**Advanced Web Development – “Simple Relationships”**

Points: 20

Due date on blackboard

**Name of students working on lab: \_Kevin Kuo & Mary Snyder\_**

**Instructions:**

The aim of this lab is to understand many\_to\_many relationships in activerecord. There are two ways to define a relationship between tables in ActiveRecord:

1. implicit: where the join table contains no fields other than the foreign keys. This does not have a model in ActiveRecord.
2. explicit: where the join table contains more fields about the relationship itself. This needs a model in ActiveRecord.

The following examples use a simple scenario and schema where there are homes that are owned by multiple owners and vice-versa. Thus, the homes and owners tables need a many-to-many relationship between them.

You may have to refer to the book/slides to understand some things.

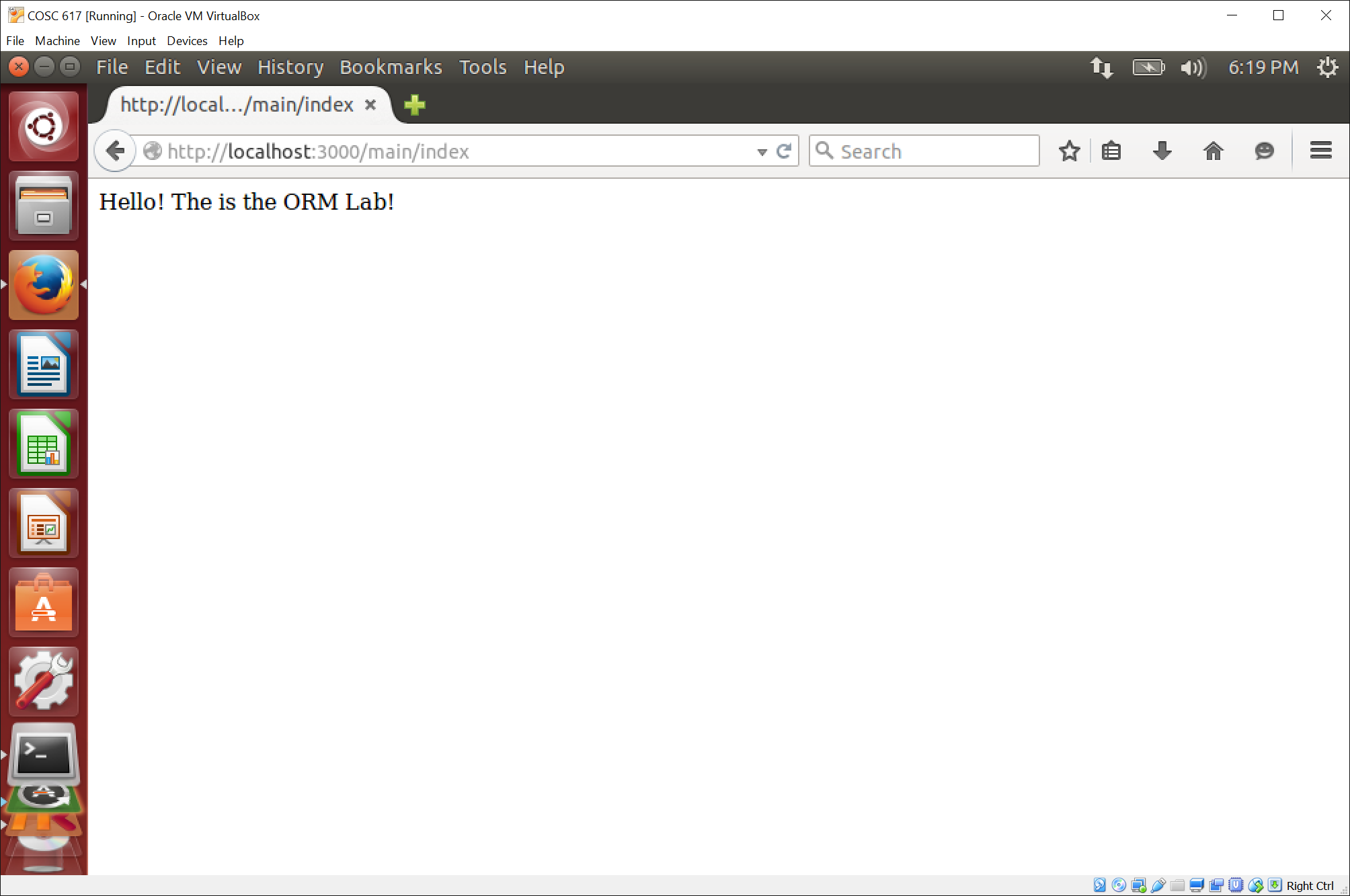
**Please answer all questions – you can be very brief.**

**Question 1.**

Clone the orm\_demo project <https://github.com/siddharthkaza/orm_homeowner_lab_42>

* 1. Run the application (call the index method) to make sure its running in the browser.

