

# INF1343, Week 6

## Implementing a Database with SQL

Yuri Takhteyev  
University of Toronto  
February 7, 2011



This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

# Normal Forms

5<sup>th</sup> Normal Form

4<sup>th</sup> Normal Form

BC Normal Form

3<sup>rd</sup> Normal Form

2<sup>nd</sup> Normal Form

1<sup>st</sup> Normal Form

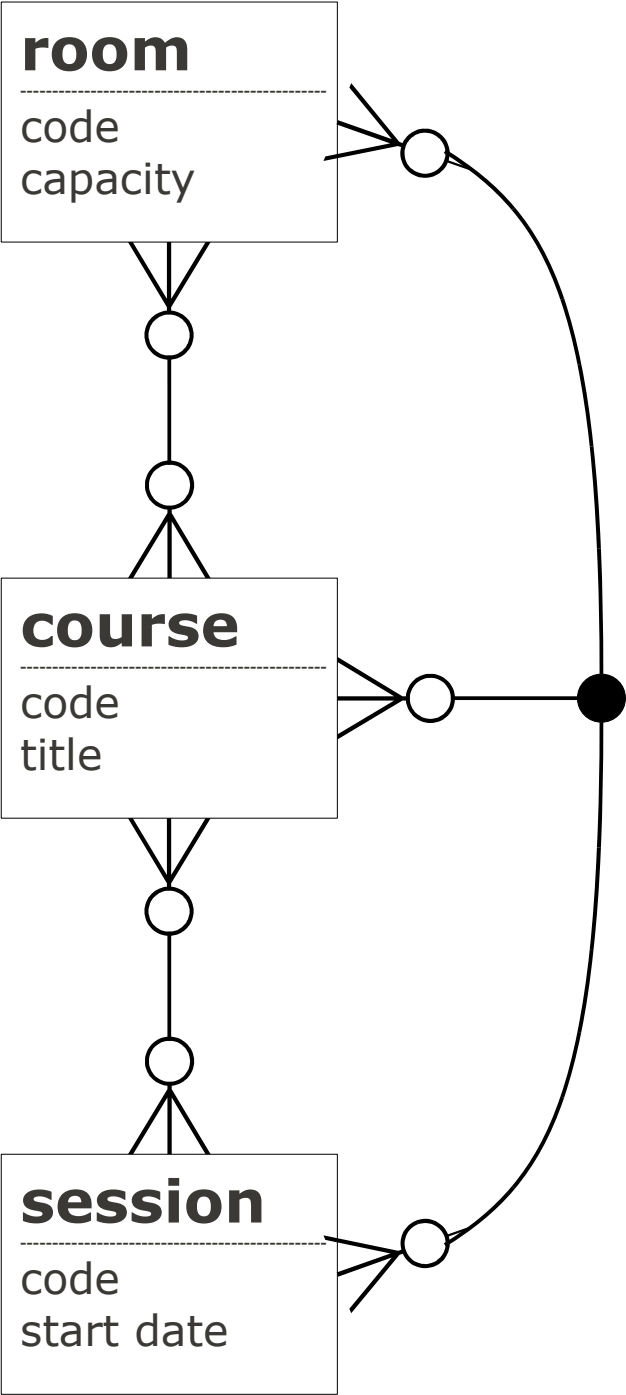
fixing  
weird  
issues

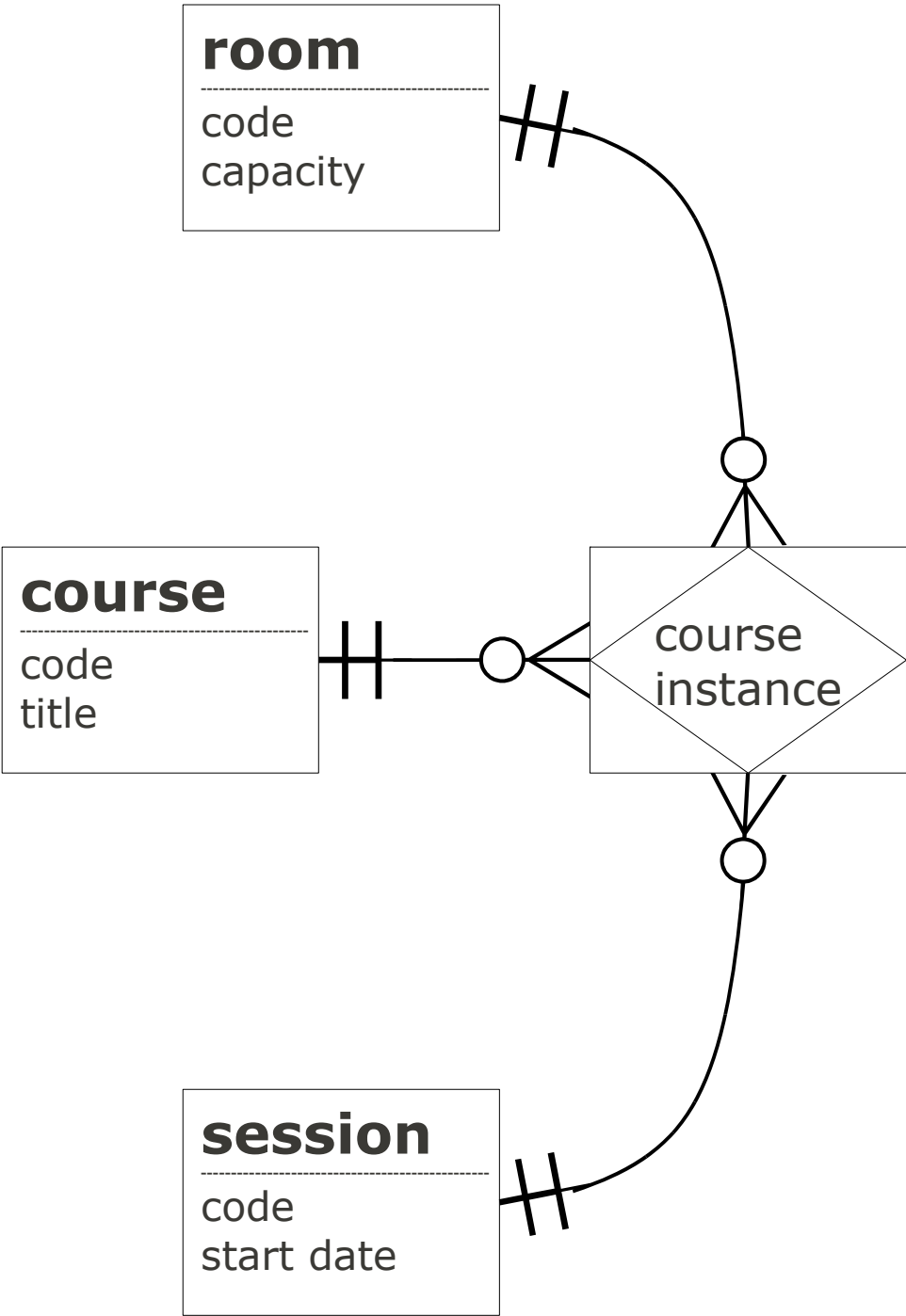
you can do it!

trivial!

