Databases

SWEN-250

Persistence

First a detour…

“Persistence is the key to solving most mysteries.”

— Christopher Pike, *Black Blood*

Persistence when you are stuck on a piece of new code is hardly ever a virtue. Try redesigning the class, try an alternative coding approach, or try coming back to it later.

“When one approach isn’t working, that’s a good time to try an alternative”

— Robert Pirsig, *Zen and the Art of Motorcycle Maintenance*

Persistence

Software Persistence refers to the **life span** of a piece of data.

Variables can persist:

- For a particular block of code (e.g. loop or function)
- Until you allow them to (in java – garbage collected; in C/C++ until you **delete**)
- For the life of the program (global variables or *static* ones)
- “Forever” (as in that which is stored in a **Database**)

Database terminology

- **Field**: “smallest” item of stored data.

  For example, a character string associated with a type of information:

  Alumni’s **`FirstName`**

- **Record**: group of related fields and associated values

  For example, the collection of attributes associated to an Alumni:

FirstName	LastName	Class	UnivID
Katie	Linendoll	2005	145-254-2541

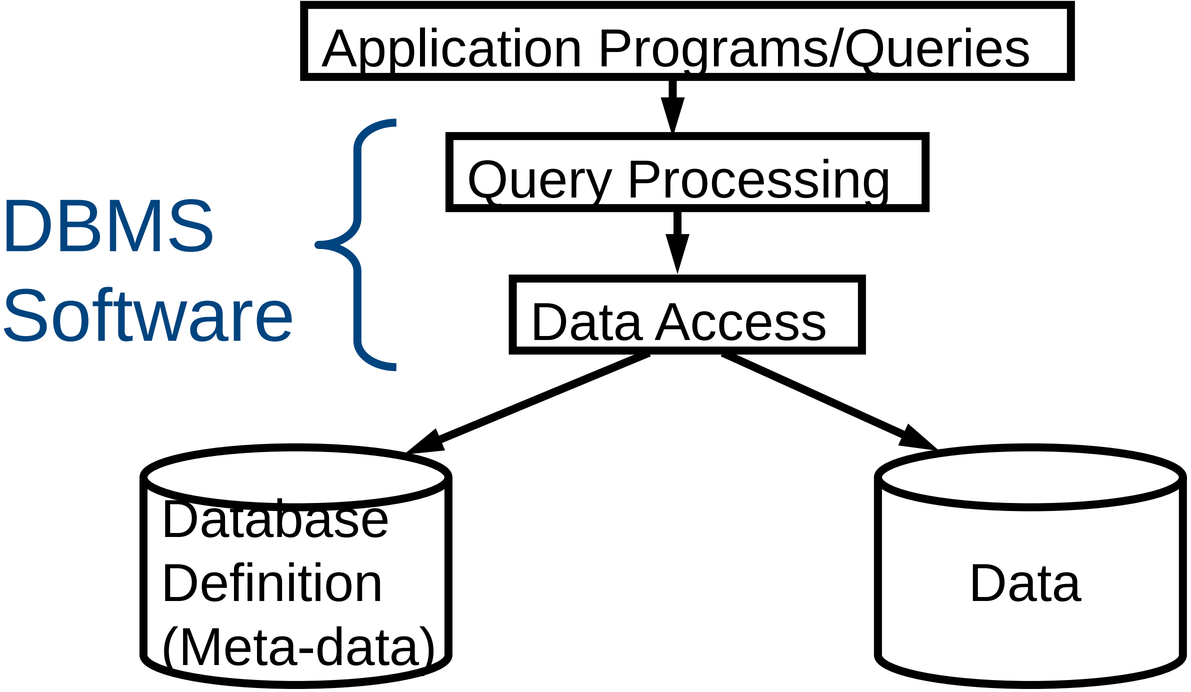
Database terminology

- **Database “file” or Table**: collection of related records that are frequently categorized for given purpose.

FirstName	LastName	Class	UnivID
Katie	Linendoll	XXXX	XXXXX4-2541
Alex	Kipman	2001	985-65-6258
Robert	Duffy	XXXX	XXXXX5-8574

Database terminology

- **Database**: a collection of related tables
- **Database Management System (DBMS)**: a system providing control over definition, access and manipulation of the information stored in the database



Database (history)

- Early database systems were ad hoc
- Hierarchical and Navigational Models
  - Affords parent-child structure but is restrictive and it is hard to maintain links.
- Relational Databases
  - Ignores how files are connected and organizes data into two-dimensional unordered-tables
  - Data can be separated into multiple tables
  - Leverages relational algebra & relational calculus
  - No need for expensive rewrites of links
- Object-oriented Databases
  - Seeks optimization by organizing data as objects and their attributes
  - Often challenged by the translation to/from objects

Database (recent history)

- NoSQL Databases incorporate any number of approaches such as:
  - Replication across nodes in a cluster
  - Fast key-value stores
  - Schema-free document storage
  - Extensible columns
  - XML oriented attributes

Example – Baseball Database

```
$ sqlite3 baseball.db
SQLite version 3.7.9
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
create table Players (id integer primary key, name text,  team_id integer, position text) ;

create table Teams (id integer primary key, name text, city text) ;

.schema
```

