CS 580 Client-Server Programming
Spring Semester, 2009
Doc 15 SQL
18 March, 2010

Databases & Your server

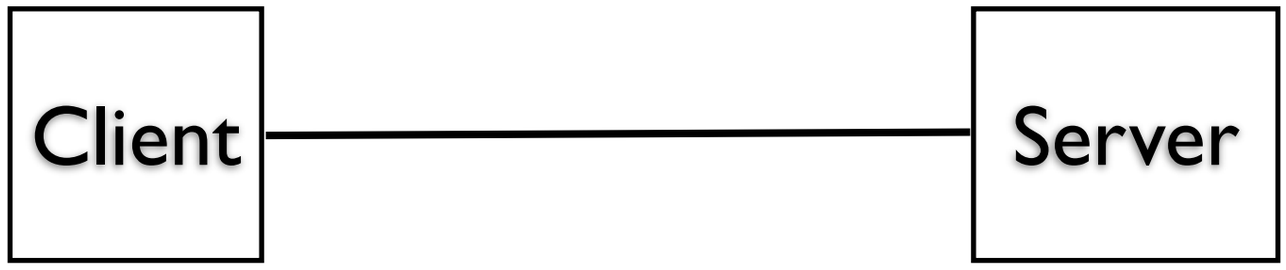
You will be creating your own tables in your database for the server

We will be using SQLite for the database

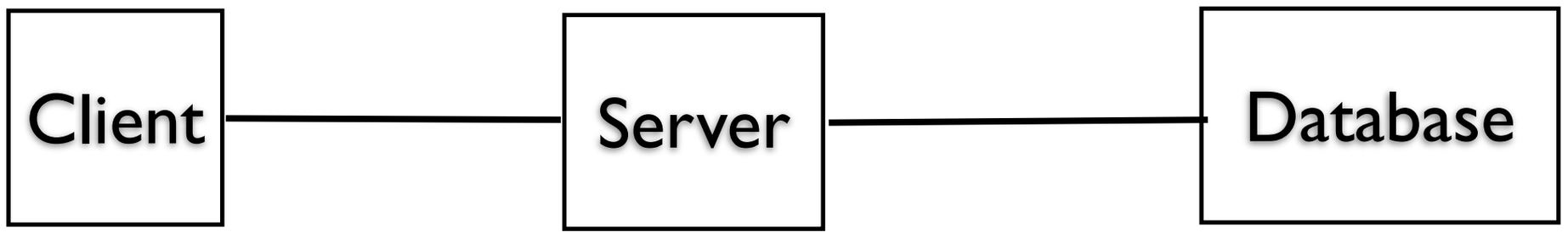
CS 514 in 51 slides

Jargon

2-Tier



3-Tier



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

In the Beginning - Relational Databases

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

Object Databases were there too

Objects are stored in the database

Research into databases fo graph structured databases

Early to mid 1970s

Oobject-oriented database term first used

1985

First commercial OO database system

1986

Relational Databases dominated Market

Relational databases standard

DB Administrators make lots money

Oracle makes ton of money

MySQL & PostgresSQL open source databases become popular

But Some Were not Happy

Large data sets not handled well

- Think Google

SQL databases not flexible enough

NoSQL Databases

Hadoop/HBase	Document Store
Cassandra	Key/Value Store
CouchDB	Eventually-Consistent Key-Value Store
MongoDB	
Amazon SimpleDB	
MemcacheDB	

Relational Databases and SQL

Database consists of a number of tables

Table is a collection of records

Each Column of data has a type

firstname	lastname	phone	code
John	Smith	555-9876	2000
Ben	Oker	555-1212	9500
Mary	Jones	555-3412	9900

Use Structured query language (SQL) to access data

Some Available Relational Databases

Commercial

Oracle

DB2

SQL Server

Access

Informix

Ingres

InterBase

Sybase

FileMaker Pro

FoxPro

Paradox

dBase

Open Source

MySQL

PostgresSQL

Public Domain

SQLite

MySQL, PostgreSQL, SQLite

Open source databases

http://www.sqlite.org/

http://www.mysql.com/

http://www.postgresql.org/

Above site have free downloads and documentation

SQLite & Clients

SQLite is embedded into your application

Use JDBC to access the database

SQLite GUI clients

Firefox https://addons.mozilla.org/en-US/firefox/addon/5817

Creating a Table

```
CREATE TABLE SampleTable (
    name text UNIQUE,
    age integer,
    isStudent boolean,
    description
)
```

name	age	isStudent	description

Adding Data

```
insert into SampleTable values(
    'Donald Knuth',
    72,
    0,
    'Computer Science deity'
)
```

select * from SampleTable

name	age	isStudent	description
Donald Knuth	72	0	

SQLite Datatypes

Storage Classes

NULL

INTEGER

Signed integer in 1, 2, 3, 4, 6, or 8 bytes

REAL

8-byte IEEE floating point number

TEXT

Text stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE)

BLOB

Stored as it is entered into the database

SQLite Datatypes

Dynamic Typing

Any column may be used to store a value of any storage class
except an INTEGER PRIMARY KEY column

So what happens when you insert text into an integer column?

