Database Management Systems
Session 3

Instructor: Vinnie Costa
vcosta@optonline.net
The Entity-Relationship Model

Chapter 2 - Redux
How To Design A Database
- In Six “Easy” Steps!

1. Requirements Analysis
2. Conceptual Database Design
3. Logical Design
4. Schema Refinement
5. Physical Database Design
6. Application And Security Design
Requirements Analysis

- This is the most difficult step
- Source of most problems
- What does the user want?
- Discussions, study of current environment, expectations (need to be managed), history, resources available, etc.
- Elicitation Methodologies
- Bit of an art form
Conceptual Database Design

◆ High-level description of data to be stored…
◆ … and constraints over the data
◆ Diagrams – semantic data model
  ▪ What are the *entities* and *relationships* in the enterprise?
  ▪ What information about these entities and relationships should we store in the database?
  ▪ What are the *integrity constraints* or *business rules* that hold?
  ▪ A database `schema’ in the ER Model can be represented pictorially (*ER diagrams*).
◆ Can map an ER diagram into a relational schema

(1) DBDesigner4 – excellent tool to translate ER Model in MySQL tables
Logical Database Design

- Choose a DBMS – we use Relational Only
- Convert the ER schema into a relational database schema
- Result is a conceptual schema also called the logical schema
The “Systems” Steps

- **Schema Refinement** – identify problems and refine the schema – eliminate redundancy through normalization
- **Physical Database Design** – workload analysis, scaling, indexing and clustering of tables, database tuning
- **Application and Security Design** – the “big” picture in design, workflow, role based security, transparency
ER Model Basics

- **Entity**: Real-world object distinguishable from other objects. An entity is described (in DB) using a set of **attributes**.

- **Entity Set**: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a **key**.
  - Each attribute has a **domain**.
ER Model Basics

Employees

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
</tr>
</tbody>
</table>
ER Model Basics

- **Relationship** - Association among two or more entities. e.g., Attishoo works in the Pharmacy department.
- This can be viewed as a set of **2-tuples**: 
  \[ \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2\} \]
- In other words, \( \text{Works}_\text{In} = \{(e,d) \mid e \in \text{Employees}, d \in \text{Departments}\} \)
- **Works}_\text{In} = \{(\text{Attishoo, Pharmacy}), (\text{Pat, Hardware}), (\text{Sue, Automotive}), \ldots\} \]
- This can also be called a **Relationship Set**
**ER Model Basics**

- **Relationship Set** - Collection of similar relationships.
  - An n-ary relationship set $R$ relates $n$ entity sets $E_1 \ldots E_n$; each relationship in $R$ involves entities $e_1 \in E_1, \ldots, e_n \in E_n$
  - Same entity set could participate in different relationship sets, or in different “roles” in same set. **supervisor** and **subordinate** are role indicators
ER Model Basic

- An **instance** is a snapshot of the relationship set at some instant in time.
- Suppose each department has offices in several locations, then we can record an association between an employee, a department, and a location – a 3-tuple. This would be a **ternary** relationship.

![ER Diagram](image-url)
ER Model Features – Key Constraints

- Consider Works_In: An employee can work in many departments; a dept can have many employees.

- In contrast, each dept has at most one manager, according to the key constraint on Manages.
ER Model Features – Key Constraints

- Relationship set like **Manages** is said to be **one-to-many**: one employee can be associated with many departments.
- In Contrast, **Works_In**, where an employee is allowed to work in several departments and a department is allowed to have several employees is said to be **many-to-many**.
Does every department have a manager?

If so, this is a **participation constraint**: the participation of Departments in Manages is said to be **total** (vs. **partial**).

- Every Departments entity must appear in an instance of the Manages relationship.

**Thick line** indicates **total** participation
ER Model Features - Weak Entities

- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.
- Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
- Weak entity set must have total participation in this identifying relationship set.
- pname is a partial key for the weak entity set
- Dependents is a weak entity and Policy is its identifying relationship. This is indicated by a thick black line
It is natural to classify the entities in an entity set into subclasses. Hourly_Emps and Contract_Emps inherit the attributes of the entity set Employees. If we declare A ISA B, every A entity is also considered to be a B entity, i.e., Hourly_Emps ISA Employees. We could add a second ISA node for Senior_Emps.
Two ways to view class hierarchies:

- Employees is **specialized** into subclasses – identify subsets of an entity set (the **superclass**) that share some distinguishing characteristic (top-down)
- Hourly_Emps and Contract_Emps are **generalized** by Employee – create new entity that has common characteristics of the subclasses (bottom-up)

Two kinds of constraint:

- **Overlap** – do two subclasses contain the same entity? Assume **no**, otherwise write ‘___ OVERLAPS ___’
- **Covering** – do the entities in the subclasses collectively include all the entities in the superclass? Assume **no**, otherwise write ‘___ AND ___ COVER ___’
**ER Model - Aggregation**

- Used when we have to model a relationship involving **entity sets** and a **relationship set** (instead of another entity set).
- **Aggregation** allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.
- Used to express a relationship among relationships
- **Aggregation vs. ternary relationship**: Could we make Sponsors a ternary relationship?
  - Monitors is a distinct relationship, with a descriptive attribute (until) and Sponsors has a unique attribute (since)
  - Also, we can say that each sponsorship is monitored by at most one employee
ER Model - Aggregation

Treat sponsors as an entity set
Conceptual Design Using the ER Model

- **Design choices:**
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?