Players

id	name	team_id	position

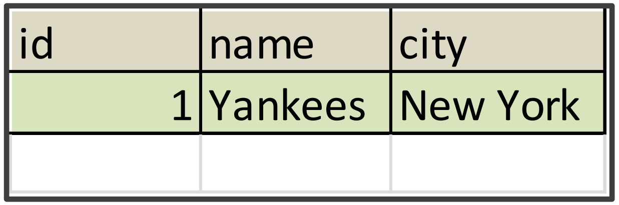
Teams

id	name	city

Example – Adding Rows

```
-- This is a comment
-- Insert First team info
insert into Teams (name, city) values ("Yankees", "New York");
```

Teams

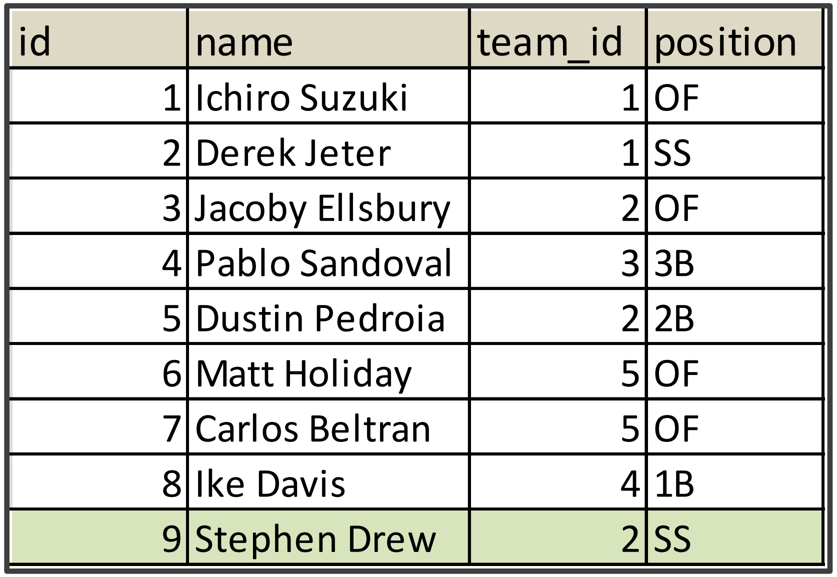


id	name	city
1	Yankees	New York

* The primary key (id) is automatically incremented on inserts

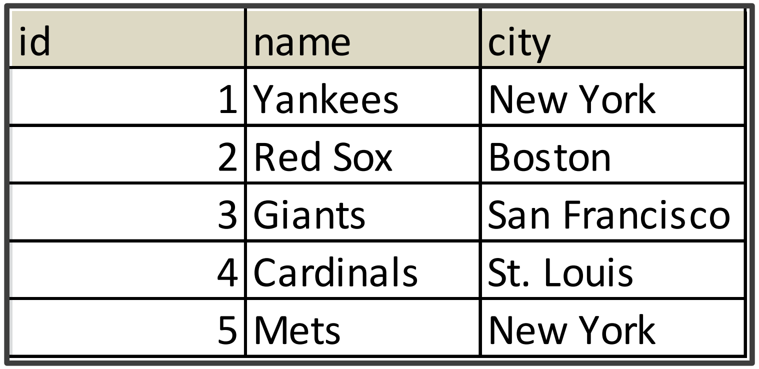
Example – Adding Rows

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- This is a comment
-- Insert Stephen Drew – the primary key is auto-incremented to ensure uniqueness.
insert into Players (name, team_id, position) values ("Stephen Drew", 2, "SS");
```

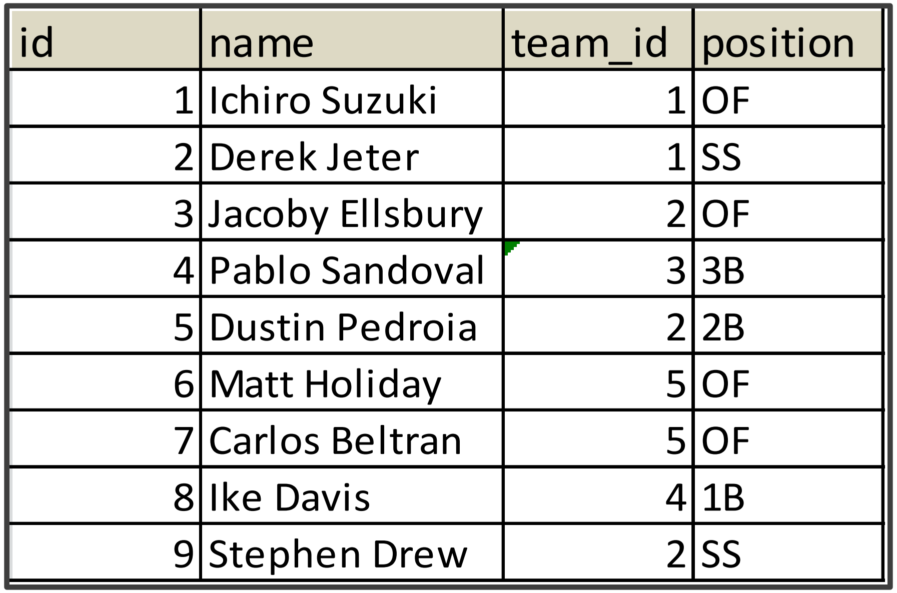
Example – Retrieving Data

Operator: **Select**

