Databases & Your server

You will be creating your own tables in your database for the server

Some students run databases on their own machines for development

I will create PostgreSQL databases for this class

While you are not required to use that database the database you use must be accessible by your server when it is graded
CS 514 in 59 slides
Jargon

2-Tier

Client

Server

3-Tier

Client

Server

Database
More Jargon

Sometimes database means a program for managing data

    Oracle Corporation is a database company.
    MS Access is database.

Sometimes database means a collection of data

    I keep a database of my CD collection on 3 by 5 cards

Sometimes database means a set of tables, indexes, and views

    My program needs to connect to the Airline Reservation database, which uses Oracle
Some Reasons for Using a Database

Persistence of data

Sharing of data between programs

Handle concurrent requests for data access

Transactions that can be rolled back

Report generation
Types of Databases

Relational

Data is stored in tables

Object-Oriented

Tables can be subclassed

Programmer can define methods on tables

Object

Objects are stored in the database
Database consists of a number of tables

Table is a collection of records

Each Column of data has a type

<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>phone</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>555-9876</td>
<td>2000</td>
</tr>
<tr>
<td>Ben</td>
<td>Oker</td>
<td>555-1212</td>
<td>9500</td>
</tr>
<tr>
<td>Mary</td>
<td>Jones</td>
<td>555-3412</td>
<td>9900</td>
</tr>
</tbody>
</table>

Use Structured query language (SQL) to access data
Some Available Databases

Oracle
DB2
SQL Server
Access
Informix
Ingres
InterBase
Sybase
FileMaker Pro
FoxPro
Paradox
dBase

Open Source Databases
MySQL
PostgresSQL
SQL History

Dr. E. F. Codd develops relational database model
   Early 1970's

IBM System R relational database
   Mid 1970's
   Contained the original SQL language

First commercial database - Oracle 1979

SQL was aimed at:
   Accountants
   Business people

SQL92
   First commonly followed standard
   ANSI X3.135-1992
   SQL2

ISO/IEC 9075-1 through 5
   New SQL standard
MySQL & PostgreSQL

Open source databases

http://www.mysql.com/

http://www.postgresql.org/

Above site have free downloads and documentation

Historically I used MySql for this course. While it had some defects MySql was easier to install, easier to use on simple situations, had better documentation, and was far more common than PostgreSql. The situation has changed a lot. PostgreSql is now as easy to install, its documentation is as good as MySql, it runs faster than MySql, and has better support for transactions. There also is the issue of Sun's purchase of MySql and some lost of direction for the free version of MySql. So at least personally I have stopped using MySql and don't provided MySql accounts for students.
MySQL – Connecting to the Database

Can be done with:
   Mysql command line tool - mysql
   GUI clients
   Program

GUI Clients

If done well are very useful

There are many of these

I use DbVisualizer, & CocoaMySQL

DbVisualizer if Java based so runs on may platforms

http://www.dbvis.com/products/dbvis/
SQL Syntax

Names

Databases, tables columns & indexes have names

Legal Characters

Alphanumeric characters, '_', '$'

Names can start with:
   Letter
   Underscore
   Letter with diacritical marks and some non-latin letters

Name length

63 characters – default in PostgreSQL
64 characters - MySQL

Names are not case sensitive
## Data Types

### Numeric Values
- Integer - decimal or hex
- Floating-point - scientific & 12.1234

### String Values
- ‘this is a string’ PostgreSQL
- ‘this is a string’ “this is also a string” MySQL

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘</td>
<td>Single quote</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\n</td>
<td>Newline</td>
</tr>
<tr>
<td>\r</td>
<td>Tab</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\xxxx</td>
<td>Character were xxxx is the octal of ASCII code (PostgreSQL)</td>
</tr>
</tbody>
</table>

Including a quote character in a string
Double quote the character
'Don"t do it'