- **Constraints in the ER Model:**
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).
Entities vs. Attributes (Contd.)

- Works_In4 does not allow an employee to work in a department for two or more periods.

- Similar to the problem of wanting to record several addresses for an employee: We want to record several values of the descriptive attributes for each instance of this relationship. Accomplished by introducing new entity set, Duration.
Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept
- What if a manager gets a discretionary budget that covers all managed depts?
  - **Redundancy** - dbudget stored for each dept managed by manager
  - **Misleading** - Suggests dbudget associated with department-mgr combination

This fixes the problem!
Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.

- What are the additional constraints in the 2nd diagram?
Binary vs. Ternary Relationships (Contd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.

- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute qty. No combination of binary relationships is an adequate substitute:
  - S “can-supply” P, D “needs” P, and D “deals-with” S does not imply that D has agreed to buy P from S.
  - How do we record qty?
Summary of Conceptual Design

- **Conceptual design** follows requirements analysis,
  - Yields a high-level description of data to be stored

- **ER model popular** for conceptual design
  - Constructs are expressive, close to the way people think about their applications

- **Basic constructs**: entities, relationships, and attributes (of entities and relationships)

- Some **additional constructs**: weak entities, ISA hierarchies, and aggregation

- **Note**: There are many variations on ER model.
Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: key constraints, participation constraints, and overlap/covering constraints for ISA hierarchies. Some foreign key constraints are also implicit in the definition of a relationship set.

- Some constraints (notably, functional dependencies) cannot be expressed in the ER model.

- Constraints play an important role in determining the best database design for an enterprise.
Summary of ER (Contd.)

- ER design is **subjective**. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - **Entity vs. attribute**, **entity vs. relationship**, **binary or n-ary relationship**, whether or not to use **ISA hierarchies**, and whether or not to use **aggregation**

- **To ensure good database design**: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful
Useful Websites

- Edgar (Ted) Codd – biographical sketch
- **Modeling Tools** – good list of available tools – checkout: DIA, ERwin, DBDesigner4, SmartDraw
Homework

- Read Chapter Two
- Exercises p.52: 2.1, 2.2 (1-5)
Exercise 2.1

- Explain the following terms briefly: attribute, domain, entity, relationship, entity set, relationship set, one-to-many relationship, many-to-many relationship, participation constraint, overlap constraint, covering constraint, weak entity set, aggregation, and role indicator.
Exercise 1.1

- **Attribute** - a property or description of an entity. A toy department employee entity could have attributes describing the employee’s name, salary, and years of service.
- **Domain** - a set of possible values for an attribute.
- **Entity** - an object in the real world that is distinguishable from other objects.
- **Relationship** - an association among two or more entities.
- **Entity set** - a collection of similar entities such as all of the toys in the toy department.
- **Relationship set** - a collection of similar relationships
- **One-to-many relationship** - a key constraint that indicates that one entity can be associated with many of another entity. An example of a one-to-many relationship is when an employee can work for only one department, and a department can have many employees.
Exercise 1.1

- **Many-to-many relationship** - a key constraint that indicates that many of one entity can be associated with many of another entity. An example of a many-to-many relationship is employees and their hobbies: a person can have many different hobbies, and many people can have the same hobby.

- **Participation constraint** - a participation constraint determines whether relationships must involve certain entities. An example is if every department entity has a manager entity. Participation constraints can either be total or partial. A total participation constraint says that every department has a manager. A partial participation constraint says that every employee does not have to be a manager.

- **Overlap constraint** - within an ISA hierarchy, an overlap constraint determines whether or not two subclasses can contain the same entity.

- **Covering constraint** - within an ISA hierarchy, a covering constraint determines whether the entities in the subclasses collectively include all entities in the superclass. For example, with an Employees entity set with subclasses HourlyEmployee and SalaryEmployee, does every Employee entity necessarily have to be within either HourlyEmployee or SalaryEmployee?
Exercise 1.1

- **Covering constraint** - within an ISA hierarchy, a covering constraint determines whether the entities in the subclasses collectively include all entities in the superclass. For example, with an Employees entity set with subclasses HourlyEmployee and SalaryEmployee, does every Employee entity necessarily have to be within either HourlyEmployee or SalaryEmployee?

