar(50) NOT NULL,
        paymentDate datetime NOT NULL,
        amount double NOT NULL,
        PRIMARY KEY  (customerNumber,checkNumber)
);
```

Querying

- Querying allows you to fetch data
- Uses a SELECT statement
- SELECT [DISTINCT]
  column_name1,column_name2...
  FROM tables
  [WHERE conditions]
  [GROUP BY group
    [HAVING group_conditions]]
  [ORDER BY sort_columns]
  [LIMIT limits];

Querying Examples

- SELECT * from employees;
- SELECT last_name, first_name FROM employees;
- SELECT first_name, last_name FROM employees WHERE jobtitle=”president”;
- SELECT first_name, last_name FROM employees WHERE jobtitle=”president” OR jobtitle=”vice-president”;
- SELECT first_name, last_name FROM employees WHERE jobtitle=”sales rep” AND salary > 1000;

SQL Grouping

- SELECT count(*), jobTitle
  FROM employees GROUP BY jobTitle;

- SELECT count(*), jobTitle
  FROM employees
  GROUP BY jobTitle
  HAVING count(*) = 1

  - HAVING is simply WHERE on a GROUP

SQL Sorting

- SELECT firstname, lastname, jobtitle
  FROM employees
  ORDER BY firstname ASC, jobtitle DESC;

SQL IN

- Allows you to select where you want the value to be one of an array of values
- SELECT *column_list*
  FROM *table_name*
  WHERE *column* IN ("*list_item1*","*list_item2*"…)
- SELECT officeCode, city, phone
  FROM offices
  WHERE country = “USA” OR country = “France”;
- SELECT officeCode, city, phone
  FROM offices
  WHERE country IN (“USA”, “France”);

Inserting Data

- INSERT INTO *table* [(*column*,...)] VALUES (*expression* | *DEFAULT*, ...);
- INSERT INTO employees (first_name, last_name), VALUES (“nick”, “palmer”);
- INSERT INTO employees VALUES (“nick”, “palmer”);

Updating Data

- UPDATE *table* SET *column=value*[,...]
  [WHERE *where*];
- UPDATE employess SET salary = 100000
  WHERE first = “Nick”
  AND last = “Palmer”;

Deleting Data

- DELETE FROM *table* [WHERE *where*];
- DELETE FROM employees WHERE salary < 10;

Transactions

All about ACID

On Transactions

- Transactions bundle a number of operations together
- Ensures that either all operations complete, or none do
- Ensures that no data seen by a transaction is modified between the time it was seen and the commit of the transaction
- Handled using locks (Either on tables or rows within a table)

Why do we need Transactions?

- First checkout:
  $count = SELECT count from inventory where item_id = 1;
- Second checkout:
  $count = SELECT count from inventory where item_id = 1;
- First checkout:
  UPDATE inventory set count = ($count-1) WHERE item_id = 1;
- Second Checkout:
  UPDATE inventory set count = ($count-1) WHERE item_id = 1;
- We sold two but only had 1 in inventory!

Connecting

- $connection = mysql_connnect($servername, $username, $password);
- if (!$connection) {
  die("Not Connected: " . mysql_error());
  }

Selecting The Database

- $selected = mysql_select_db('database', $connection);
- if (!$selected) {
  die("Can't use db." . mysql_error());
  }

Query the Database

```
$result = mysql_query("SELECT * FROM Persons");

if (!$result) die("Unable to query: " . mysql_error());

while($row = mysql_fetch_array($result)) {
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br />";
}
```

Closing a Connection

- You must close connections you open!
- mysql_close($connection);

SQL Security

Avoiding SQL Injection

SQL: Security

- SQL Injection
  - Another class of lack of input validation
  - Causes corruption of database or security leak
  - “SELECT * FROM users WHERE name='$username' AND pass='$password'”;
  - $username = “admin” $password = “ ‘ OR ‘1’=’1”

PHP: Security

- Preventing SQL Injection:
  - Use addslashes() to sanitize data or better yet a database specific version like mysql_real_escape_string()
  - Or use a library with bound queries like ADOdb (http://adodb.sourceforge.net/) or PDO prepare() execute()

PHP: Reference

- http://www.php.net/manual/en/
- http://www.w3schools.com/PHP/
- Some PDFs with information on JavaScript, PHP and MySQL in the "Course Materials" folder on the server.
- Check the webpage version of the slides on the server! (BIG HINT!)