```
-- List the Players table
select * from Players ;
```

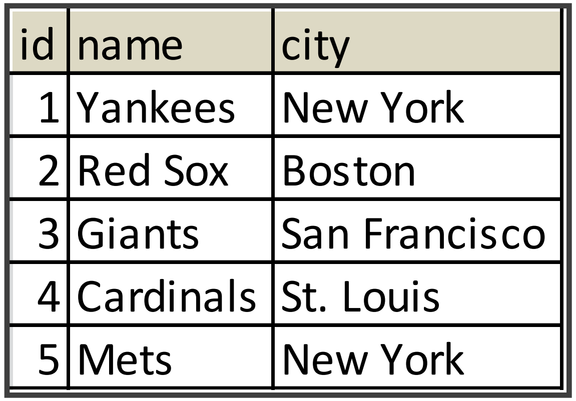
```
-- List the Teams table
select * from Teams ;
```

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



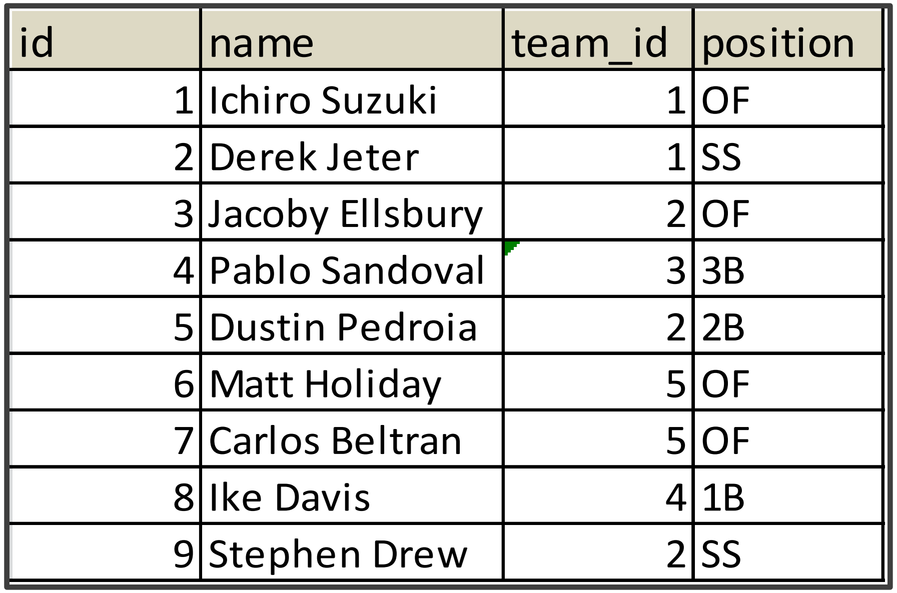
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

select * means ALL THE COLUMNS

Example – Retrieving Data

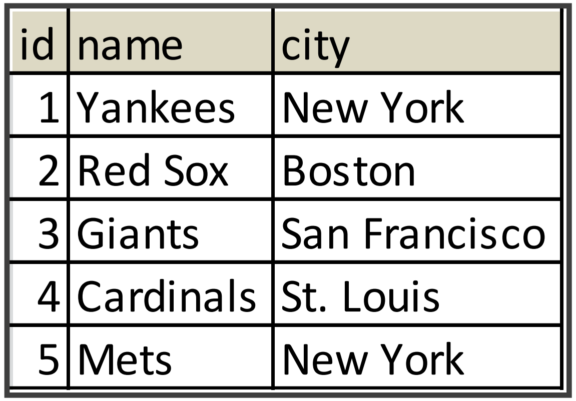
Operator: **Select** with additional criteria

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



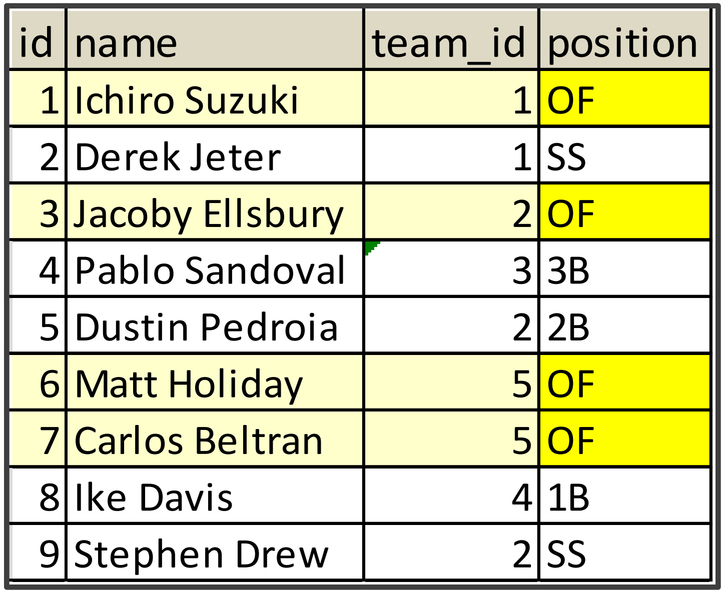
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the Players table plus criteria
select * from Players where position = "OF" ;
```