*Hint:* It’s always a good idea to run ‘bundle install’ and then ‘rake db:migrate’ and ‘rake db:seed’ first - when using an application from another location. This updates any gems needed and brings the DB to its newest version.



* 1. Start the ‘rails console’ within the application – **Type ‘rails console’ on the terminal window for the application.**

If you get the ‘cannot find readLine’ error, then you will need to install the necessary libraries for the console.

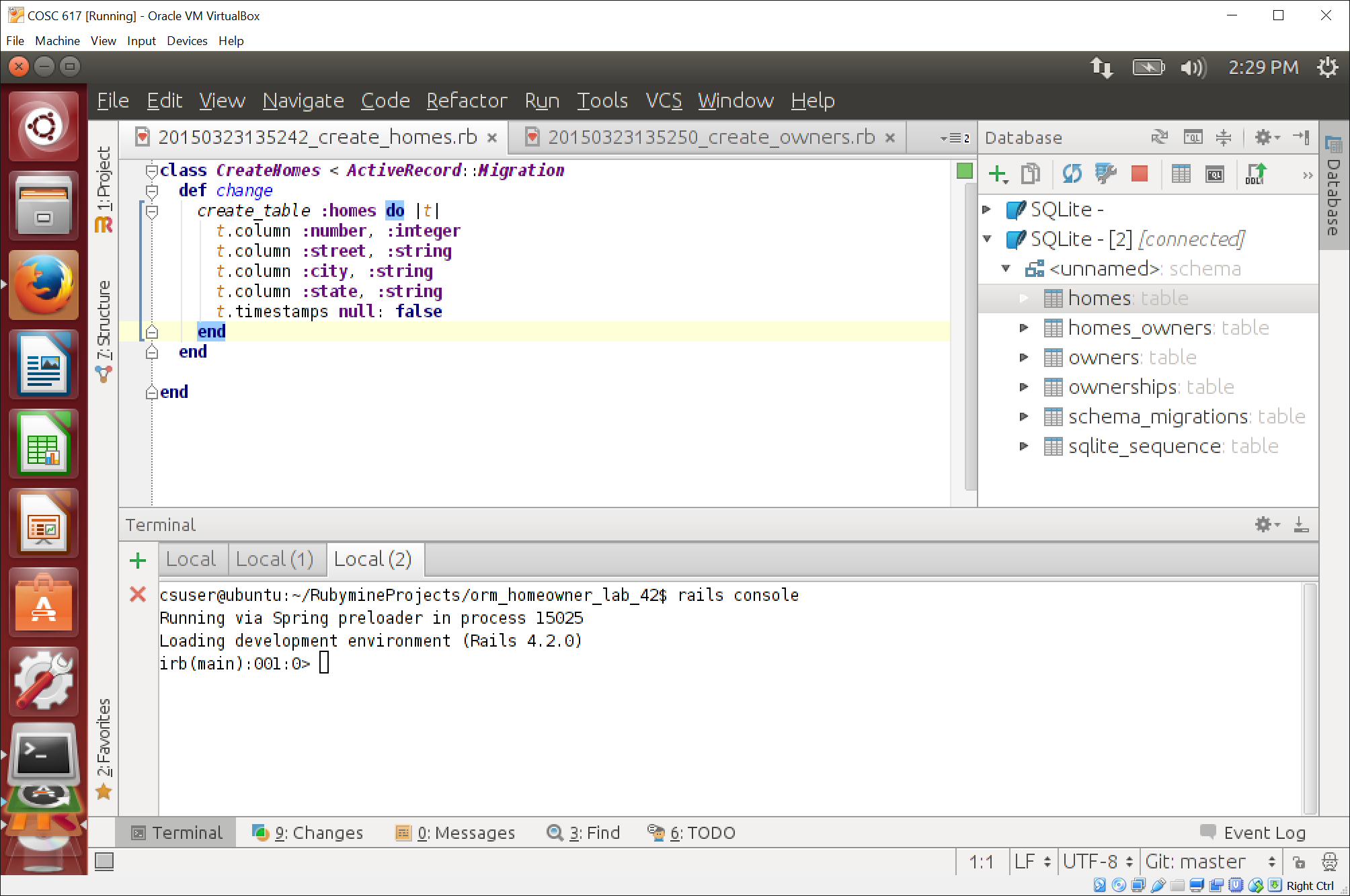


**Implementing many-to-many relationships using implicit join tables (10)**

* 1. Study the migrations for creating the homes, owners, and homes\_owners table. Understand the schema.
     1. Open the database in your favorite GUI tool and see the result of the migrations (ignore the ownerships table for now). How many tables does the database contain in total (again, ignore ownerships table)? How many were you expecting?