Escape the quote character with a backslash
'Don\'t do it'
Comments

-- this is a comment in MySQL and PostgreSQL

/* this is also a comment in MySQL and PostgreSQL */

# this is a comment in MySQL
# Numeric Data Types

<table>
<thead>
<tr>
<th>Type name</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallint</td>
<td>Fixed-precision</td>
<td>-32768 to +32767</td>
</tr>
<tr>
<td>integer</td>
<td>Usual choice for fixed-precision</td>
<td>-2147483648 to +2147483647</td>
</tr>
<tr>
<td>bigint</td>
<td>Very large range fixed-precision</td>
<td>-9223372036854775808 to 9223372036854775807</td>
</tr>
<tr>
<td>decimal</td>
<td>user-specified precision, exact</td>
<td>no limit</td>
</tr>
<tr>
<td>numeric</td>
<td>user-specified precision, exact</td>
<td>no limit</td>
</tr>
<tr>
<td>real</td>
<td>variable-precision, inexact</td>
<td>6 decimal digits precision</td>
</tr>
<tr>
<td>double precision</td>
<td>variable-precision, inexact</td>
<td>15 decimal digits precision</td>
</tr>
<tr>
<td>serial</td>
<td>autoincrementing integer</td>
<td>1 to 2147483647</td>
</tr>
</tbody>
</table>

Numeric(10, 2) defines a number with maximum of 10 digits with 2 of the 10 to the right of the decimal point

```
12345678.91
```

decimal and numeric are different names for the same type
String Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char(n)</td>
<td>Fixed-length blank padded</td>
</tr>
<tr>
<td>varchar(n)</td>
<td>Variable-length with limit</td>
</tr>
<tr>
<td>text</td>
<td>Variable unlimited length</td>
</tr>
<tr>
<td>bytea (PostgreSQL)</td>
<td>Variable (not specifically limited) length binary string</td>
</tr>
<tr>
<td>blob (MySQL)</td>
<td>Variable (not specifically limited) length binary string</td>
</tr>
</tbody>
</table>

CHAR & VARCHAR are the most common string types

CHAR is fixed-width
   Shorter strings are padded

TEXT can be any size

PostgreSQL limits a string to 1GB in storage space

MySQL limits CHAR and VARCHAR to 255 characters
## Date & Time Types - PostgreSQL

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp [(p)] without time zone</td>
<td>both date and time</td>
</tr>
<tr>
<td>timestamp [(p)] [ with time zone ]</td>
<td>both date and time</td>
</tr>
<tr>
<td>interval [(p)]</td>
<td>for time intervals</td>
</tr>
<tr>
<td>date</td>
<td>dates only</td>
</tr>
<tr>
<td>time [(p)] [ without time zone ]</td>
<td>times of day only</td>
</tr>
<tr>
<td>time [(p)] [ with time zone ]</td>
<td>times of day only</td>
</tr>
</tbody>
</table>

(p) indicates optional number of fractional digits retained in the seconds field
### Date Formats - PostgreSQL

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 8, 1999</td>
<td>Unambiguous</td>
</tr>
<tr>
<td>1999-01-08</td>
<td>ISO-8601 format, preferred</td>
</tr>
<tr>
<td>1/8/1999</td>
<td>U.S.; read as August 1 in European mode</td>
</tr>
<tr>
<td>8/1/1999</td>
<td>European; read as August 1 in U.S. mode</td>
</tr>
<tr>
<td>1/18/1999</td>
<td>U.S.; read as January 18 in any mode</td>
</tr>
<tr>
<td>19990108</td>
<td>ISO-8601 year, month, day</td>
</tr>
<tr>
<td>990108</td>
<td>ISO-8601 year, month, day</td>
</tr>
<tr>
<td>1999.008</td>
<td>Year and day of year</td>
</tr>
<tr>
<td>99008</td>
<td>Year and day of year</td>
</tr>
<tr>
<td>J2451187</td>
<td>Julian day</td>
</tr>
<tr>
<td>January 8, 99 BC</td>
<td>Year 99 before the Common Era</td>
</tr>
</tbody>
</table>
Setting Date Formats - PostgreSQL

SET DateStyle TO 'US'
SET DateStyle TO 'NonEuropean'

Sets date format to month day year

SET DateStyle TO 'European'

Sets date format to day month year

Default is ISO style
Dates – MySQL

DATETIME – ‘YYYY-MM-DD HH:MM:SS’ format

DATE – ‘YYYY-MM-DD’ format

TIMESTAMP
   Changed in MySQL 4.1
   Basically now is same as DATETIME
## Common SQL Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Retrieves data from table(s)</td>
</tr>
<tr>
<td>INSERT</td>
<td>Adds row(s) to a table</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Changes field(s) in record(s)</td>
</tr>
<tr>
<td>DELETE</td>
<td>Removes row(s) from a table</td>
</tr>
<tr>
<td>CREATE TABLE</td>
<td>Define a table and its columns(fields)</td>
</tr>
<tr>
<td>DROP TABLE</td>
<td>Deletes a table</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>Adds a new column, add/drop primary key</td>
</tr>
<tr>
<td>CREATE INDEX</td>
<td>Create an index</td>
</tr>
<tr>
<td>DROP INDEX</td>
<td>Deletes an index</td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>Define a logical table from other table(s)/view(s)</td>
</tr>
<tr>
<td>DROP VIEW</td>
<td>Deletes a view</td>
</tr>
</tbody>
</table>

