Web Programming Step by Step

Appendix A
Database Design

Except where otherwise noted, the contents of this presentation are Copyright 2009 Marty Stepp and Jessica Miller.

Database design principles (Appendix A)

- **database design** : the act of deciding the schema for a database
- **database schema**: a description of what tables a database should have, what columns each table should contain, which columns' values must be unique, etc.
- some database design principles:
    - keep it simple, stupid (KISS)
    - provide an identifier, or **key**, by which any row can be uniquely fetched
    - eliminate redundancy, especially redundancy of lengthy data (strings)
        - when redundancy is needed, integers are smaller than strings and better to repeat

First database design

student_grades

name	email	course	grade
Bart	bart@fox.com	Computer Science 142	B-
Bart	bart@fox.com	Computer Science 143	C
Milhouse	milhouse@fox.com	Computer Science 142	B+
Lisa	lisa@fox.com	Computer Science 143	A+
Lisa	lisa@fox.com	Computer Science 190M	A+
Ralph	ralph@fox.com	Informatics 100	D+

- what's good and bad about this design?
  - contains redundancy (name, email, course repeated frequently)
  - there is no "key" column unique to each row

Second database design

students

id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

courses

id	name
10001	Computer Science 142
10002	Computer Science 143
10003	Computer Science 190M
10004	Informatics 100

grades

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

- splitting data into multiple tables avoids redundancy
- **normalizing**: splitting tables to improve structure and remove redundancy / anomalies
- normalized tables are often linked by unique integer IDs

Related tables and keys

students		
id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

courses	
id	name
10001	Computer Science 142
10002	Computer Science 143
10003	Computer Science 190M
10004	Informatics 100

grades		
student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

- records of one table may be associated with record(s) in another table
  - record in `Student` table with `student_id` of 888 is Lisa Simpson's student info
  - records in `Grade` table with `student_id` of 888 are Lisa Simpson's course grades
- **primary key**: a table column guaranteed to be unique for each record

Design question

students		
id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

courses	
id	name
10001	Computer Science 142
10002	Computer Science 143
10003	Computer Science 190M
10004	Informatics 100

grades		
student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

- suppose we want to keep track of the teachers who teach each course
  - e.g. Ms. Krabappel always teaches CSE 142 and INFO 100
  - e.g. Ms. Hoover always teaches CSE 143
  - e.g. Mr. Stepp always teaches CSE 190M
- what tables and/or columns should we add to the database?

Design answer

teachers		courses		
id	name	id	name	teacher_id
1234	Krabappel	10001	Computer Science 142	1234
5678	Hoover	10002	Computer Science 143	5678
9012	Stepp	10003	Computer Science 190M	9012
		10004	Informatics 100	1234

- add a `teachers` table containing information about instructors
- link this to courses by teacher IDs
- why not just skip the `teachers` table and put the teacher's name as a column in `courses`?
    - repeated teacher names are redundant and large in size

Entities and relationships
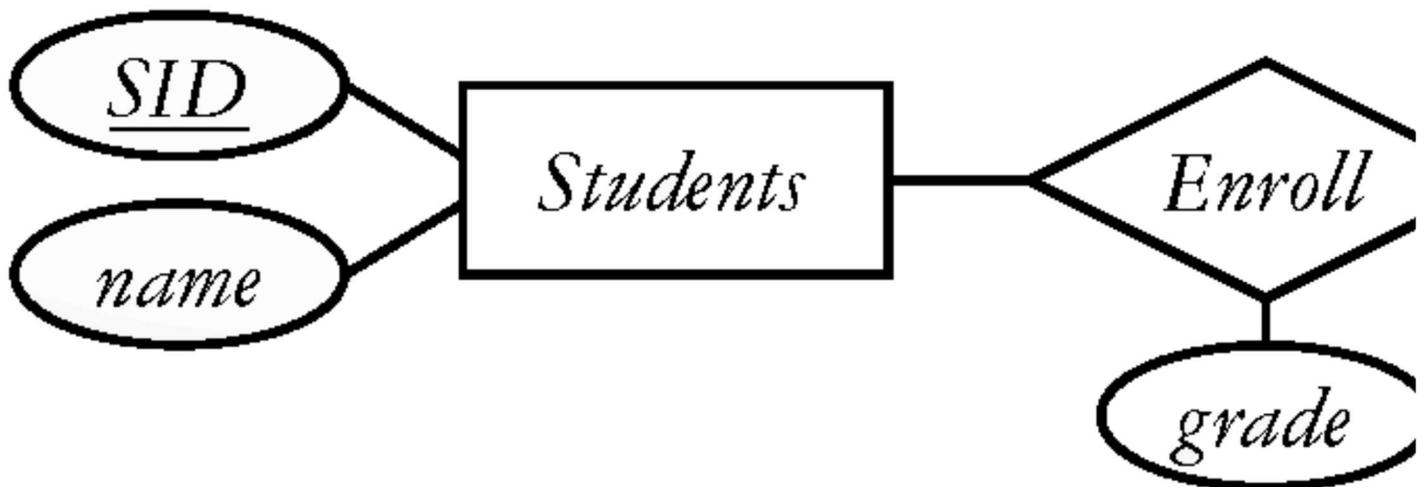
- an **entity** is a record in a table in the database
- a **relationship** is a connection between two or more entities
- database designers often draw **ER diagrams** like the above to represent the entities and relationships in their databases

The SQL `CREATE TABLE` statement (A.1.2)

```sql
CREATE TABLE name (
    columnName  type  constraints,
    ...
    columnName  type  constraints
);
```

```sql
CREATE TABLE students (
    sid INTEGER UNSIGNED NOT NULL PRIMARY KEY,
    name VARCHAR(20),
    email VARCHAR(32)
);
```

- adds/deletes an entire new table from this database
- you can add constraints such as `NOT NULL` for a field that cannot be blank or `PRIMARY KEY` for a column that must be unique for every row
- related commands: `CREATE DATABASE`, `DROP TABLE`, `ALTER TABLE`

SQL data types

- `BOOLEAN`
- `INTEGER`
- `FLOAT`
- `VARCHAR` : a string
- `DATE`, `TIME`, `DATETIME`
- `BLOB` : binary data
- quick reference