**Ignoring ownerships, we expected three tables: homes, homes\_owners, and owners.**

**We were expecting four (not ignoring ownerships): homes, homes\_owners, owners, and ownerships.**



* 1. Study the models for homes and owners. Notice the has\_and\_belongs\_to\_many method. Think about what that method expects in the schema design.
     1. What tables and fields does it expect to be present?

**Tables: homes and owners**

**Fields: id (homes table) and id (owners table) since they are the table keys**

* + 1. Why does the homes\_owners table not have a corresponding model?

**Because there is an implicit join on the tables owners and homes**

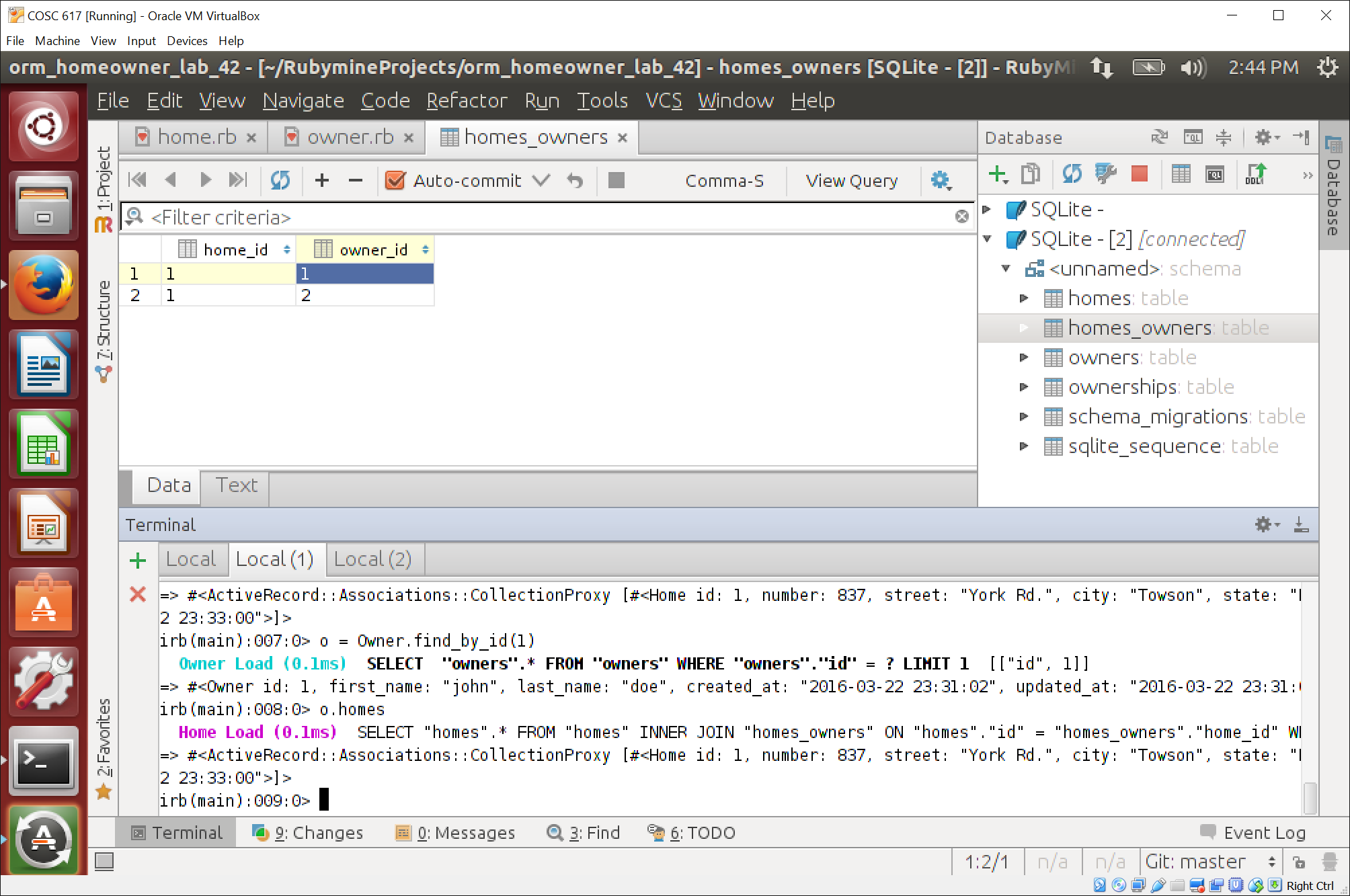
* 1. Enter the statements in the add\_homes method in the main controller that correspond to the implicit relationship in the **rails console** **one-by-one**. **Please do not copy and paste.** 
     1. See the result after each statement. This result is shown clearly in the console. – what is the difference between the ‘create’ and the ‘new’ method in Activerecord?