# 4NF

Mistaking binary relationships  
for ternary



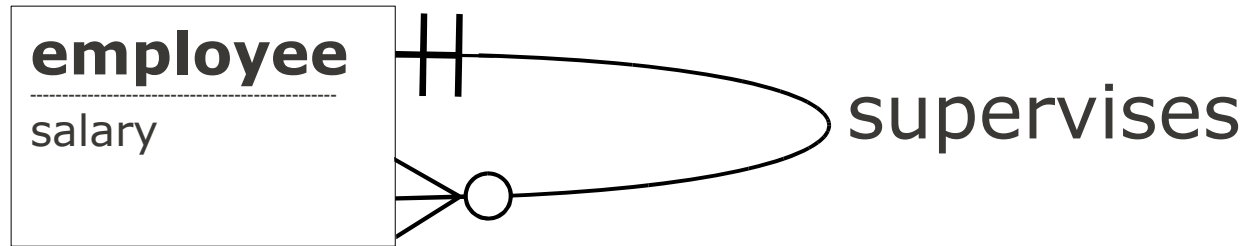


<b>name</b>
student_id
course_id
internship_id

# 5NF

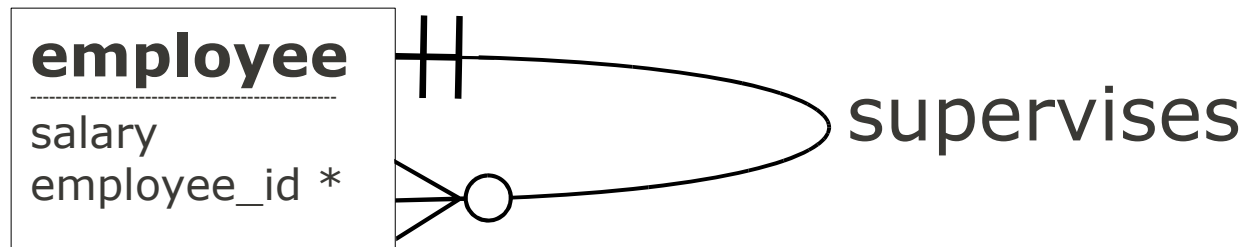
“We could break it up yet more”

# Recursive Relationships





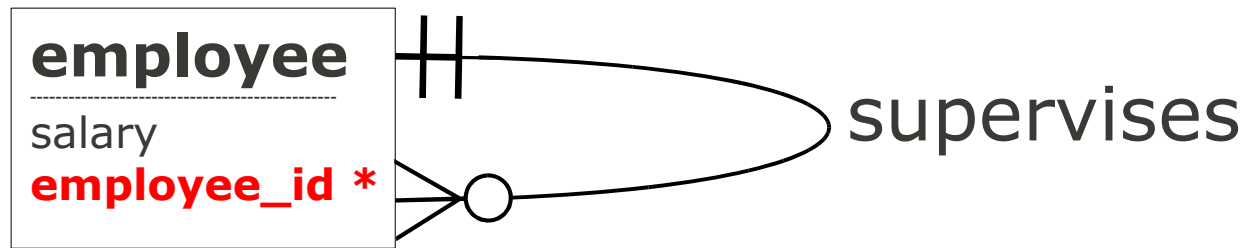
# Recursive Relationships



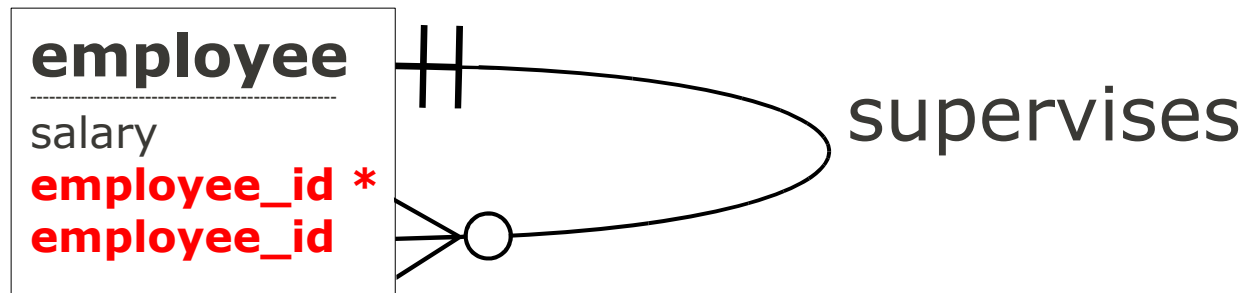
# Step 3

“For each entity that is at the ‘many’ end of one or more relationships, include the primary key of each parent entity in the table as foreign keys.”