SQL is not case sensitive
Examples That Follow

Will use mysql command line tool

Used the command

```
mysql -h host -u user -p
```

to connect to the database, where host and user are given the correct value

On rohan the full name of command is:

```
/opt/local/mysql/bin/mysql
```

Some examples will also show postgresSQL text client
CREATE DATABASE

General Form
CREATE DATABASE [IF NOT EXISTS] db_name
    [create_specification [, create_specification] ...]

create_specification:
    [DEFAULT] CHARACTER SET charset_name
| [DEFAULT] COLLATE collation_name

Example

mysql> create database lectureExamples;
Query OK, 1 row affected (0.00 sec)
Al 15-> psql -h bismarck.sdsu.edu cs580whitney cs580whitney
Password:
Welcome to psql 7.4, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
    \h for help with SQL commands
    \? for help on internal slash commands
    \g or terminate with semicolon to execute query
    \q to quit

cs580whitney=> create database lectureExamples;
ERROR: permission denied to create database

Student accounts do not have authority to create new databases
USE

Sets a default database for subsequent queries

General Form

USE db_name

Example

mysql> use lectureExamples;
Database changed
# CREATE table

## General Form

```
CREATE TABLE table_name ( 
    col_name  col_type [ NOT NULL | PRIMARY KEY]
    [, col_name col_type 
        [ NOT NULL | PRIMARY KEY]]*
)
```

## Example

```
mysql> CREATE TABLE students 
    
    ( 
        firstname  CHAR(20) NOT NULL, 
        lastname   CHAR(20), 
        phone      CHAR(10), 
        code       INTEGER 
    );

mysql> CREATE TABLE codes 
    
    ( 
        code       INTEGER, 
        name       CHAR(20) 
    );
```
PostgreSQL Example

cs580whitney=> CREATE TABLE students
   (firstname  CHAR(20) NOT NULL,
    lastname   CHAR(20),
    phone      CHAR(10),
    code       INTEGER
   );

CREATE TABLE

   cs580whitney=> select * from students;
   firstname | lastname | phone | code
   -----------+----------+-------+------
   (0 rows)
Select

Gets data from one or more tables

General Form

SELECT [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
    [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE]
    [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
    [DISTINCT | DISTINCTROW | ALL]
select_expression,...
[INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
[FROM table_references
    [WHERE where_definition]
[GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]
    [WITH ROLLUP]]
[HAVING where_definition]
[ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]
[LIMIT [offset,) row_count | row_count OFFSET offset]
[PROCEDURE procedure_name(argument_list)]
[FOR UPDATE | LOCK IN SHARE MODE]
Select Example

mysql> SELECT * FROM students;
Empty set (0.00 sec)
Insert

Add data to a table

General Form

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
INTO tbl_name [(col_name,...)]
VALUES ((expression | DEFAULT),...),(...),...
[ ON DUPLICATE KEY UPDATE col_name=expression, ... ]
Insert Examples

```sql
mysql> INSERT
    INTO students (firstname, lastname, phone, code)
    VALUES ('Roger', 'Whitney', '594-3535', 2000);

mysql> INSERT
    INTO codes (code, name)
    VALUES (2000, 'marginal');

mysql> SELECT * FROM students;
+-----------+----------+----------+------+
| firstname | lastname | phone    | code |
+-----------+----------+----------+------+
| Roger     | Whitney  | 594-3535 | 2000 |
+-----------+----------+----------+------+
1 row in set (0.01 sec)
```
More Select Examples

```
mysql> SELECT firstname , phone FROM students;
+-----------+----------+
| firstname | phone    |
+-----------+----------+
| Roger     | 594-3535 |
+-----------+----------+
1 row in set (0.00 sec)

mysql> SELECT lastname, name
    FROM students, codes
    WHERE students.code = codes.code;
+----------+----------+
| lastname | name     |
+----------+----------+
| Whitney  | marginal |
+----------+----------+
1 row in set (0.00 sec)
```
More Select Examples

mysql> SELECT students.lastname, codes.name
    FROM students, codes
    WHERE students.code = codes.code;