**Create: automatically inserts the record into the table**

**New: saves it locally into a variable, still needs to be saved into the table**

* + 1. What is the net effect on the database after all statements are run? Are the owner\_id and home\_id fields in the relationship table updating correctly.

**The homes\_owners table was updated correctly for the following home\_id from the homes table and the owner\_id from the owners table.**

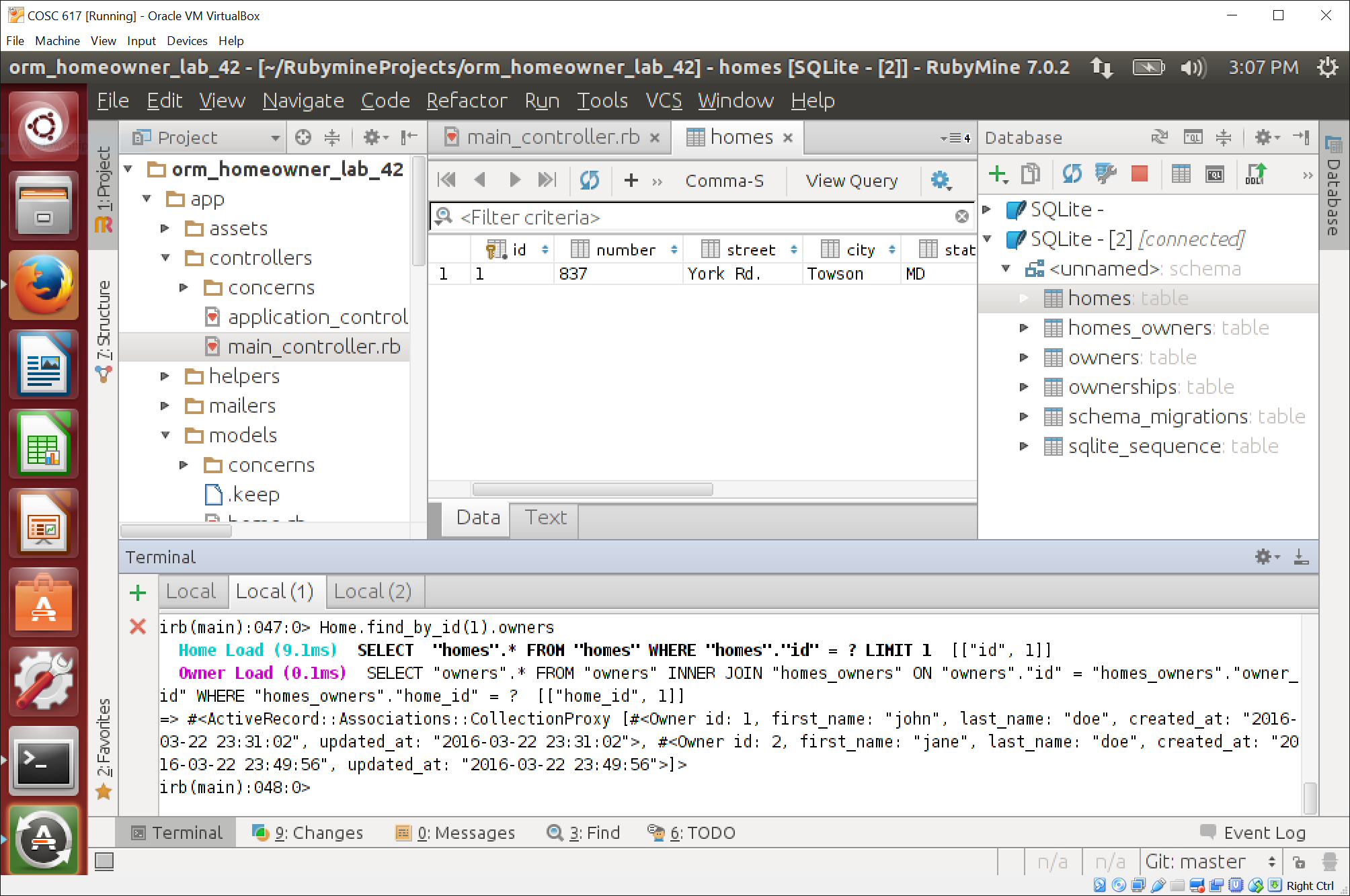