Operator: Select

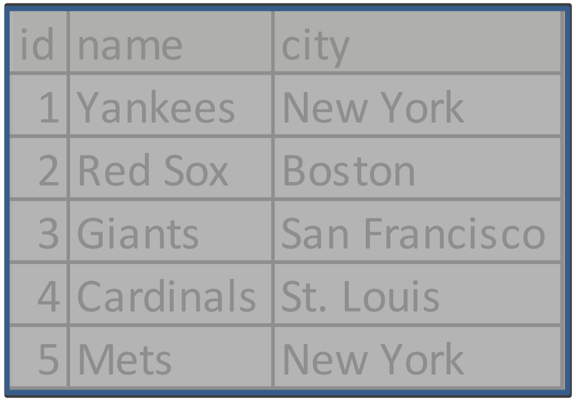
Retrieve rows based on criteria

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



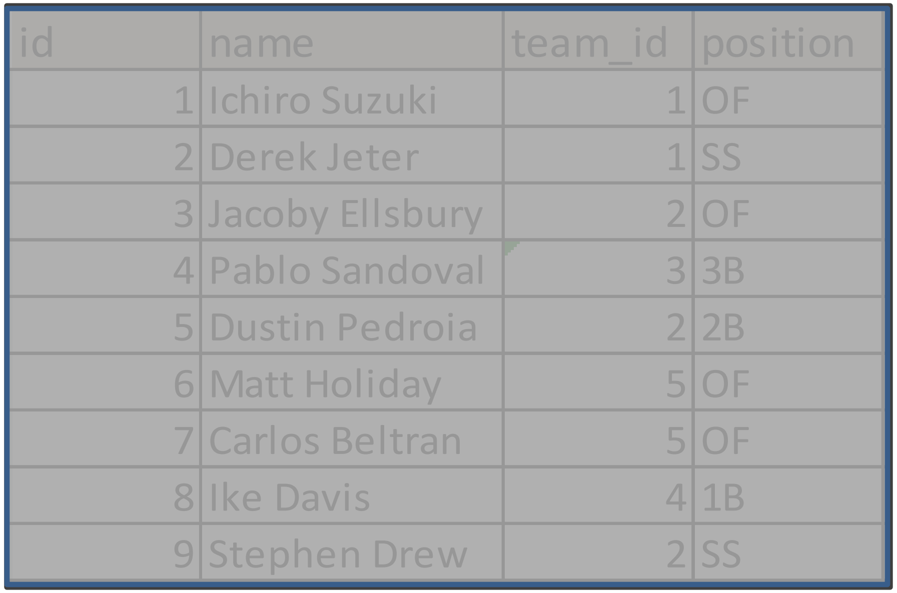
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the Players table plus criteria
select * from Players where position = "OF" ;
```

Operator: Select

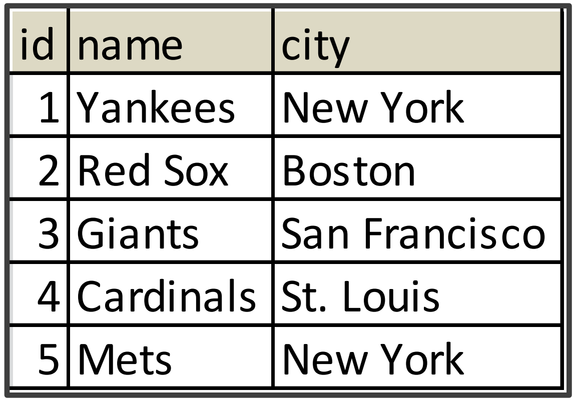
Select some rows based on criteria

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



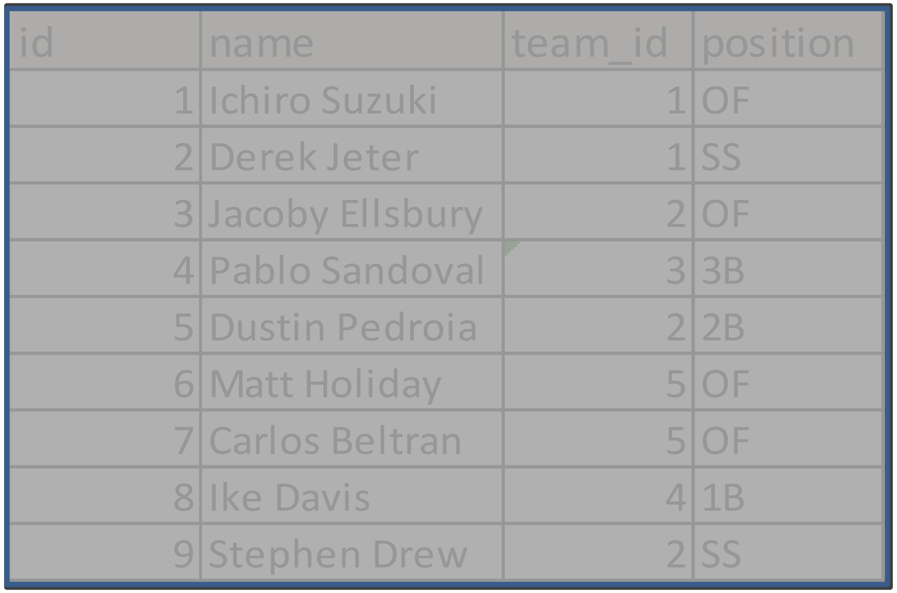
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the Teams table plus criteria
select * from Teams where city = "New York";
```