+----------+----------+
| lastname | name     |
+----------+----------+
| Whitney  | marginal |
+----------+----------+
1 row in set (0.00 sec)
Update

Modify existing data in a database

General Form

UPDATE [LOW_PRIORITY] [IGNORE] tbl_name [, tbl_name ...]  
SET col_name1=expr1 [, col_name2=expr2 ...]  
[WHERE where_definition]

Example

mysql> UPDATE students  
    SET firstname='Sam'  
    WHERE lastname='Whitney';

Query OK, 1 row affected (0.00 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
Few More SQL Commands

mysql> ALTER TABLE students ADD column foo CHAR(40);
Query OK, 1 row affected (0.03 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> DROP TABLE students;
Query OK, 0 rows affected (0.01 sec)

mysql> DROP DATABASE lectureexamples;
Query OK, 0 rows affected (0.00 sec)
An Example

PostgreSQL Version
CREATE TABLE faculty (  
    name  CHAR(20) NOT NULL,  
    faculty_id  SERIAL  PRIMARY KEY  
) ;

MySQL Version
CREATE TABLE faculty (  
    name  CHAR(20) NOT NULL,  
    faculty_id  INTEGER AUTO_INCREMENT  PRIMARY KEY  
) ;

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
</tbody>
</table>
Indices

Indices make accessing faster

Primary keys automatically have an index

The CREATE INDEX command creates indices

```sql
CREATE INDEX faculty_name_key on faculty (name);
```
Adding Values

INSERT INTO faculty (name) VALUES ('Whitney');
INSERT INTO faculty (name) VALUES ('Beck');
INSERT INTO faculty (name) VALUES ('Anantha');
INSERT INTO faculty (name) VALUES ('Vinge');

SELECT * FROM faculty;

Result

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
<tr>
<td>Vinge</td>
<td>4</td>
</tr>
</tbody>
</table>

(4 rows)
### Second Table

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
<th>faculty_id</th>
<th>office_hour_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00</td>
<td>11:00</td>
<td>Wed</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8:00</td>
<td>12:00</td>
<td>Mon</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17:00</td>
<td>18:30</td>
<td>Tue</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Tue</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Thu</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>15:00</td>
<td>16:00</td>
<td>Fri</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
<tr>
<td>Vinge</td>
<td>4</td>
</tr>
</tbody>
</table>
Generating Second Table

PostgreSQL
CREATE TABLE office_hours (  
    start_time    TIME NOT NULL,  
    end_time    TIME NOT NULL,  
    day    CHAR(3) NOT NULL,  
    faculty_id INTEGER REFERENCES faculty,  
    office_hour_id    SERIAL    PRIMARY KEY  
);  

MySQL
CREATE TABLE office_hours (  
    start_time    TIME NOT NULL,  
    end_time    TIME NOT NULL,  
    day    CHAR(3) NOT NULL,  
    faculty_id INTEGER REFERENCES faculty,  
    office_hour_id INTEGER AUTO_INCREMENT PRIMARY KEY  
);
Adding Office Hours

Simple Insert

```
INSERT
    INTO office_hours ( start_time, end_time, day, faculty_id )
VALUES ( '10:00:00', '11:00:00', 'Wed', 1 );
```

The problem is that we need to know the id for the faculty
Adding Office Hours

Using Select

```
INSERT INTO office_hours (start_time, end_time, day, faculty_id)
SELECT '8:00:00' AS start_time,
     '12:00:00' AS end_time,
     'Mon' AS day,
     faculty_id AS faculty_id
FROM faculty
WHERE name = 'Beck'
```
Selecting Office Hours

```sql
SELECT
    name, start_time, end_time, day
FROM
    office_hours, faculty
WHERE
    faculty.faculty_id = office_hours.faculty_id;
```

<table>
<thead>
<tr>
<th>name</th>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00:00</td>
<td>11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Beck</td>
<td>08:00:00</td>
<td>12:00:00</td>
<td>Mon</td>
</tr>
<tr>
<td>Whitney</td>
<td>17:00:00</td>
<td>18:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00</td>
<td>16:00:00</td>
<td>Fri</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00</td>
<td>10:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00</td>
<td>10:30:00</td>
<td>Thu</td>
</tr>
</tbody>
</table>
SELECT
    name AS Instructor,
    TEXT(start_time) || ' to ' || TEXT(end_time) AS Time,
    day AS Day
FROM
    office_hours, faculty
WHERE
    faculty.faculty_id = office_hours.faculty_id
ORDER BY
    Name