****

* + 1. Think about how the statement ‘h.owners << Owner.find (1)’ is working. Where is the ‘h.owners’ method written?

**The h.owners method is not written explicitly in the code. It is actually an “automatic” join because of the “has\_and\_belongs\_to\_many”.**

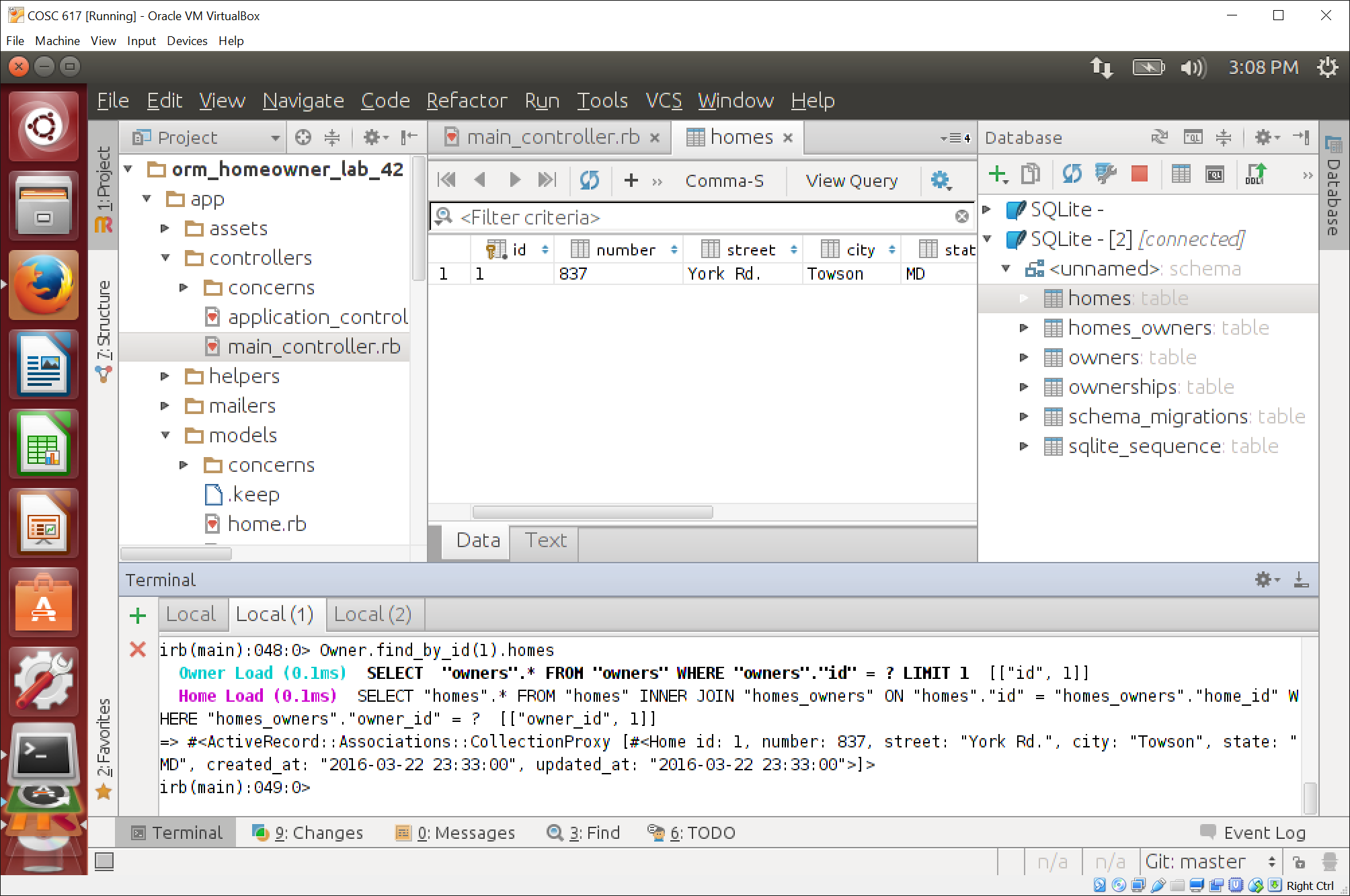
* + 1. Type the command to list all of the owners of the home you just created. You have to figure this out yourself (it’s not in the file).