Operator: Select

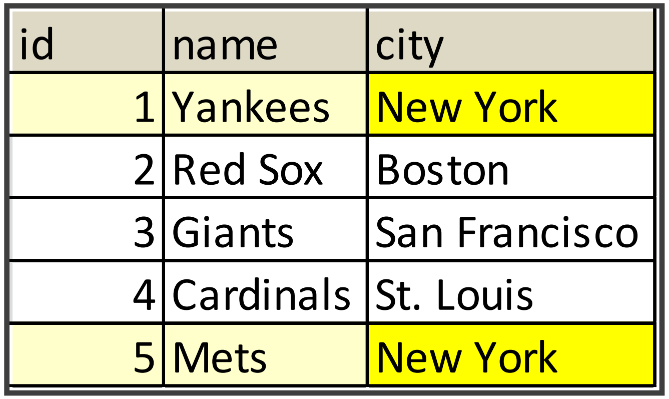
Select some rows based on criteria

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



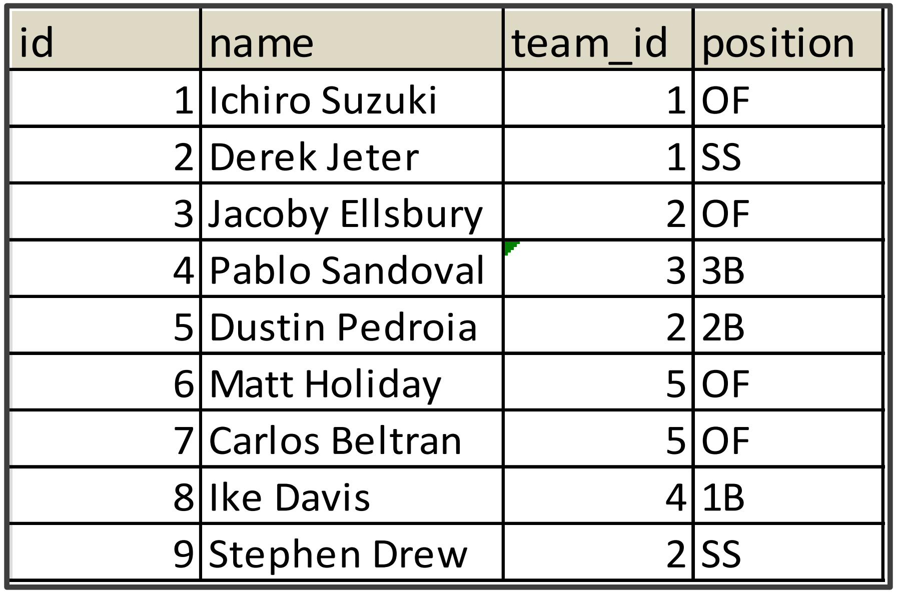
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the Teams table
select * from Teams where city = "New York";
```

Operator: Select

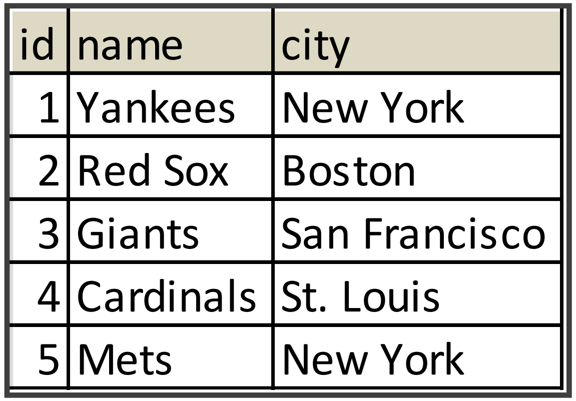
Select some rows based on criteria

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the name and position of each player
select name, position from Players ;

-- List the name and city of each team
select name, city from Teams ;
```