SQLite Column Affinity

Each column has its preferred datatype (affinity)

if data can be converted losslessly

SQLite stores data using the column preferred datatype

else stores type as it is

SQLite Affinity

SQLType	SQLite Affinity
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB	TEXT

SQLType	SQLite Affinity
BLOB no datatype specified	NONE
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC

Common SQL Statements

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

SQL is not case sensitive

CREATE table

General Form

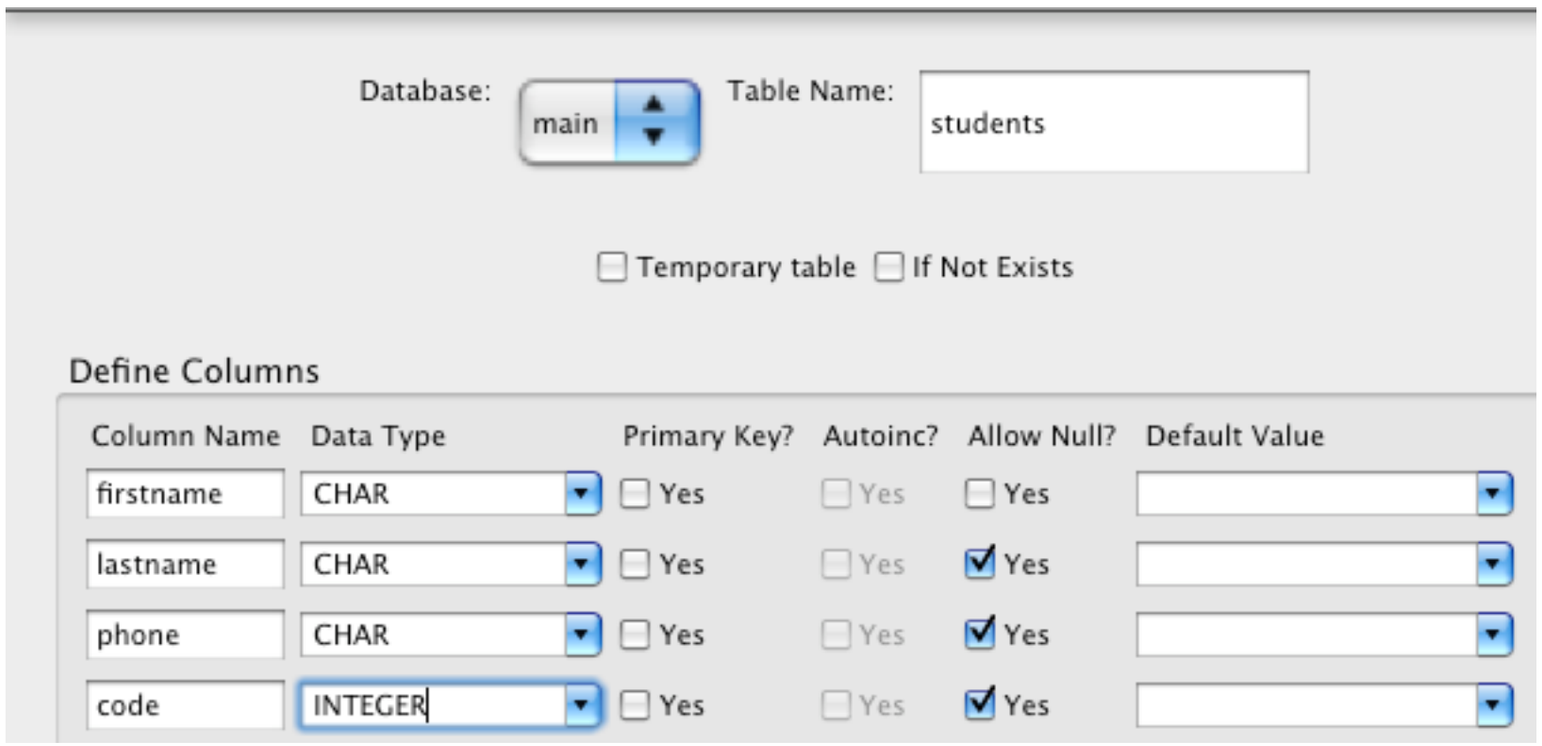
```
CREATE TABLE table_name (
    col_name  col_type [ NOT NULL | PRIMARY KEY]
    [, col_name col_type [ NOT NULL | PRIMARY KEY]]*
)
```

Example

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

```
CREATE TABLE codes
      (
      code       INTEGER,
      name       CHAR(20)
      )
```

SQLite Firefox Client




```
CREATE  TABLE "main"."students" ("firstname" CHAR NOT
NULL , "lastname" CHAR, "phone" CHAR, "code" INTEGER)
```

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, ... ]
```

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]]
```

Insert Examples

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

INSERT
    INTO codes (code, name)
    VALUES (2000, 'marginal' )
```

```
SELECT * FROM students;
+----------+---------+---------+------+
| firstname | lastname | phone     | code |
+----------+---------+---------+------+
| Roger     | Whitney  | 594-3535 | 2000 |
+----------+---------+---------+------+
```

More Select Examples

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