**h.owners [since h.owners was set to Owner.find(1) in step iii] or Home.find\_by\_id(1).owners [since the home has an id of 1]**

****

* + 1. Type a command to do the reverse – list all the homes of an owner with id=1. Paste code here.

**Owner.find\_by\_id(1).homes**



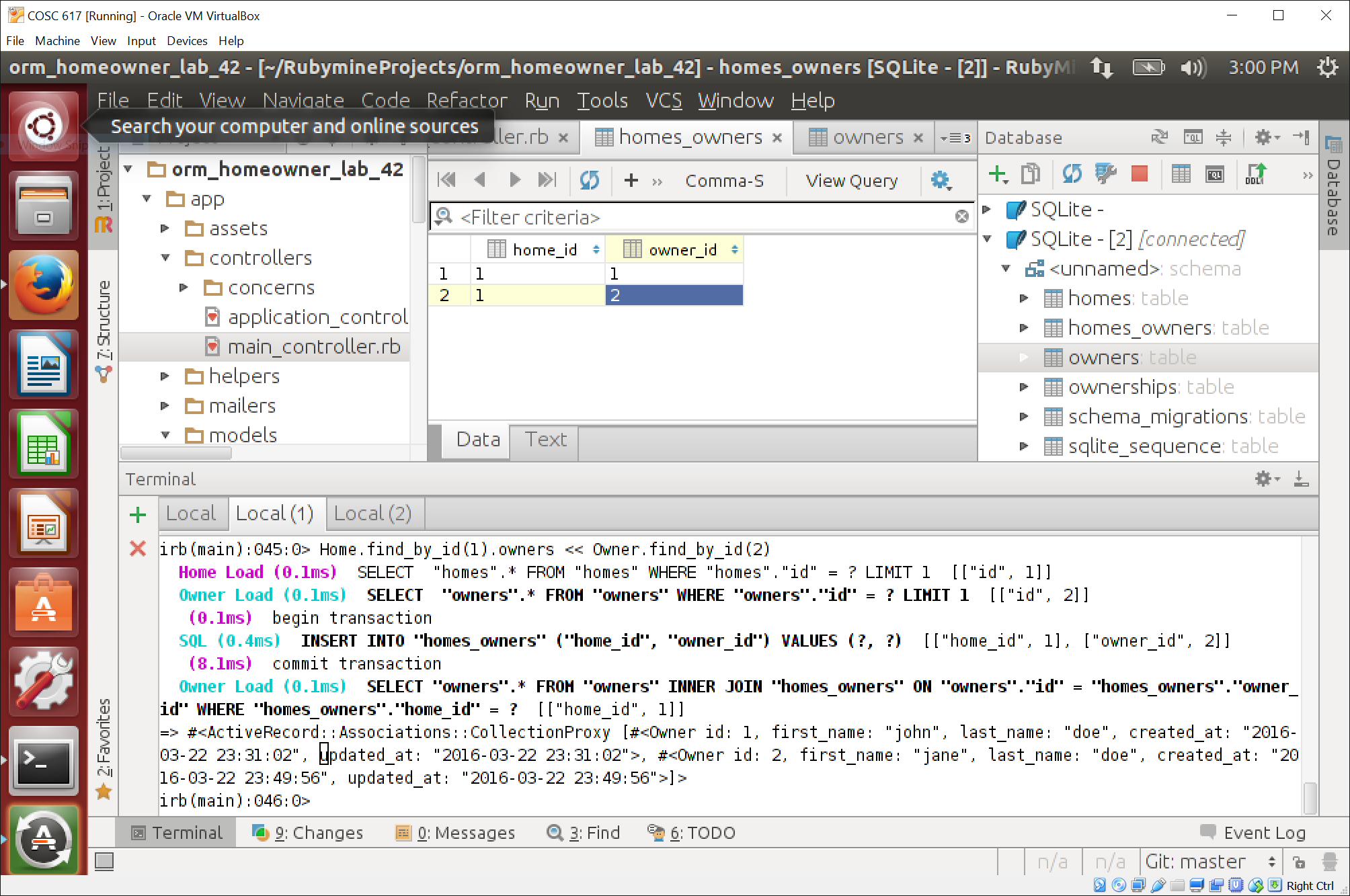
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Complete till here before lecture** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. Add a second owner to the home (837 York Rd). Paste the code here.