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Time</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anantha</td>
<td>09:00:00 to 10:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Anantha</td>
<td>09:00:00 to 10:30:00</td>
<td>Thu</td>
</tr>
<tr>
<td>Beck</td>
<td>08:00:00 to 12:00:00</td>
<td>Mon</td>
</tr>
<tr>
<td>Whitney</td>
<td>10:00:00 to 11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Whitney</td>
<td>17:00:00 to 18:30:00</td>
<td>Tue</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00 to 16:00:00</td>
<td>Fri</td>
</tr>
</tbody>
</table>
Sample Selection

```
SELECT
    name, start_time, end_time, day
FROM
    office_hours, faculty
WHERE
    faculty.faculty_id = office_hours.faculty_id
AND
    start_time > '09:00:00'
AND
    end_time < '16:30:00'
ORDER BY
    Name;
```

<table>
<thead>
<tr>
<th>name</th>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00:00</td>
<td>11:00:00</td>
<td>Wed</td>
</tr>
<tr>
<td>Whitney</td>
<td>15:00:00</td>
<td>16:00:00</td>
<td>Fri</td>
</tr>
</tbody>
</table>
## Joins

### People

<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Roger</td>
<td>Whitney</td>
</tr>
<tr>
<td>2</td>
<td>Leland</td>
<td>Beck</td>
</tr>
<tr>
<td>3</td>
<td>Carl</td>
<td>Eckberg</td>
</tr>
</tbody>
</table>

### Email_Addresses

<table>
<thead>
<tr>
<th>id</th>
<th>user_name</th>
<th>host</th>
<th>person_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>foo</td>
<td>rohan.sdsu.edu</td>
<td></td>
</tr>
</tbody>
</table>
**Inner Join**

Only uses entries linked in two tables

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
</tbody>
</table>

```
select first_name, last_name, user_name, host
from people, email_addresses
where people.id = email_addresses.person_id;
```

```
select first_name, last_name, user_name, host
from people inner join email_addresses
on (people.id = email_addresses.person_id);
```
Outer Left Join

Use all entries from the left table

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
<tr>
<td>Carl</td>
<td>Eckberg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sql
select
    first_name, last_name, user_name, host
from
    people left outer join email_addresses
on
    (people.id = email_addresses.person_id);
```
### Right Outer Join

Use all entries from the right table

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>user_name</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leland</td>
<td>Beck</td>
<td>beck</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>cs.sdsu.edu</td>
</tr>
<tr>
<td>Roger</td>
<td>Whitney</td>
<td>whitney</td>
<td>rohan.sdsu.edu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>foo</td>
<td>rohan.sdsu.edu</td>
</tr>
</tbody>
</table>

```sql
select
    first_name, last_name, user_name, host
from
    people right outer join email_addresses
on
    (people.id = email_addresses.person_id);
```
A right outer join B == B left outer join A

The following two statements are equivalent