- **Weak entity set** - an entity that cannot be identified uniquely without considering some primary key attributes of another identifying owner entity. An example is including Dependent information for employees for insurance purposes.

- **Aggregation** - a feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.

- **Role indicator** - If an entity set plays more than one role, role indicators describe the different purpose in the relationship. An example is a single Employee entity set with a relation Reports-To that relates supervisors and subordinates.
Exercise 2.2

A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each situation, draw an ER diagram that describes it (assuming no further constraints hold).

1. Professors can teach the same course in several semesters, and each offering must be recorded.
2. Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)
3. Every professor must teach some course.
4. Every professor teaches exactly one course (no more, no less).
5. Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.
Exercise 2.2

Figure 2.1  ER Diagram for Exercise 2.2, Part 1

Figure 2.2  ER Diagram for Exercise 2.2, Part 2
Exercise 2.2

Figure 2.3  ER Diagram for Exercise 2.2, Part 3

Figure 2.4  ER Diagram for Exercise 2.2, Part 4
Exercise 2.2

Figure 2.5  ER Diagram for Exercise 2.2, Part 5
Term Paper

- Due Saturday, Oct 8
- Should be about 3-4 pages (9 or 10 font)
- This should be an opportunity to explore a selected area
- Please submit your topics!!!
Practicum

- Install Apache
- Install Nvu
- …on our way to WAMP!!!
Apache

- [The Apache Software Foundation](http://www.apache.org/)

- [httpd.apache.org](http://httpd.apache.org)

- The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

- Apache has been the most popular web server on the Internet since April of 1996. More than 68% of the web sites on the Internet are using Apache, thus making it more widely used than all other web servers combined.
Install Apache

- [http://httpd.apache.org/docs/2.0/platform/windows.html](http://httpd.apache.org/docs/2.0/platform/windows.html)
- Installing apache is easy if you download the Microsoft Installer (.msi) package. Just double click on the icon to run the installation wizard. Click next until you see the Server Information window. You can enter localhost for both the Network Domain and Server Name. As for the administrator's email address you can enter anything you want.
- If using Windows XP, installed Apache as Service so every time I start Windows Apache is automatically started.
Click the **Next** button and choose **Typical installation.** Click Next one more time and choose where you want to install Apache (I installed it in the default location C:\Program Files\Apache Group). Click the Next button and then the Install button to complete the installation process.
Installing Apache

To see if your Apache installation was successful open up your browser and type http://localhost (or http://127.0.0.1) in the address bar. You should see something like this:

If you can see this, it means that the installation of the Apache web server software on this system was successful. You may now add content to this directory and replace this page.

---

**Seeing this instead of the website you expected?**

This page is here because the site administrator has changed the configuration of this web server. Please contact the person responsible for maintaining this server with questions. The Apache Software Foundation, which wrote the web server software this site administrator is using, has nothing to do with maintaining this site and cannot help resolve configuration issues.

---

The Apache documentation has been included with this distribution.

You are free to use the image below on an Apache-powered web server. Thanks for using Apache!
Installing Apache

- By default Apache's **document root** is set to **htdocs** directory. The document root is where you must put all your PHP or HTML files so it will be processed by Apache (and can be seen through a web browser). Of course you can change it to point to any directory you want. The configuration file for Apache is stored in `C:\Program Files\Apache Group\Apache2\conf\httpd.conf` (assuming you installed Apache in `C:\Program Files\Apache Group`). It's just a plain text file so you can use Notepad to edit it.

- For example, if you want to put all your PHP or HTML files in `C:\www` just find this line in the `httpd.conf`:
  ```
  DocumentRoot "C:/Program Files/Apache Group/Apache2/htdocs"
  ```
  and change it to:
  ```
  DocumentRoot "C:/www"
  ```

- After making changes to the configuration file you have to restart Apache (Start > Programs > Apache HTTP Server 2.0 > Control Apache Server > Restart) to see the effect.
Installing Apache

- Another configuration you may want to change is the directory index. This is the file that Apache will show when you request a directory. As an example if you type `http://www.php-mysql-tutorial.com/` without specifying any file the index.php file will be automatically shown.