**Owner.create(:first\_name=>"jane", :last\_name=>"doe")**

**Home.find\_by\_id(1).owners << Owner.find\_by\_id(2) [new owner “jane doe” has id 2]**

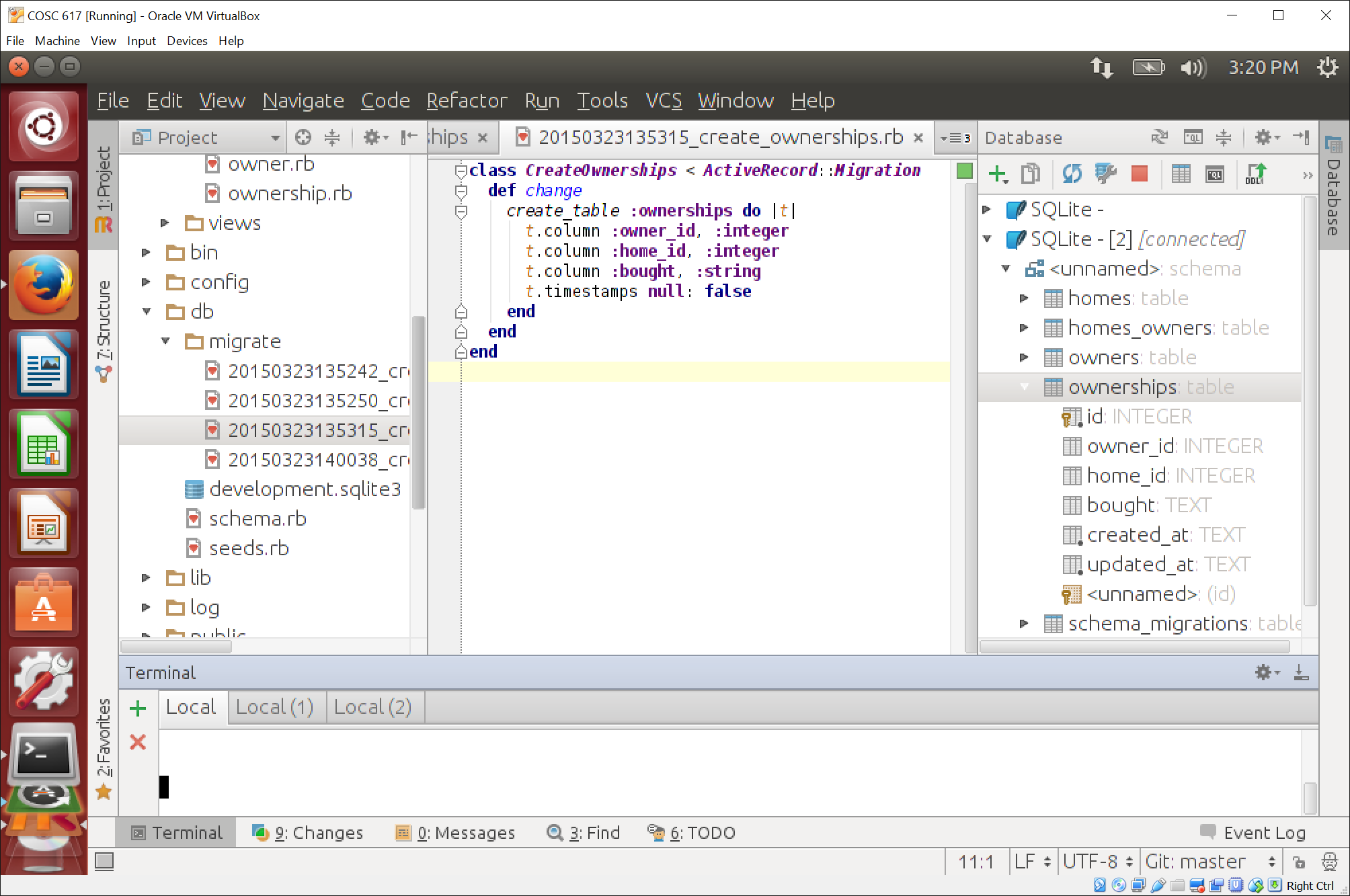
****

****

**Implementing many-to-many relationships using explicit join tables (10)**

1. This is a continuation of the home\_owners example from last lab. The ownerships table is used as the join table here. Study the migration for creating the ownerships table. Understand the schema in the DB.
2. What is the extra field that needs to be recorded about an ownership of a home? This field is the reason we need an explicit model in ActiveRecord.