```
select first_name, last_name, user_name, host from people right outer join email_addresses on (people.id = email_addresses.person_id);
select first_name, last_name, user_name, host from email_addresses left outer join people on (people.id = email_addresses.person_id);
```
Normal forms

Defined by Dr. E. F. Codd in 1970

Reduce redundant data and inconsistencies
First Normal Form (1NF)

An entity is in the first normal form when all its attributes are single valued.

<table>
<thead>
<tr>
<th>Name</th>
<th>OfficeHour1</th>
<th>OfficeHour2</th>
<th>OfficeHour3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>10:00-11:00 W</td>
<td>17:00-18:30 Tu</td>
<td>15:00-16:00 Fri</td>
</tr>
<tr>
<td>Beck</td>
<td>8:00-12:00 M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ananthaa</td>
<td>9:00-10:30 Tu</td>
<td>9:00-10:30 Thu</td>
<td></td>
</tr>
</tbody>
</table>

What if someone has more than 3 office hours?
Wasted space for those that have fewer office hours.

Not is 1NF since office hours are repeated.
## In 1NF

### Faculty

<table>
<thead>
<tr>
<th>name</th>
<th>faculty_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whitney</td>
<td>1</td>
</tr>
<tr>
<td>Beck</td>
<td>2</td>
</tr>
<tr>
<td>Anantha</td>
<td>3</td>
</tr>
</tbody>
</table>

### Office Hours

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>day</th>
<th>faculty_id</th>
<th>office_hour_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00</td>
<td>11:00</td>
<td>Wed</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8:00</td>
<td>12:00</td>
<td>Mon</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17:00</td>
<td>18:30</td>
<td>Tue</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Tue</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9:00</td>
<td>10:30</td>
<td>Thu</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>15:00</td>
<td>16:00</td>
<td>Fri</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
## Second Normal Form (2NF)

<table>
<thead>
<tr>
<th>cd_title</th>
<th>artist</th>
<th>music_type</th>
<th>cd_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Songs from the Trilogy</td>
<td>Glass</td>
<td>Modern Classical</td>
<td>1</td>
</tr>
<tr>
<td>I Stoten</td>
<td>Falu Spelmanslag</td>
<td>Swedish</td>
<td>2</td>
</tr>
<tr>
<td>Photographer</td>
<td>Glass</td>
<td>Modern Classical</td>
<td>3</td>
</tr>
</tbody>
</table>

An entity is in the second normal form if:

- It is in 1NF and
- All non-key attributes must be fully dependent on the entire primary key

Table is not in 2NF since different CDs

- Can have the same artists
- Can have same music type
### Example 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>Days</th>
<th>Term</th>
<th>Schedule Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS635</td>
<td>1700-1815</td>
<td>MW</td>
<td>Spring01</td>
<td>9461</td>
</tr>
<tr>
<td>CS651</td>
<td>1700-1815</td>
<td>MW</td>
<td>Spring01</td>
<td>9472</td>
</tr>
<tr>
<td>CS672</td>
<td>1700-1815</td>
<td>MW</td>
<td>Spring01</td>
<td>9483</td>
</tr>
<tr>
<td>CS683</td>
<td>1830-1945</td>
<td>MW</td>
<td>Spring01</td>
<td>9494</td>
</tr>
<tr>
<td>CS696</td>
<td>1530-1645</td>
<td>MW</td>
<td>Spring01</td>
<td>9505</td>
</tr>
<tr>
<td>CS696</td>
<td>1830-1945</td>
<td>MW</td>
<td>Spring01</td>
<td>9516</td>
</tr>
<tr>
<td>CS696</td>
<td>1530-1645</td>
<td>TTh</td>
<td>Spring01</td>
<td>9520</td>
</tr>
</tbody>
</table>

At SDSU the schedule number uniquely identifies a course in a semester. So the term and schedule number uniquely identifies a course at SDSU. We can use term and schedule as the primary key.

The table is in 1NF but not 2NF.

Name, Time and Days are not fully dependent on the primary key.
## Schedule in 2NF

### Schedule

<table>
<thead>
<tr>
<th>course_id</th>
<th>time_id</th>
<th>term_id</th>
<th>schedule_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>9461</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>9472</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>9483</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>9494</td>
</tr>
</tbody>
</table>

### Courses

<table>
<thead>
<tr>
<th>course</th>
<th>title</th>
<th>name_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS635</td>
<td>Adv Obj Orient Dsgn Prog</td>
<td>1</td>
</tr>
<tr>
<td>CS651</td>
<td>Adv Multimedia Systems</td>
<td>2</td>
</tr>
<tr>
<td>CS683</td>
<td>Emerging Technologies</td>
<td>3</td>
</tr>
<tr>
<td>CS696</td>
<td>Writing Device Drivers</td>
<td>4</td>
</tr>
</tbody>
</table>

### Time

<table>
<thead>
<tr>
<th>start_time</th>
<th>end_time</th>
<th>days</th>
<th>time_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:00:00</td>
<td>18:15:00</td>
<td>MW</td>
<td>1</td>
</tr>
<tr>
<td>18:30:00</td>
<td>19:45:00</td>
<td>MW</td>
<td>2</td>
</tr>
<tr>
<td>15:30:00</td>
<td>16:45:00</td>
<td>MW</td>
<td>3</td>
</tr>
<tr>
<td>15:30:00</td>
<td>16:45:00</td>
<td>TTh</td>
<td>4</td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Term

<table>
<thead>
<tr>
<th>semester</th>
<th>year</th>
<th>term_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall</td>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>Spring</td>
<td>2001</td>
<td>2</td>
</tr>
</tbody>
</table>
Comments about Previous Slide

The schedule table is now in 2NF

What about the other tables?

If not how would you fix them?

Can you find a better way to decompose the original table?
An entity is in third normal form if

- It is in 2NF and
  - All non-key attributes must only be dependent on the primary key

State abbreviation depends on State Name

Table is not in 3NF