- Suppose you want apache to use index.html, index.php or main.php as the directory index you can modify the DirectoryIndex value like this:

  ```
  DirectoryIndex index.html index.php main.php
  ```

- Now whenever you request a directory such as `http://localhost/` Apache will try to find the index.html file or if it's not found Apache will use index.php. In case index.php is also not found then main.php will be used.
Installing Nvu

- www.nvu.com/
- A complete Web Authoring System for Linux Desktop users as well as Microsoft Windows and Macintosh users to rival programs like FrontPage and Dreamweaver.
- **Nvu** (pronounced N-view, for a "new view") makes managing a web site a snap. Now anyone can create web pages and manage a website with no technical expertise or knowledge of HTML.
Make A Home Page

- **Create an** index.html **page with Nvu**
- **Copy** C:\Program Files\Apache Group\Apache2\htdocs to old_htdocs
- **Put the** index.html **into** htdocs
- **Test with** [http://localhost](http://localhost) or [http://127.0.0.1](http://127.0.0.1)
- **Explore Cascading Style Sheets (CSS)**
Useful Websites

- [www.w3.org/Style/CSS/](http://www.w3.org/Style/CSS/) - the authoritative source
- [http://www.w3.org/Style/Examples/011/firstcss](http://www.w3.org/Style/Examples/011/firstcss) – *Starting with HTML + CSS* – good beginners guide
- [www.csszengarden.com](http://www.csszengarden.com) – A demonstration of what can be accomplished visually through CSS-based design
Homework

- Install Apache On Your System
- Install Nvu
- Create your own home page
- Play with HTML
- Play with CSS
- Play, play, play, ...
Google’s Major Coup

- September 8, 2005 – Google Inc. today announced that it hired Vinton (Vint) Cerf, the longtime technologist who is widely known as a "founding father" of the Internet, as Chief Internet Evangelist.

- He played a key role in leading the development of the TCP/IP protocols and the Internet.
Side Effects

http://www.thinkgeek.com/tshirts/
The Relational Model

Chapter 3
Why Study the Relational Model?

◆ Most widely used model
  ▪ Vendors: IBM, Microsoft, Oracle, Sybase, MySQL
◆ Started with Ted Codd’s pioneering work in ’70s
◆ Legacy systems in older models
  ▪ e.g., IBM’s IMS
◆ Recent competitor: object-oriented model
  ▪ ObjectStore, Versant, Ontos
  ▪ A synthesis emerging: object-relational model
    • Informix Universal Server, UniSQL, O2, Oracle, DB2
◆ Another Competitor: XML Data Stores
Relational Database: Definitions

- **Relational database**: a set of relations
- **Relation** is made up of two parts:
  - **Schema** - specifies name of relation, plus name and type (domain) of each column (field or attribute)
    
    ```
    Students(sid::string, name::string, login::string, age::integer, gpa::real)
    ```
  - **Instance** - a table, with rows and columns
    Number of rows = **cardinality**
    Number of fields = **degree / arity**

- Can think of a relation as a set of rows or **tuples** (i.e., all rows are distinct).
Example Instance of Students Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

- Cardinality = 3, degree = 5, all rows distinct
- Do all columns in a relation instance have to be distinct?
- Domain of a field is essentially the type
Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.
The SQL Query Language

- Developed by IBM (system R) in the 1970s
- Need for a standard since it is used by many vendors
- Standards:
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision)
  - SQL-99 (major extensions, current standard)
The SQL Query Language

- Ask a question, get an answer!
- To find all 18 year old students, we can write:

```sql
SELECT *
FROM Students S
WHERE S.age=18
```

- To find just names and logins, replace the first line:

```sql
SELECT S.name, S.login
```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@ee</td>
<td>18</td>
<td>3.2</td>
</tr>
</tbody>
</table>
Querying Multiple Relations

What does the following query compute?

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Given the following instances of Enrolled and Students:

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

we get:

<table>
<thead>
<tr>
<th>S.name</th>
<th>E.cid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>Topology112</td>
</tr>
</tbody>
</table>
Creating Relations in SQL

- Creates the Students relation. Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

- As another example, the Enrolled table holds information about courses that students take.

```sql
CREATE TABLE Students
(sid: CHAR(20),
 name: CHAR(20),
 login: CHAR(10),
 age: INTEGER,
 gpa: REAL)

CREATE TABLE Enrolled
(sid: CHAR(20),
 cid: CHAR(20),
 grade: CHAR(2))
```
Destroying and Altering Relations

DROP TABLE Students

- Destroys the relation Students. The schema information and the tuples are deleted.

ALTER TABLE Students
  ADD COLUMN firstYear: integer

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a null value in the new field.
Adding and Deleting Tuples

- Can insert a single tuple using:

  \[
  \text{INSERT INTO Students (sid, name, login, age, gpa) VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)}
  \]

- Can delete all tuples satisfying some condition (e.g., name = Smith):

  \[
  \text{DELETE FROM Students S WHERE S.name = 'Smith'}
  \]
Integrity Constraints (ICs)

- **IC** - condition that must be true for *any* instance of the database; e.g., *domain constraints*.
  - ICs are **specified** when schema is **defined**.
  - ICs are **checked** when relations are **modified**.
- A **legal** instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!
Primary Key Constraints

- A set of fields is a **candidate key** for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
- If part 2 false? A **superkey**.
  \{sid, name\} is an example of a superkey

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eeecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Primary Key Constraints

- If there’s >1 key for a relation, one of the keys is chosen (by DBA) to be the primary key.
- e.g., sid is a key for Students. (What about name?) The set \{sid, gpa\} is a superkey.