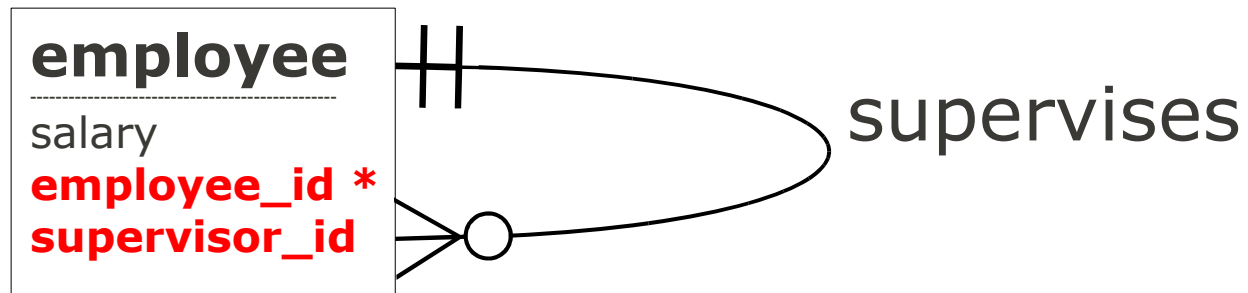
# Recursive Relationships



# Recursive Relationships



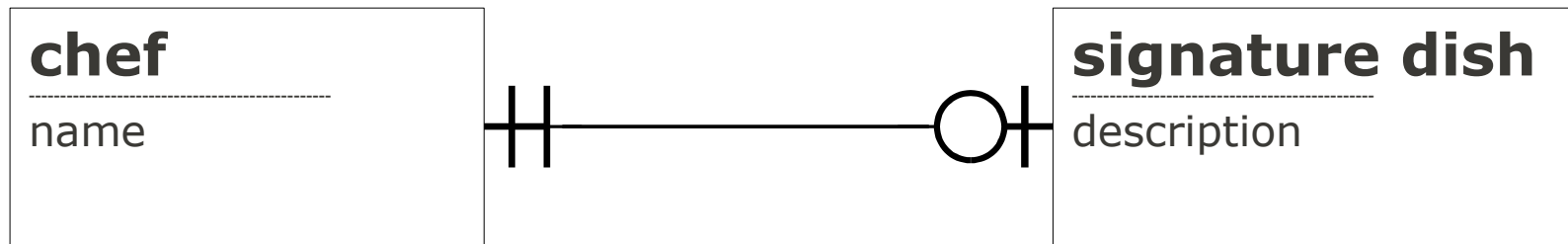
# Recursive Relationships



## employee

name	type	null	key
employee_id	int	NO	PRI
supervisor_id	int		
salary	decimal		

# 1-1 Relationships



# Option 1: Merge



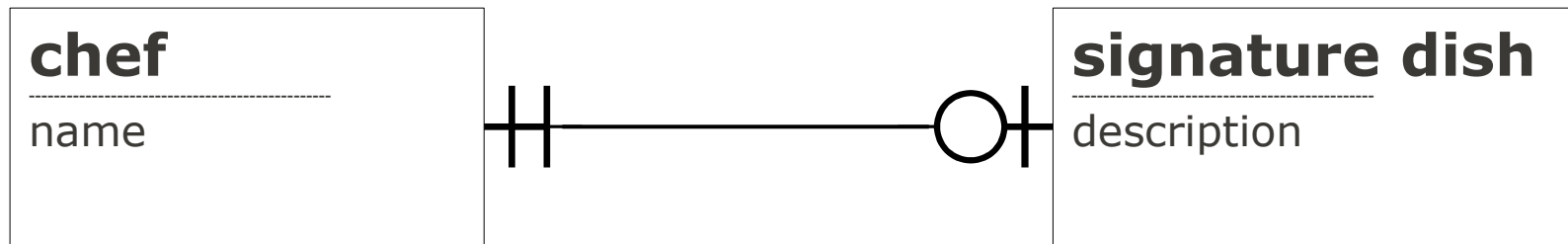


# Option 1: Merge

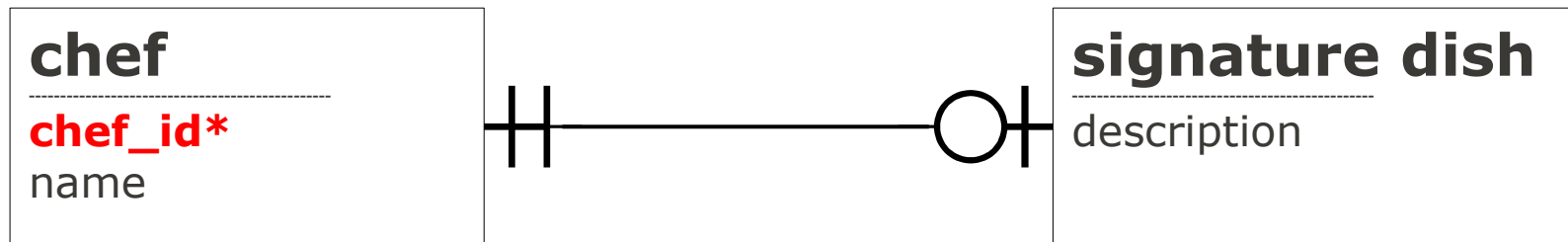
## **chef**

name	type	null	key
chef_id	int	NO	PRI
name	varchar(100)		
signature_dish _description	varchar(1000)		

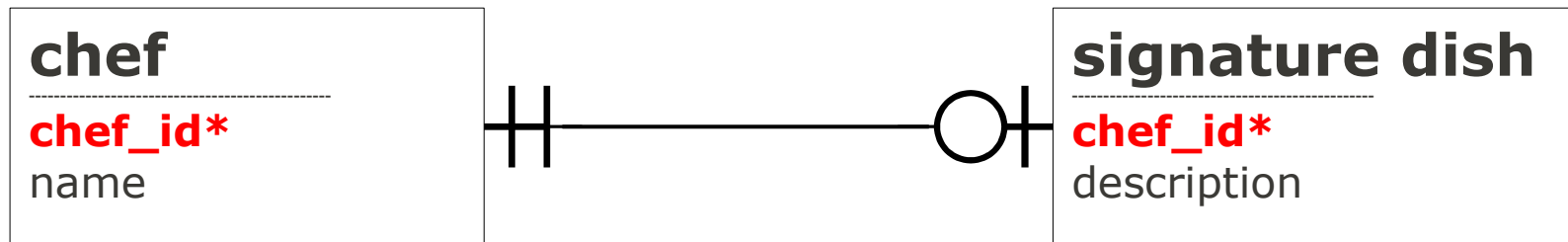
# Option 2: FK as PK



# Option 2: FK as PK



# Option 2: FK as PK



# Option 2: FK as PK

## signature dish

name	type	null	key
chef_id	int	NO	PRI
description	varchar(1000)		

Questions?

# Implementation

- Create a database/schema
- Define domains
- Create the tables

# Domains / Data Types

011010010111100000111111000001101000110111001010110010010000  
010000011101101010010001100100111100100000010000001011100110  
00011100000000110100001111010111001111100001110100110011010101  
0011111101101001110110010110000110111101000001111011101111001  
100100000101100111001100001100001111101101100000110000110011011  
00111011101110110011101000011000010011111001100000110010011011  
00001100100000101100111111101101100000100101011101010000110101  
110110101100100100000011000001001110010000110111100110110000011  
0010000000000000001011011001101111100000010110111110000110100100  
1110011001011011001010001111111110010100000001111101101000010  
0011011101101000000101111001110110010011111111111000001111111  
00001001100111111101100110011010101101111111101111001010101  
1010111111100111110111101010100011111111110111001111000010110  
10010001101010001110100010111001110110011110101010000110010100  
1011110010001000000000000100101100110100001100000101111100100010  
001100010000000000000000101000001111101101101000110100001110011100  
01100110100111100000110000001000011111011111011010111100100011  
000010010000001001001001110111111001110000100000000001000000010  
0011101110110110000100001111010001111101100100000101001000010100