Operator: Select

Choose only some of the columns

Players

id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams

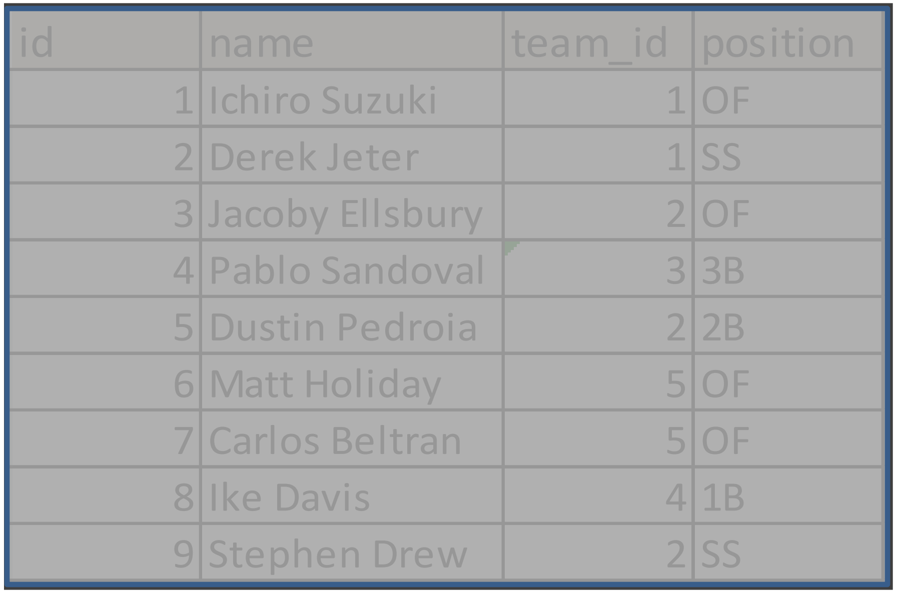
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the name and position of each player
select name, position from Players ;

-- List the name and city of each team
select name, city from Teams ;
```

Operator: Select

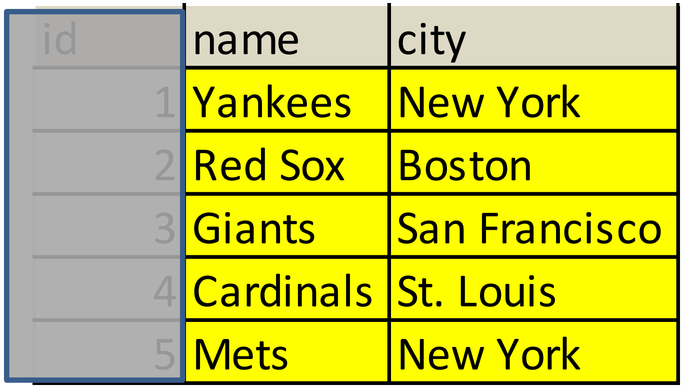
Choose only some of the columns

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams

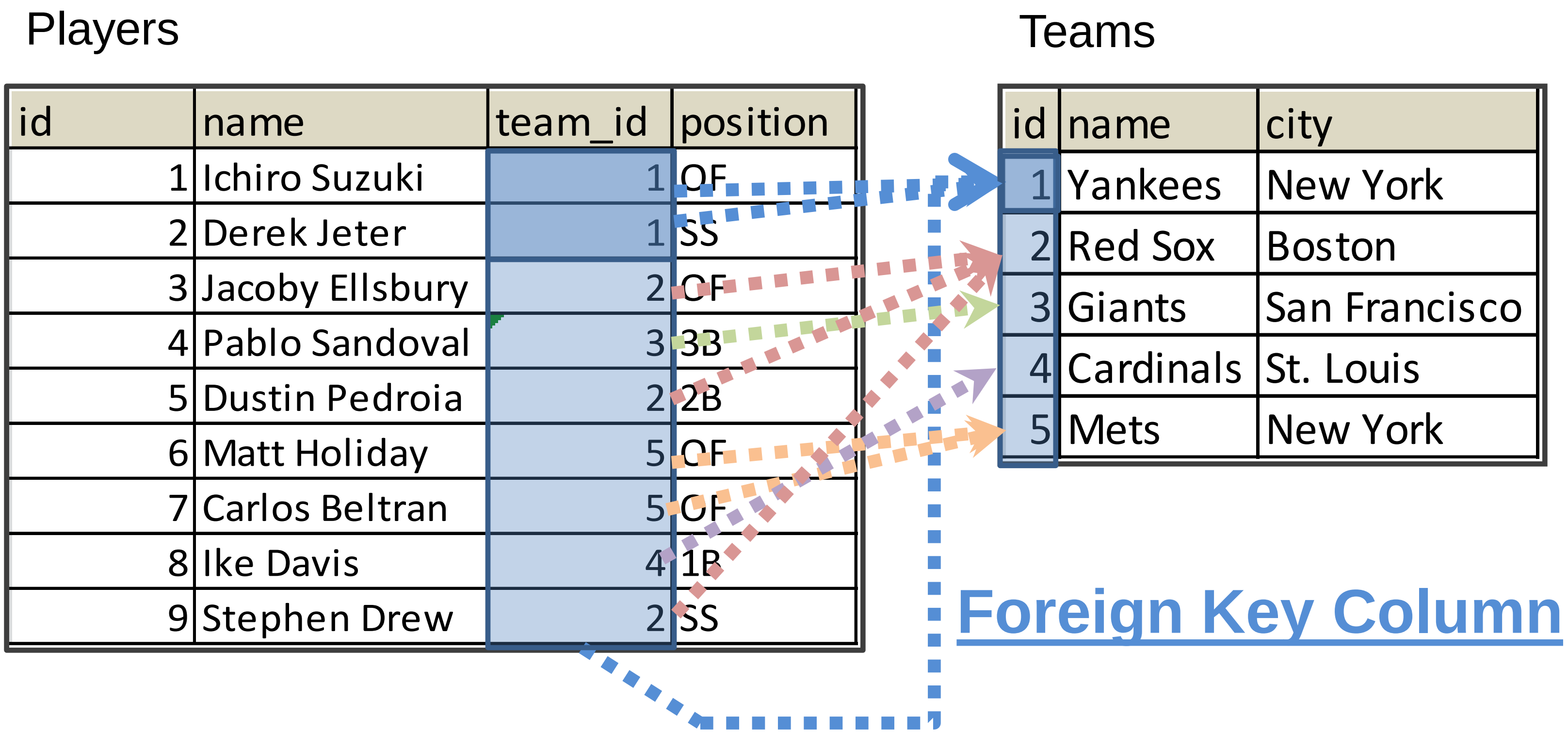


id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the name and position of each player
select name, position from Players ;

-- List the name and city of each team
select name, city from Teams ;
```

