



Database Schema Design

Organizing Data

Databases



Why you care

SQL is the gateway to the world of Databases

Reading & Writing Raw SQL is a skill that will set you apart from other Devs

Super Important for implementing & debugging advanced Database systems

Database Schema Design

Good Database Design should:

- Capture state, not behavior
- Design relationships between tables

State and Behavior

```
class OrangeTree
  def initialize
    @age = 5
    @height = 5
    @oranges = produce_oranges
  end

  def produce_oranges
    Array.new(3) { Orange.new }
  end
end
```

```
class Orange
  def initialize
    @diameter = rand(2..4)
  end
end
```

Temporary State

- We create objects to hold data
- Data is lost when our program has finished executing



Temporary State Problems

- What if we want to use the current state later?

Temporary State Problems

- What if we want to use the current state later?

Mega Man vs. Mega Man II

Phone number from a super hot crush

50 million client purchase records

Persistent State

To convert temp state to persistent state
we need to save the data. But where?

CSV

Pen && Paper

😊Database😊

Databases && Ruby

- Like object-oriented programming, databases model the data in real world systems
- How do objects in databases relate to Ruby?

Databases and Ruby

Modeling State	
<i>Ruby</i>	<i>Database</i>
<i>Classes</i>	
<i>Instances of classes</i>	
<i>Instance variables</i>	

Databases and Ruby

Modeling State	
<i>Ruby</i>	<i>Database</i>
<i>Classes</i>	<i>Tables</i>
<i>Instances of classes</i>	
<i>Instance variables</i>	

Databases and Ruby

Modeling State	
<i>Ruby</i>	<i>Database</i>
<i>Classes</i>	<i>Tables</i>
<i>Instances of classes</i>	<i>Rows</i>
<i>Instance variables</i>	

Databases and Ruby

Modeling State	
<i>Ruby</i>	<i>Database</i>
<i>Classes</i>	<i>Tables</i>
<i>Instances of classes</i>	<i>Rows</i>
<i>Instance variables</i>	<i>Fields</i>

AKA: Schema

Schema Design

orange_trees
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design

orange_trees
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>diameter</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>name</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>diameter</i>
<i>created_at</i>
<i>updated_at</i>

Schema Design WorkFlow

- Identify Tables
 - Identify Relationships
 - Add Foreign Keys
 - Identify Attributes & Data Types
-
- Accomplish the above steps by modeling real world observations within your schema.

Relationships

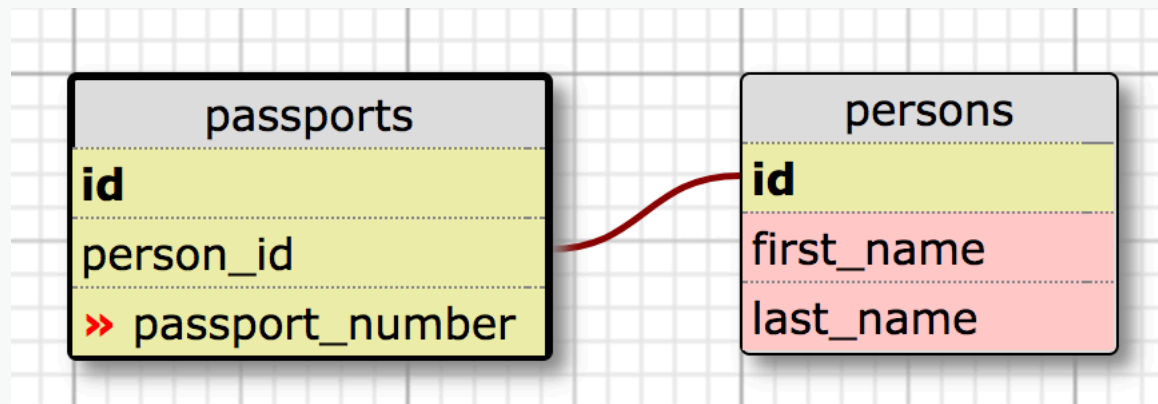
- Schema is more than just static tables
- SQL is a relational DB

Types of Relationships:

- One to One
- One to Many
- Many to Many

One to One

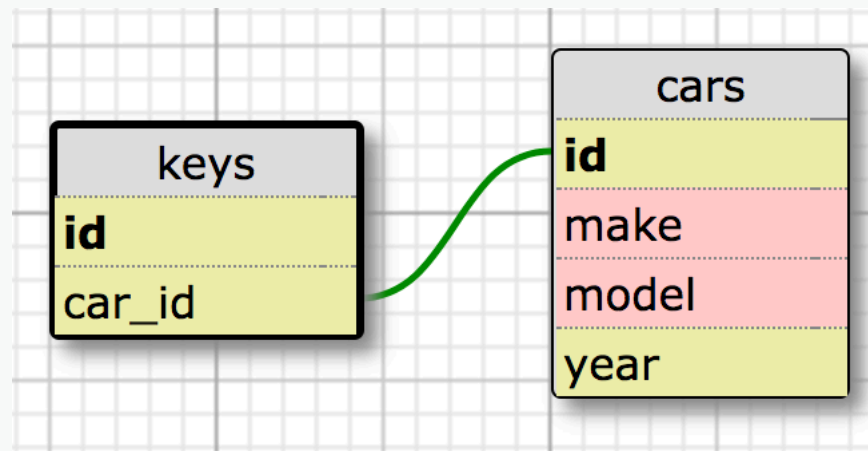
- Least common type of relationship.
- Must ensure that only one of each record will be associated with a corresponding record.



One to Many

Most common type of relationship

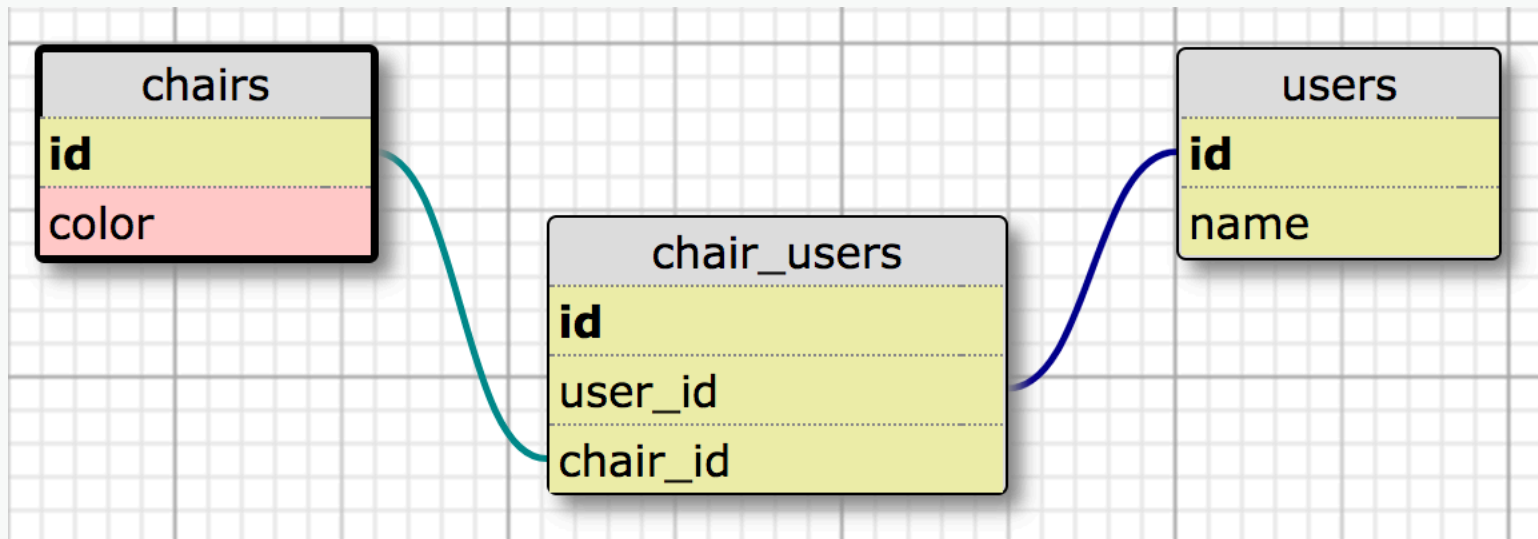
- A **Car** has many **Keys**
- A **Key** belongs to a **Car**



Many to Many

Uses a Join Table

- A **Chair** has many **Users** through **chair_users**
- A **User** has many **Chairs** through **chair_users**
- A **chair_user** belongs_to **user** && **chair**



Schema Design

- What is the relationship between orange trees and oranges?