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Primary and Candidate Keys in SQL

- Possibly many candidate keys (specified using UNIQUE), one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.” vs. “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```sql
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid, cid) )
```

- Used carelessly, an IC can prevent the storage of database instances that arise in practice!

```sql
CREATE TABLE Enrolled (sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid),
UNIQUE (cid, grade) )
```
Foreign Keys, Referential Integrity

- **Foreign key** - Set of fields in one relation that is used to refer to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a logical pointer.

- e.g. *sid* is a foreign key referring to Students:
  - Enrolled(*sid*: string, *cid*: string, *grade*: string)
  - If all foreign key constraints are enforced, referential integrity is achieved, i.e., no dangling references.

- Can you name a data model w/o referential integrity? …
- …Links in HTML!
Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses

```
CREATE TABLE Enrolled ( sid CHAR(20),
    cid CHAR(20),
    grade CHAR(2),
    PRIMARY KEY (sid,cid),
    FOREIGN KEY (sid) REFERENCES Students )
```

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53666</td>
<td>Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53650</td>
<td>Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53666</td>
<td>History105</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Enforcing Referential Integrity

Consider Students and Enrolled; \textit{sid} in Enrolled is a foreign key that references Students.

\textbf{Options:}

1. What should be done if an Enrolled tuple with a non-existent student id is inserted? \textit{(Reject it!)}

2. What should be done if a Students tuple is deleted?
   - Also delete all Enrolled rows that refer to it
   - Disallow deletion of a Students rows that is referred to
   - Set \textit{sid} in Enrolled rows that refer to it to a default \texttt{sid}
   - Set \textit{sid} in Enrolled rows that refer to it to a special value \texttt{null}, denoting \texttt{unknown} or \texttt{inapplicable}

3. Similar if primary key of Students rows is updated
Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all rows that refer to deleted row)
  - SET NULL / SET DEFAULT (sets foreign key value of referencing row)

```sql
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
  ON DELETE CASCADE
  ON UPDATE SET DEFAULT )
```
Where do ICs Come From?

- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations.

- We can check a database instance to see if an IC is violated, but we can **NEVER** infer that an IC is true by looking at an instance.
  - An IC is a statement about *all possible* instances!
  - From example, we know *name* is not a key, but the assertion that *sid* is a key is given to us.

- Key and foreign key ICs are the most common; more general ICs supported too.
Logical DB Design: ER to Relational

- Translating entity sets into tables:

CREATE TABLE Employees
(ssn CHAR(11),
name CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
</tr>
</tbody>
</table>
In translating a relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys). This set of attributes forms a superkey for the relation.
- All descriptive attributes.

```sql
CREATE TABLE Works_In (ssn CHAR(11),
                        did INTEGER,
                        since DATE,
                        PRIMARY KEY (ssn, did),
                        FOREIGN KEY (ssn) REFERENCES Employees,
                        FOREIGN KEY (did) REFERENCES Departments)
```
Review: Key Constraints

- Each dept has at most one manager, according to the key constraint on Manages.

Translation to relational model?

1-to-1  1-to Many  Many-to-1  Many-to-Many

<1,1,x>  <2,2,x>  <1,3,x>  <2,3,x>
Translating ER Diagrams with Key Constraints

- Map relationship to a table:
  - Note that **did** is the key now!
  - Separate tables for Employees and Departments

- Since each department has a unique manager, we could instead **combine Manages and Departments** (usually a better approach)

- Eliminates need for a separate Manages relation

```sql
CREATE TABLE Manages(
    ssn CHAR(11),
    did INTEGER,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11),
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees)
```
Translating Participation Constraints

Does every department have a manager?

◆ If so, this is a **participation constraint** - the participation of Departments in Manages is said to be **total** (vs. **partial**).

◆ Every *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to table constraints or assertions).

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11) NOT NULL,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE NO ACTION)
```
Translating Weak Entities

- A **weak entity** can be identified uniquely only by considering the primary key of another **owner** entity (remember partial key).

- Owner entity set and weak entity set must participate in a **one-to-many relationship** set (1 owner, many weak entities).