Operator: Join

Match rows from one table against another



Players

id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams

id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

Foreign Key Column

It will complete this process for ALL team_id’s

~~-- List all combinations of Players & Teams (not very useful)~~
~~select * from Players, Teams ;~~

-- List combinations where Player's team_id = Teams' id (join & select)
select * from Players, Teams **where Players.team_id = Teams.id ;**

Exercises

Join / Select / Project

Players

id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams

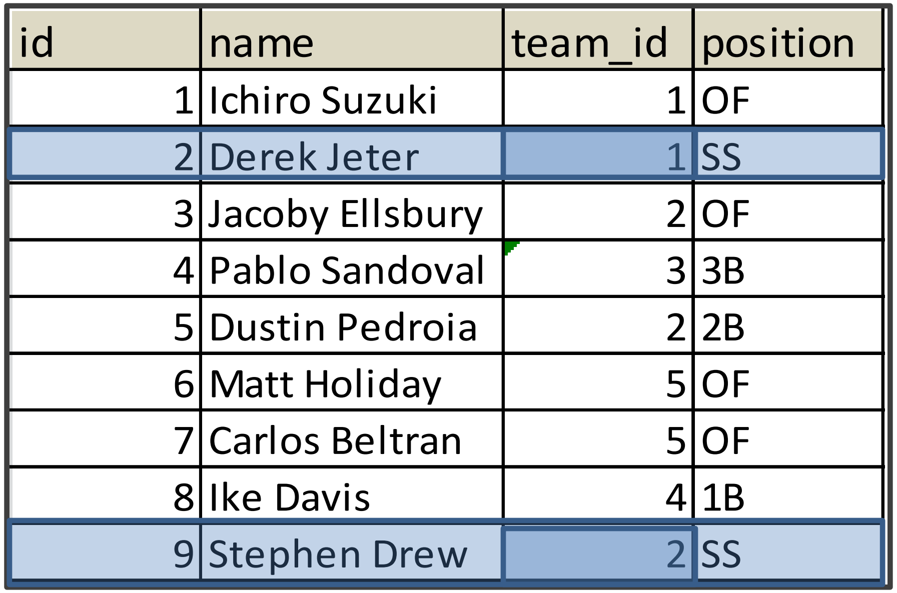
id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

```
-- List the names of the out fielders playing in New York
select Players.name from Players, Teams where Players.team_id = Teams.id and
   Players.position = "OF " and Teams.city = "New York" ;
```

Exercises

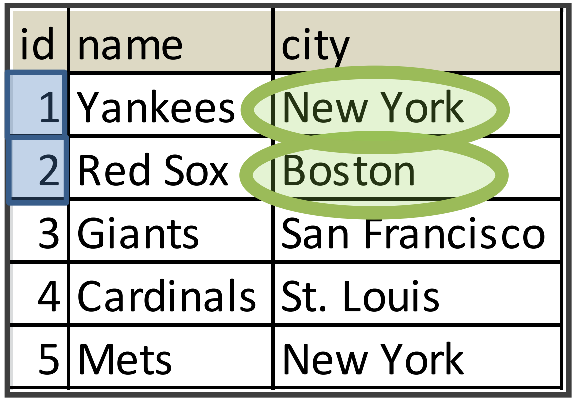
Join / Select / Project

Players



id	name	team_id	position
1	Ichiro Suzuki	1	OF
2	Derek Jeter	1	SS
3	Jacoby Ellsbury	2	OF
4	Pablo Sandoval	3	3B
5	Dustin Pedroia	2	2B
6	Matt Holiday	5	OF
7	Carlos Beltran	5	OF
8	Ike Davis	4	1B
9	Stephen Drew	2	SS