Schema Design

- What is the relationship between orange trees and oranges?

An **orange tree** has many **oranges**.

Schema Design

- What is the relationship between orange trees and oranges?

An **orange tree** has many **oranges**.

An **orange** belongs to an **orange tree**.

DB Relationships: Talk it out.

- Relationships are two way streets
- Should sound like a natural sentence

An **orange tree** has many **oranges**.

An **orange** belongs to an **orange tree**.

Schema Design

- How to link tables?

Schema Design

- How to link tables?

Primary key: unique identifier table field

Every table has autogenerated Primary Keys, unless specifically disabled

Foreign key: another table's unique identifier

Only tables which belong to another table contain a Foreign Key

Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>diameter</i>
<i>created_at</i>
<i>updated_at</i>

On which table does the foreign key belong?

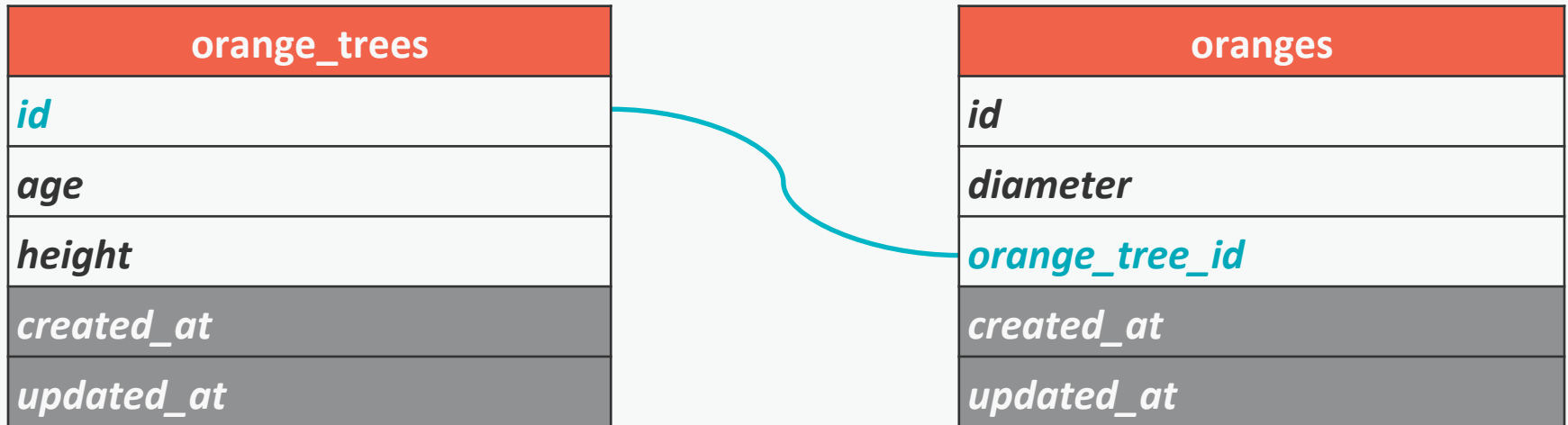
Schema Design

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>created_at</i>
<i>updated_at</i>

oranges
<i>id</i>
<i>diameter</i>
<i>orange_tree_id</i>
<i>created_at</i>
<i>updated_at</i>