- Weak entity set must have **total participation** in this **identifying** relationship set.
Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
- When the owner entity is deleted, all owned weak entities must also be deleted.

```sql
CREATE TABLE Dep_Policy (  
  pname CHAR(20),  
  age INTEGER,  
  cost REAL,  
  ssn CHAR(11) NOT NULL,  
  PRIMARY KEY (pname, ssn),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE CASCADE)
```
ER Model – Class Hierarchies

- **Hourly_Emps** and **Contract_Emps** inherit the attributes of the entity set **Employees**
- If we declare A ISA B, every A entity is also considered to be a B entity, i.e., Hourly_Emps ISA Employees

Two kinds of constraints:
- **Overlap constraints** - Can Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
- **Covering constraints** - Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
Translating ISA Hierarchies to Relations

◆ General approach:
  - 3 relations: Employees, Hourly_Emps and Contract_Emps
    • Hourly_Emps: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (hourly_wages, hours_worked, ssn); must delete Hourly_Emps row if referenced Employees row is deleted)
    • Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes

◆ Alternative:
  - Just Hourly_Emps and Contract_Emps
    • Hourly_Emps: ssn, name, lot, hourly_wages, hours_worked.
    • Each employee must be in one of these two subclasses.
Review: Binary vs. Ternary Relationships

- What are the additional constraints in the 2nd diagram?
Binary vs. Ternary Relationships

The key constraints allow us to combine Purchaser with Policies and Beneficiary with Dependents.

```
CREATE TABLE Policies (  
policyid INTEGER,  
cost REAL,  
ssn CHAR(11) NOT NULL,  
PRIMARY KEY (policyid),  
FOREIGN KEY (ssn) REFERENCES Employees,  
ON DELETE CASCADE)
```

```
CREATE TABLE Dependents (  
pname CHAR(20),  
age INTEGER,  
policyid INTEGER,  
PRIMARY KEY (pname, policyid),  
FOREIGN KEY (policyid) REFERENCES Policies,  
ON DELETE CASCADE)
```
Views

- A view is just a relation, but we store a definition, rather than a set of tuples

```
CREATE VIEW YoungActiveStudents (name, grade) AS
   SELECT S.name, E.grade
   FROM Students S, Enrolled E
   WHERE S.sid = E.sid and S.age<21
```

- Views can be dropped using the `DROP VIEW` command
Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s)

- Given YoungStudents, but not Students or Enrolled, we can find students s who have are enrolled, but not the cid’s of the courses they are enrolled in
Relational Model: Summary

- A **tabular** representation of data
- Simple and intuitive, currently the most widely used
- **Integrity constraints** can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important **ICs**: primary and foreign keys
  - In addition, we *always* have **domain constraints**
- Powerful and natural query languages exist
- Rules (and tools) to translate ER to relational model
PHP

- www.php.net

- **PHP** is a popular open-source, reflective programming language used mainly for developing server-side applications and dynamic web content. It was originally developed in 1994 and PHP stood for "Personal Home Page". In 2000 the Zend Engine was added and now the official meaning is the recursive acronym "**PHP** Hypertext Preprocessor".

- PHP is currently one of the most popular server-side scripting systems on the Web. It has been widely adopted since the release of version 4. On the desktop it has been favored by some new programmers as a rapid prototyping environment.
Installing PHP

- [www.php.net/downloads.php#v4](http://www.php.net/downloads.php#v4)
- We want to install PHP 4.4.0 and use the ZIP package
- Extract the PHP package (PHP 4.4.0 zip package). Extract the package in the directory where Apache was installed (C:\Program Files\Apache Group\Apache2). Change the newly created directory name to **php** (just to make it shorter).
- Then copy the file `php.ini-dist` in PHP directory to your windows directory (C:\Windows or C:\Winnt depends on where you installed Windows) and rename the file to `php.ini`. This is the PHP configuration file and we'll take a look what's in it later on.
- Next, move the `php4ts.dll` file from the newly created php directory into the **sapi** subdirectory.
Installing PHP

- Apache doesn't know that you just installed PHP. We need to tell Apache about PHP and where to find it. Open the Apache configuration file in \C:\Program Files\Apache Group\Apache2\conf\httpd.conf and add the following three lines:
  
  LoadModule php4_module php/sapi/php4apache2.dll
  AddType application/x-httpd-php .php
  AddType application/x-httpd-php-source .phps

- The first line tells Apache where to load the dll required to execute PHP and the second line means that every file that ends with .php should be processed as a PHP file. The third line is added so that you can view your php file source code in the browser window.

- Now restart Apache for the changes to take effect (Start > Programs > Apache HTTP Server 2.0.50 > Control Apache Server > Restart).
Installing PHP

- Now we want to test PHP to verify our installation. Create a new file using Nvu, name it hello.php, and put it in document root directory (C:\Program Files\Apache Group\Apache2\htdocs). The content of this file should be:
  ```php
  <?php
  echo 'Hello World!';
  ?>
  ```
  (Note: Nvu will do the php encapsulation for you)

- Type http://localhost/hello.php on your browser's address bar and if everything works well you should see the traditional “Hello World!” display in your browser.