# Domains / Data Types

011010010111100000111111000001101000110111001010110010010000  
010000011101101010010001100100111100100000010000001011100110  
00011100000000110100001111010111001111100001110100110011010101  
0011111101101001110110010110000110111101000001111011101111001  
100100000101100111001100001100001111101101100000110000110011011  
00111011101110110011101000011000010011111001100000110010011011  
00001100100000101100111111101101100000100101011101010000110101  
110110101100100100000110000100111001000011011110011011000011  
0010000000000000001011011001101111100000010110111110000110100100  
1110011001011011001010001111111110010100000001111101101000010  
0011011101101000000101111001110110010011111111111000001111111  
00001001100111111101100110011010101101111111101111001010101  
1010111111100111110111101010100011111111110111001111000010110  
10010001101010001110100010111001110110011110101010000110010100  
1011110010000100000000000100101100110100001100000101111100100010  
001100010000000000000000101000001111101101101000110100001110011100  
01100110100111100000110000001000011111011111011010111100100011  
000010010000001001001001110111111001110000100000000001000000010  
001110111011011000010000111101001111101100100000101001000010100

# Domains / Data Types

11011010110010010000011000010011100100011011110011011000011  
0010000000000000010110110011011111000001011011111000110100100

“Harold\n”?

1300457425722198016?

130045742.5722198016?

0.5237304801548 \* 10<sup>-36</sup>?

# String (Text)

CHAR (n)

BINARY (n)

VARCHAR (n)

VARBINARY (n)

TEXT

BLOB

ENUM

# Numeric

INTEGER                      NUMERIC (n, m)

UNSIGNED                    DECIMAL (n, m)

SMALLINT

BIGINT

BOOLEAN

REAL

DOUBLE PRECISION

FLOAT (m, d)

# Date and Time

DATE

TIME

DATETIME

TIMESTAMP

# Extensions

**Spatial:**

POINT

POLYGON

LINestring

GEOMETRY

# Type Casting

Explicit casting:

```
CAST (weight AS INTEGER)
```

Implicit casting:

```
WHERE weight > "25"
```

Beware:

```
3.2 < 25
```

```
"3.2" > "25"
```

# Create a Database

```
create database «name»;
```

For instance:

```
create database university;
```



Query OK, 1 row affected (0.00 sec)

# Delete a Database

```
drop database <<name>>;
```

For instance:

```
drop database university;
```

Query OK, 0 rows affected (0.01 sec)

# Create a Table

```
create table «name»;
```

For instance:

```
use university;  
create table student;
```

ERROR 1113 (42000): A table must have at least 1 column

# Create a Table

```
create table <name>  
  (<column> <vartype>);
```

For instance:

```
create table student  
  (student_id int);
```

Query OK, 0 rows affected (0.02 sec)

# Create a Table

```
create table «name» (  
  «column1» «vartype1»,  
  «column2» «vartype2»  
);
```

For instance:

```
create table student (  
  student_id int,  
  last_name varchar(100)  
);
```



ERROR 1050 (42S01): Table 'student' already exists

# Delete a Table

```
drop table «name»;
```

For instance:

```
drop table student;
```

Query OK, 0 rows affected (0.00 sec)

# Create a Table

```
create table «name»> (  
  «column1» «vartype1»,  
  «column2» «vartype2»  
);
```

For instance:

```
create table student (  
  student_id int,  
  last_name varchar(100)  
);
```

Query OK, 0 rows affected (0.02 sec)

# Required Fields

```
create table student (  
  student_id int not null,  
  last_name varchar(100)  
);
```

# Primary Keys

```
create table student (  
  student_id int not null,  
  last_name varchar(100),  
  primary key (student_id)  
);
```

```
mysql> describe student;
```

Field	Type	Null	Key	Default	Extra
student_id	int(11)	NO	PRI	NULL	
last_name	varchar(100)	YES		NULL	

```
2 rows in set (0.00 sec)
```



# Foreign Keys

```
create table enrollment (  
  student_id int not null,  
  foreign key (student_id)  
    references  
      student (student_id),  
  instance_id int not null  
);
```

A dirty secret: MySQL 5.0 ignores FKs!