The foreign key belongs on the table of the objects that are owned

Schema Design



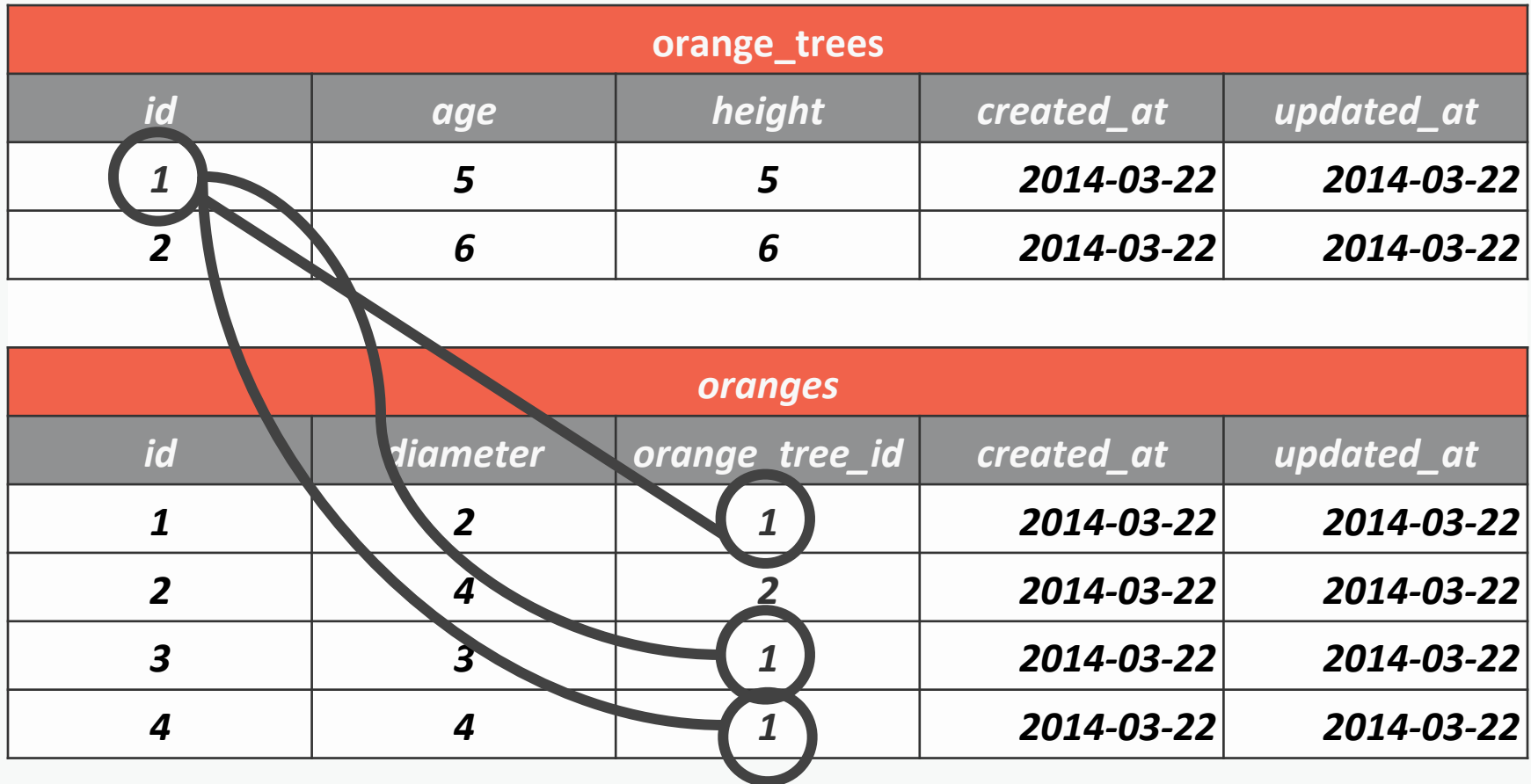
The foreign key belongs on the table of the objects that are owned

Schema Design

orange_trees				
<i>id</i>	<i>age</i>	<i>height</i>	<i>created_at</i>	<i>updated_at</i>
1	5	5	2014-03-22	2014-03-22
2	6	6	2014-03-22	2014-03-22