- Another common test is to create a new file named test.php and put it in document root directory. The content of this file is:
  ```php
  <?php
  phpinfo();
  ?>
  ```
Installing PHP

- `phpinfo()` is the infamous PHP function which will spit out all kinds of stuff about PHP and your server configuration. Type `http://localhost/test.php` on your browser's address bar and if everything works well you should see something like this:

```
PHP Version 4.4.0

<table>
<thead>
<tr>
<th>System</th>
<th>Windows NT LVINGROOM 5.1 build 2600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Date</td>
<td>Jul 11 2005 16:06:47</td>
</tr>
<tr>
<td>Server API</td>
<td>Apache 2.0 Handler</td>
</tr>
<tr>
<td>Virtual Directory Support</td>
<td>enabled</td>
</tr>
<tr>
<td>Configuration File (php.ini) Path</td>
<td>C:\WINNT\php.ini</td>
</tr>
<tr>
<td>PHP API</td>
<td>20020018</td>
</tr>
<tr>
<td>PHP Extension</td>
<td>20020429</td>
</tr>
<tr>
<td>Zend Extension</td>
<td>20050606</td>
</tr>
<tr>
<td>Debug Build</td>
<td>no</td>
</tr>
</tbody>
</table>
```
MySQL

- MySQL is a multithreaded, multi-user, SQL (Structured Query Language) Database Management System (DBMS) with an estimated six million installations. MySQL is open source software available either under the GNU General Public License (GPL) or under other licenses when the GPL is inapplicable to the intended use.¹

- Unlike projects such as Apache, where the software is developed by a public community, and is essentially not owned by anyone, MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB. The company develops and maintains the system, selling support and service contracts, as well as commercially-licensed copies of MySQL, and employing people all over the world who work together via the Internet. Two Swedes and a Finn founded MySQL AB: David Axmark, Allan Larsson and Michael "Monty" Widenius.²

¹ [en.wikipedia.org/wiki/MySQL](http://en.wikipedia.org/wiki/MySQL)
² [Wikipedia](http://en.wikipedia.org/wiki/MySQL) is based on MySQL. There are more than 200 million queries and 1.2 million updates per day with peak loads of 11,000 queries per second
Installing MySQL

- http://dev.mysql.com/downloads/
- We want:
  - MySQL database server & standard clients
    - MySQL 4.1 -- Generally Available (GA) release (recommended)
- This should bring us to this page:
- Scroll down to this section:

<table>
<thead>
<tr>
<th>Windows downloads</th>
<th>4.1.14</th>
<th>16.4M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Essentials (x86)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows (x86)</td>
<td>4.1.14</td>
<td>37.0M</td>
</tr>
<tr>
<td>Without installer (unzip in C:)</td>
<td>4.1.14</td>
<td>38.8M</td>
</tr>
</tbody>
</table>
Installing MySQL

- We will be downloading: mysql-essential-4.1.14-win32.msi
- Fill in the form if you want and go to the closest US mirror. The download will take a few minutes. When finished, you should have the .msi file on your desktop.
- Double-Click the newly downloaded .msi file
- Accept the typical installation
- You’ll be prompted to create a MySQL account (recommended) – monthly newsletter - save this info
- When the install finishes you’ll get a configuration option window. Be sure it is checked.
Installing MySQL

- We will be downloading: mysql-essential-4.1.14-win32.msi
- Fill in the form if you want and go to the closest US mirror. The download will take a few minutes. When finished, you should have the .msi file on your desktop.
- Double-Click the newly downloaded .msi file
- Accept the typical installation
- You’ll be prompted to create a MySQL account (recommended) – monthly newsletter - save this info
- When the install finishes you’ll get a configuration option window. Be sure it is checked.
Installing MySQL

Wizard Completed

Setup has finished installing MySQL Server 4.1. Click Finish to exit the wizard.

Configure the MySQL Server now

Use this option to generate an optimized MySQL configuration file, setup a Windows service running on a dedicated port and to set the password for the root account.
Installing MySQL

Welcome to the MySQL Server Instance Configuration Wizard 1.0.5

The Configuration Wizard will allow you to configure the MySQL Server 4.1 server instance. To continue, click Next.
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

Please select a configuration type.

- **Detailed Configuration**
  Choose this configuration type to create the optimal server setup for this machine.

- **Standard Configuration**
  Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.

< Back   Next >   Cancel
Installing MySQL

MySQL Server Instance Configuration Wizard

Configure the MySQL Server 4.1 server instance.

Please select a server type. This will influence memory, disk and CPU usage.

- **Developer Machine**
  - This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.

- **Server Machine**
  - Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.

- **Dedicated MySQL Server Machine**
  - This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 4.1 server instance.

Please select the database usage.

- **Multifunctional Database**
  General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.

- **Transactional Database Only**
  Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.