# FKs with InnoDB

```
create table enrollment (  
  student_id int not null,  
  foreign key (student_id)  
    references  
      student (student_id),  
  instance_id int not null  
) engine=InnoDB;
```

(We'll need to re-create the student table first.)

# Load Some Data!

```
insert into student values  
(1, "Kenobi");  
insert into enrollment values  
(1, 2);
```

Query OK, 1 row affected (0.00 sec)

```
mysql> select * from student;
```

student_id	last_name
1	Kenobi

1 row in set (0.00 sec)

# From a File

```
load data  
infile "/tmp/students.csv"  
into table student;
```

# Using SCP

**scp** = **secure copy** (or **ssh + cp**)  
copy files over an ssh connection

Hint: You will usually be running this in your **local** bash session (i.e. on your laptop/desktop).

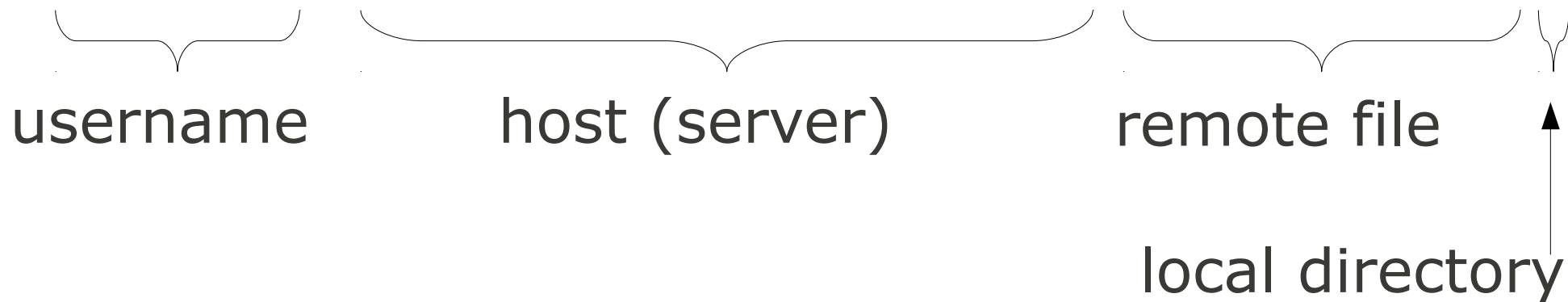
Hint: Windows users can use WinSCP instead.

# Remote to Local

```
scp user@host:/remote/file /local/dir
```

e.g.:

```
scp kenobio7@yoda.ischool.utoronto.ca:~/humans.txt .
```



# Local to Remote

```
scp /local/file user@host:/remote/dir
```

e.g.:

```
scp students.csv kenobio7@yoda.ischool.utoronto.ca:~/
```



# Permissions

```
chmod a+r /tmp/students.csv
```

# Editing Code Locally

Windows: **Notepad++**

Mac: **TextWrangler**

Linux: **gedit** (or emacs, vi)

Key feature: syntax highlighting

# Comma-Separated

```
load data
infile "/tmp/students.csv"
into table student
fields terminated by ",";
```

Query OK, 1 row affected (0.00 sec)

```
mysql> select * from student;
```

```
+-----+-----+
| student_id | last_name |
+-----+-----+
|          1 | Kenobi    |
+-----+-----+
```

1 row in set (0.00 sec)

```
mysql> load data infile "/tmp/students.csv" into table
student;
```