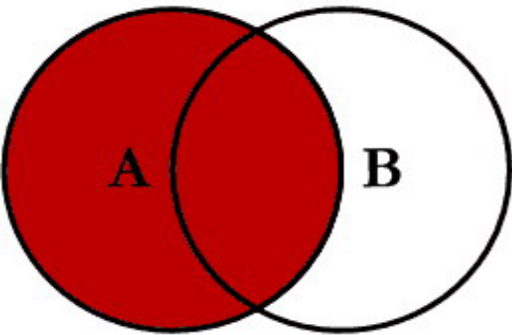
oranges				
<i>id</i>	<i>diameter</i>	<i>orange_tree_id</i>	<i>created_at</i>	<i>updated_at</i>
1	2	1	2014-03-22	2014-03-22
2	4	2	2014-03-22	2014-03-22
3	3	1	2014-03-22	2014-03-22
4	4	1	2014-03-22	2014-03-22

Schema Design

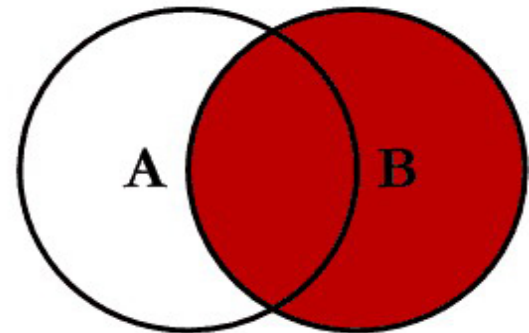


SQL JOINS

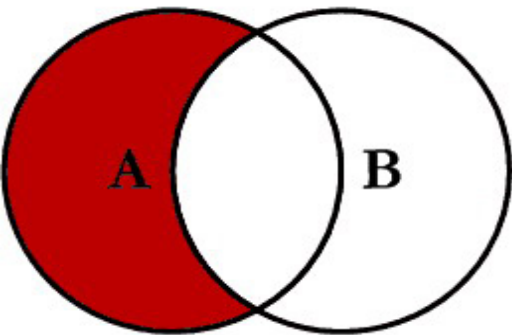
More Detail @ [the Source](#)



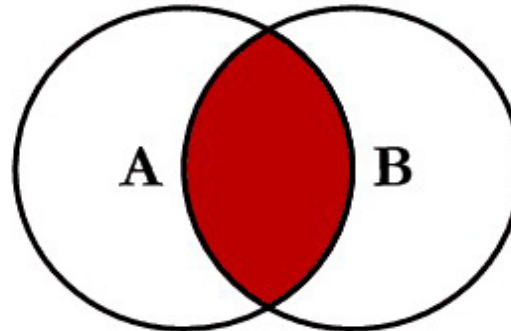
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



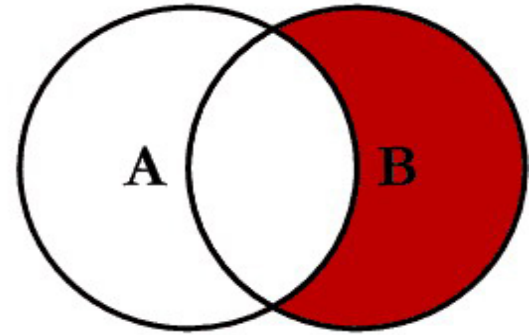
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



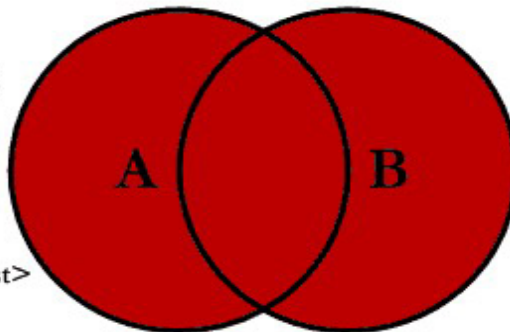
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



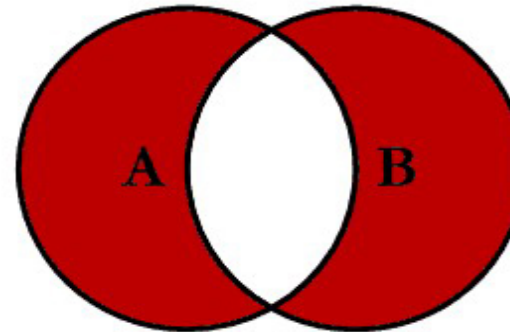
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