- **Non-Transactional Database Only**
  Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

InnoDB Tablespace Settings
Please select the drive for the InnoDB datafile, if you do not want to use the default settings.

InnoDB Tablespace Settings
Please choose the drive and directory where the InnoDB tablespace should be placed.

C: Installation Path

Drive Info
Volume Name: NTFS
File System:

39 GB Diskspace Used
35.6 GB Free Diskspace

< Back Next > Cancel
Installing MySQL

MySQL Server Instance Configuration

Configure the MySQL Server 4.1 server instance.

Please set the approximate number of concurrent connections to the server.

- **Decision Support (DSS)/OLAP**
  
  Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.

- **Online Transaction Processing (OLTP)**
  
  Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.

- **Manual Setting**
  
  Please enter the approximate number of concurrent connections:

  ![Input field for concurrent connections]

  Concurrent connections: 15
Installing MySQL
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

Please select the default character set.

- **Standard Character Set**
  - **Hello**
  - Makes Latin1 the default charset. This character set is suited for English and other West European languages.

- **Best Support For Multilingualism**
  - **日本語**
  - Makes UTF8 the default character set. This is the recommended character set for storing text in many different languages.

- **Manual Selected Default Character Set / Collation**
  - **?**
  - Please specify the character set to use.
  - **Character Set:** latin1

< Back  |  Next >  |  Cancel
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

Please set the Windows options.

✔️ Install As Windows Service
This is the recommended way to run the MySQL server on Windows.

Service Name: MySQL

✔️ Launch the MySQL Server automatically

✔️ Include Bin Directory in Windows PATH
Check this option to include the directory containing the server/client executables in the Windows PATH variable so they can be called from the command line.
Installing MySQL

The image shows a window for configuring MySQL Server Instance. It asks to set security options. There is an option to modify security settings.

- **New root password:** Enter the root password.
- **Confirm password:** Retype the password.
- **Enable root access from remote machines**

There is also an option to create an anonymous account. This will create an anonymous account on the server. Please note that this can lead to an insecure system.
Installing MySQL

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

Ready to execute ...

- Prepare configuration
- Write configuration file
- Start service
- Apply security settings

Please press [Execute] to start the configuration.
Installing MySQL

MySQL Server Instance Configuration
Configure the MySQL Server 4.1 server instance.

Processing configuration ...

- Prepare configuration
- Write configuration file  (C:\Program Files\MySQL\MySQL Server 4.1\my.ini)
- Start service
- Apply security settings

Configuration file created.
Windows service MySQL installed.
Service started successfully.
Security settings applied.

Press [Finish] to close the Wizard.
Installing MySQL

- Type “status” for info, then type “exit” to quit
Configuring PHP

- PHP stores all kinds of configuration information into a file called `php.ini`. Recall that we moved this to the `C:\Windows` directory.
- For now, **we do not needed to alter this file.**
- If you are interested in the *systems* side of DBMS, then read this file carefully.
- The following two slides are for reference only!
Configuring PHP

- **error_reporting** and **display_errors** – the default values that come with the installation are fine for development. When you go to production you’ll want to change to:
  ```
  error_reporting = E_NONE
  display_errors = Off
  ```
  This is because in a production environment you don’t want too much detail about your errors because it may reveal security error.

- **register_globals** – this value should be set to **Off**, which is the default, otherwise it exposes possible security problems.

- **session.save_path** – If you use sessions, something you may want to do as an advanced function, but not now, then this configuration tells PHP where to save the session data. You will need to set this value to an existing directory or you will not be able to use session. In Windows you can set this value as:
  ```
  session.save_path = C:\WINDOWS\Temp\n  ```
Configuring PHP

- **extension** – PHP4 comes with many extensions such as Java, SSL, LDAP, Oracle, etc. These are not turned on automatically. If you need to use the extension, first you need to specify the location of the extensions and then uncomment the extension you want.

For Windows you will need to uncomment the extension you want to use. In php.ini a comment is started using a semicolon (;). As an example if you want to use OpenSSL, then you must remove the semicolon at the beginning of;

```
;extension=php_openssl.dll
to
extension=php_openssl.dll
```

**Note:** MySQL and ODBC support is now built in, so no dll is needed for it.

- **max_execution_time** – the default is 30 seconds
WAMP Install Completed

◆ That’s it!
◆ You have finish installing and configuring Apache, MySQL and PHP on Windows
◆ Now we are ready to create, modify, and query tables using SQL under the Relational Model
Playing With MySQL

- Create Database
- Create, Modify, Delete Tables and Rows
- Delete Database
Homework

- Install PHP On Your System
- Install MySQL
- Create, Delete, Modify Tables
- Insert, Modify, Delete Data Into Tables
- Play with MySQL
Homework

- Read Chapter Three
- No exercises for next class; MidTerm instead
MidTerm Exam

- Due next class September 17
- No late submissions