ERROR 1062 (23000): Duplicate entry '1' for key 1

# Deleting Data

```
delete from student;
```

# Deleting Select Data

```
delete from student  
where ... ;
```

# ON DELETE

```
create table enrollment (  
  student_id int not null,  
  foreign key (student_id)  
    references  
    student (student_id)  
    on delete cascade,  
  instance_id int not null  
) engine=InnoDB;
```

# Data

1273	Smith	Robert
1274	Ivanov	Boris
1275	Almeida	João



# Easier

**Smith**

**Robert**

**Ivanov**

**Boris**

**Almeida**

**João**

# Auto-Increment

```
create table student (  
  student_id int not null  
    auto_increment,  
  last_name varchar(100),  
  primary key (student_id)  
) engine=InnoDB;
```

# Auto-Increment

```
insert into student values  
(NULL, "Kenobi");
```

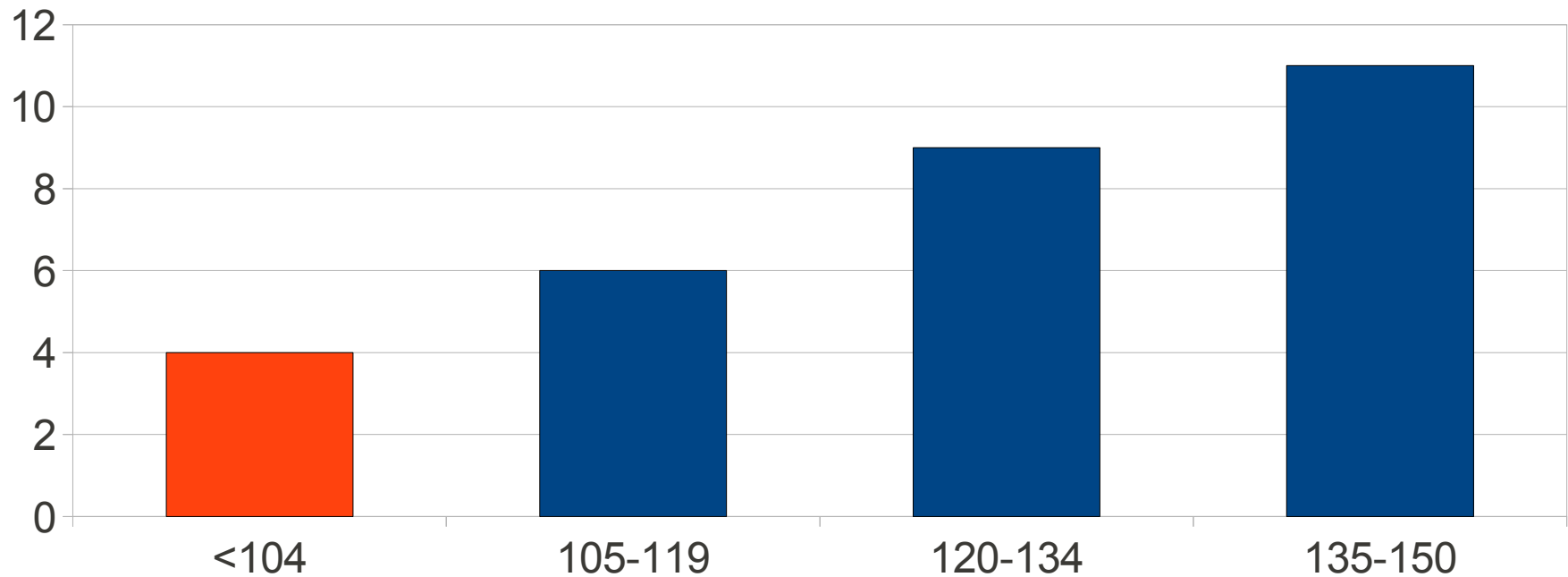
```
insert into student (last_name)  
values ("Kenobi");
```

# Auto-Increment

```
load data  
infile "/tmp/students.csv"  
into table student;
```

Questions?

# Assignment 1



See the answer sheet on the website.