SQL Statements

orange_trees
<i>id</i>
<i>age</i>
<i>height</i>
<i>name</i>
<i>created_at</i>
<i>updated_at</i>

```
CREATE TABLE orange_trees
(
  id integer PRIMARY KEY AUTOINCREMENT,
  age int NOT NULL,
  height int NOT NULL,
  name varchar(255),
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

SQL Statements

oranges
<i>id</i>
<i>diameter</i>
<i>created_at</i>
<i>updated_at</i>

```
CREATE TABLE oranges
(id integer PRIMARY KEY,
diameter integer NOT NULL,
created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

SQL Statements

```
INSERT INTO orange_trees
(age, height, name)
VALUES
(1,10,'Brick'),
(1,15,'Bill'),
(2,22,'Jordan');

INSERT INTO oranges(diameter)
VALUES(3);
```


SQL Statements

```
SELECT name FROM orange_trees  
WHERE age = 1 AND height < 20;  
  
SELECT * FROM oranges  
WHERE diameter = 3;
```

SQL Links:

<http://www.cheat-sheets.org/sites/sql.su/>

<https://dev.mysql.com/doc/refman/5.0/en/group-by-functions.html>

<http://lmgtfy.com/?q=SQL+group+by+aggregate+functions>

SQL CHEET SHEET

SQL SELECT STATEMENTS

```
SELECT * FROM t
SELECT c1,c2 FROM t
SELECT c1,c2 FROM t
WHERE conditions
SELECT c1,c2 FROM t
WHERE conditions
ORDER BY c1 ASC,c2 DESC
SELECT DISTICT c1,c2
FROM t
SELECT c1, aggregate(c2 * c3)
FROM t
GROUP BY c1
SELECT c1, aggregate(c2 * c3)
FROM t
GROUP BY c1
HAVING c1 > v1
```

SQL UPDATE DATABASE

```
INSERT INTO t (c1,c2...)
VALUES (v1,v2...)
INSERT INTO t1 (c1,c2...)
SELECT c1,c2... FROM t2
WHERE conditions
UPDATE t
SET c1 = v1, c2 = v2,...
WHERE conditions
DELETE FROM t
WHERE conditions
TRUNCATE TABLE t
```

SQL OPERATORS

```
SELECT * FROM t
WHERE c1 [NOT] BETWEEN v1
AND v2
SELECT * FROM t
WHERE c1 [NOT ] IN (v1,v2,...)
SELECT* FROM t
WHERE c1 > v1 AND c1 < v2
SELECT * FROM t
WHERE c1 < v1 OR c1 > v2
SELECT * FROM t
WHERE c1 = v1
SELECT * FROM t
WHERE c1 <> v1
```

SQL TABLE STATEMENTS

```
CREATE TABLE t(
c1 dt1(l1),
c2 dt2(l2),
...
)
DROP TABLE t
ALTER TABLE t
ADD COLUMN c dt(l)
ALTER TABLE t
DROP COLUMN c
```

SQL VIEW STATEMENTS

```
CREATE UNIQUE
INDEX idx ON t(c1,c2..)
DROP INDEX t.idx
```

SQL JOIN STATEMENTS

```
SELECT * FROM t1
INNER JOIN t2 ON conditions
SELECT * FROM t
WHERE c1 [NOT ] IN (v1,v2,...)
SELECT * FROM t1
INNER JOIN t2 ON conditions
WHERE conditions
SELECT *
FROM t1, t2
WHERE conditions
SELECT * FROM t1
LEFT JOIN t2 ON conditions
SELECT * FROM t1
RIGHT JOIN t2 ON conditions
SELECT * FROM t1
FULL OUTER JOIN t2 ON conditions
SELECT * FROM t1 AS at1
INNER JOIN t2 AS at2 ON at1.c1 =
at2.c2
```

SQL VIEW STATEMENTS

```
CREATE VIEW vw
AS
SELECT c1,c2
FROM t
ALTER VIEW vw
AS
SELECT c1,c2
FROM t
DROP VIEW vw
```

SQL && Schema

Questions