Teams



id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York

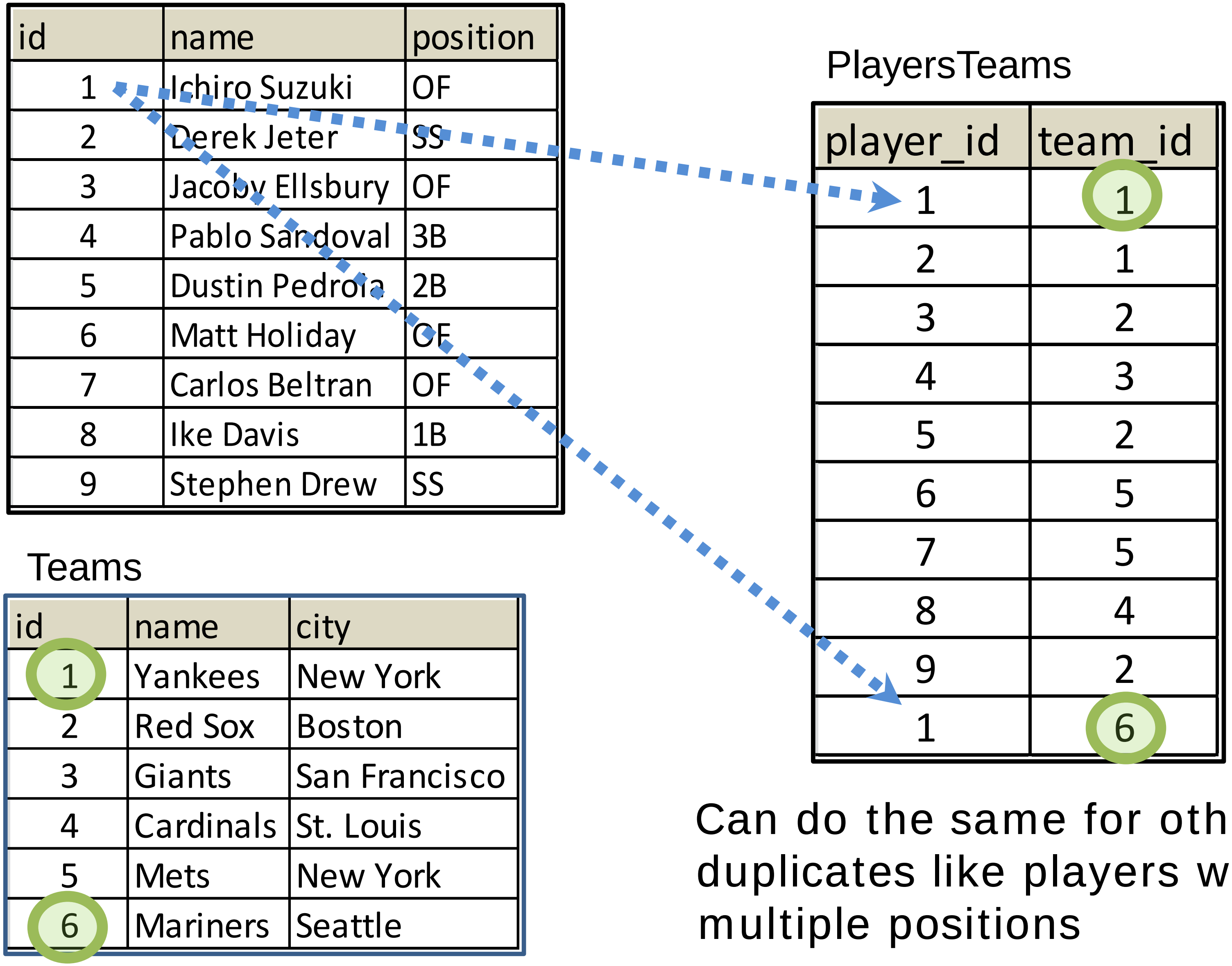
```
-- List the cities of all the short stops
select Teams.city from Players, Teams where Players.team_id;
-- Only shows city for each and all players

-- we need to "filter" the SS position
select Teams.city from Players, Teams where Players.team_id = Teams.id and
   Players.position = "SS" ;
```

What About Players w/ Multiple Teams?

- Example: See Ichiro Suzuki – 2012 Season
  - Started season with Seattle - Mariners
  - Ended season with New York - Yankees
- Approach #1:
  - Duplicate records – what we have
  - That is, two or more records for a player – one per position.
  - Issue of redundancy (DRY violation).
- Approach #2:
  - Table for player
  - Table for team
  - Table *linking* players to teams

Linking Table

(a way to capture Relationship)



Players

id	name	position
1	Ichiro Suzuki	OF
2	Derek Jeter	SS
3	Jacoby Ellsbury	OF
4	Pablo Sandoval	3B
5	Dustin Pedroia	2B
6	Matt Holiday	OF
7	Carlos Beltran	OF
8	Ike Davis	1B
9	Stephen Drew	SS

PlayersTeams

player_id	team_id
1	1
2	1
3	2
4	3
5	2
6	5
7	5
8	4
9	2
1	6

Teams

id	name	city
1	Yankees	New York
2	Red Sox	Boston
3	Giants	San Francisco
4	Cardinals	St. Louis
5	Mets	New York
6	Mariners	Seattle

Can do the same for other duplicates like players with multiple positions

SQLManager for Firefox

- Installation:
  - From Firefox visit **firefox/addon/sqlite-manager/**
  - Select the **Add to Firefox** button.
  - Follow the installation instructions

    [You may have to restart Firefox]
- Open the manager – it's in the Firefox **Menu Bar** under **Tools**.
- You can either open an existing database or create a new one.
  - As you manipulate the database, you'll see the SQL statements that are executed.
  - Joins require you to type the SQL in the Execute SQL tab.