```
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

```
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]

Update Example

```
UPDATE students
    SET firstname='Sam'
    WHERE lastname='Whitney'
```

Few More SQL Commands

ALTER TABLE students ADD column foo CHAR(40);

DROP TABLE students;

An Example

name	faculty_id
Whitney	1
Beck	2
Anantha	3

CREATE TABLE "faculty" ("name" VARCHAR NOT NULL , "faculty_id" INTEGER PRIMARY KEY AUTOINCREMENT)

Indices

Indices make accessing faster

Primary keys automatically have an index

The CREATE INDEX command creates indices

```
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    ('Lewis')
INSERT INTO faculty ( name) VALUES ('Eckberg')
```

```
select * from faculty;
```

Result

```
          name            | faculty_id
--------------------------+------------
 Whitney                  |          1
 Beck                     |          2
 Lewis                    |          3
 Eckberg                  |          4
(4 rows)
```

Second Table

start_time	end_time	day	faculty_id	office_hour_id
10:00	11:00	Wed	1	1
8:00	12:00	Mon	2	2
17:00	18:30	Tue	1	3
9:00	10:30	Tue	3	4
9:00	10:30	Thu	3	5
15:00	16:00	Fri	1	6

name	faculty_id
Whitney	1
Beck	2
Lewis	3
Eckberg	4

Generating Second Table

```
CREATE TABLE "office_hours" (
    "start_time" TEXT NOT NULL ,
    "end_time" TEXT NOT NULL ,
    "day" TEXT NOT NULL ,
    "faculty_id" INTEGER NOT NULL  check(typeof("faculty_id") = 'integer') ,
    "office_hour_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL
    )
```

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

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

name	start_time	end_time	day
Whitney	10:00:00	11:00:00	Wed
Beck	08:00:00	12:00:00	Mon
Whitney	17:00:00	18:30:00	Tue
Whitney	15:00:00	16:00:00	Fri
Lewis	09:00:00	10:30:00	Tue
Eckberg	09:00:00	10:30:00	Thu

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;
```

name	start_time	end_time	day
Whitney	10:00:00	11:00:00	Wed
Whitney	15:00:00	16:00:00	Fri

Joins

People

id	first_name	last_name
1	Roger	Whitney
2	Leland	Beck
3	Carl	Eckberg

Email_Addresses

id	user_name	host	person_id
1	beck	cs.sdsu.edu	2
2	whitney	cs.sdsu.edu	1
3	whitney	rohan.sdsu.edu	1
4	foo	rohan.sdsu.edu	

Inner Join

Only uses entries linked in two tables

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu

```
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

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
Carl	Eckberg		

```
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 - Not supported in SQLite

first_name	last_name	user_name	host
Leland	Beck	beck	cs.sdsu.edu
Roger	Whitney	whitney	cs.sdsu.edu
Roger	Whitney	whitney	rohan.sdsu.edu
		foo	rohan.sdsu.edu

```
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

Name	OfficeHour1	OfficeHour2	OfficeHour3
Whitney	10:00-11:00 W	17:00-18:30 Tu	15:00-16:00 Fri
Beck	8:00-12:00 M		
Anantha	9:00-10:30 Tu	9:00-10:30 Thu	

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

name	faculty_id
Whitney	1
Beck	2
Anantha	3

Office Hours

start_time	end_time	day	faculty_id	office_hour_id
10:00	11:00	Wed	1	1
8:00	12:00	Mon	2	2
17:00	18:30	Tue	1	3
9:00	10:30	Tue	3	4
9:00	10:30	Thu	3	5
15:00	16:00	Fri	1	6

Second Normal Form (2NF)

cd_title	artist	music_type	cd_id
Songs from the Trilogy	Glass	Modern Classical	1
I Stoten	Falu Spelmanslag	Swedish	2
Photographer	Glass	Modern Classical	3

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

Name	Time	Days	Term	Schedule Number
CS635	1700-1815	MW	Spring01	9461
CS651	1700-1815	MW	Spring01	9472
CS672	1700-1815	MW	Spring01	9483
CS683	1830-1945	MW	Spring01	9494
CS696	1530-1645	MW	Spring01	9505
CS696	1830-1945	MW	Spring01	9516
CS696	1530-1645	TTh	Spring01	9520

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

course_id	time_id	term_id	schedule_number
1	1	2	9461
2	1	2	9472
3	1	2	9483
4	2	2	9494

Term

semester	year	term_id
Fall	2000	1
Spring	2001	2

Courses

course	title	name_id
CS635	Adv Obj Orient Dsgn Prog	1
CS651	Adv Multimedia Systems	2
CS683	Emerging Technologies	3
CS696	Writing Device Drivers	4

Time

start_time	end_time	days	time_id
17:00:00	18:15:00	MW	1
18:30:00	19:45:00	MW	2
15:30:00	16:45:00	MW	3
15:30:00	16:45:00	TTh	4
Etc.			

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?

Third Normal Form (3NF)

Customer

Name	Address	City	State Name	State abbreviation	zip	id

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