**The extra field is the “bought” field, which is a string.**

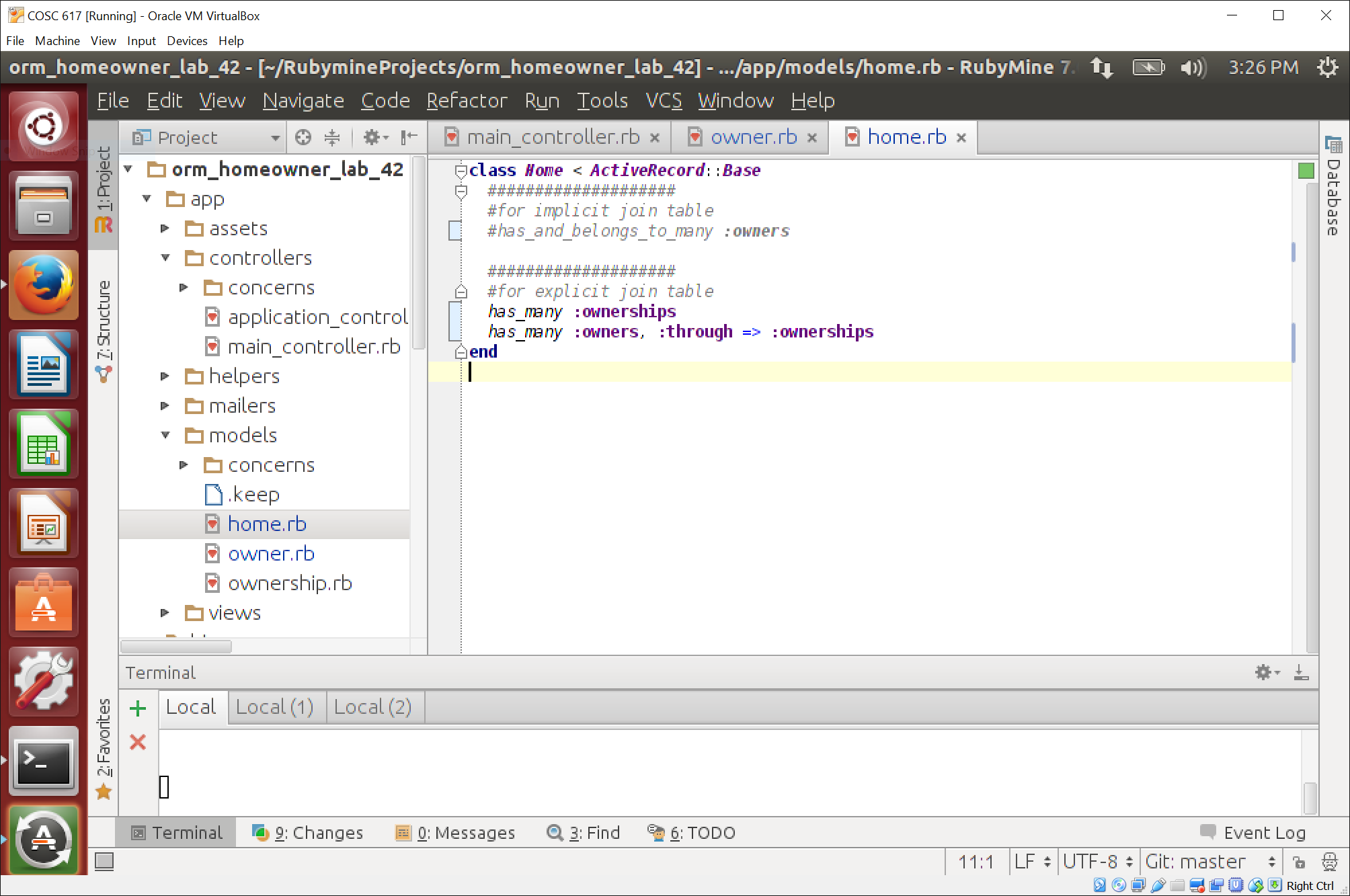
****

1. Study the models for homes, owners, and ownerships. *Uncomment* the part needed for the explicit join table and *comment*  the part of the implicit join table. Notice the ‘:through’ clause. Think about what that method expects in the migrations.
2. What tables and fields does it expect to be present? What does the ownerships table do?

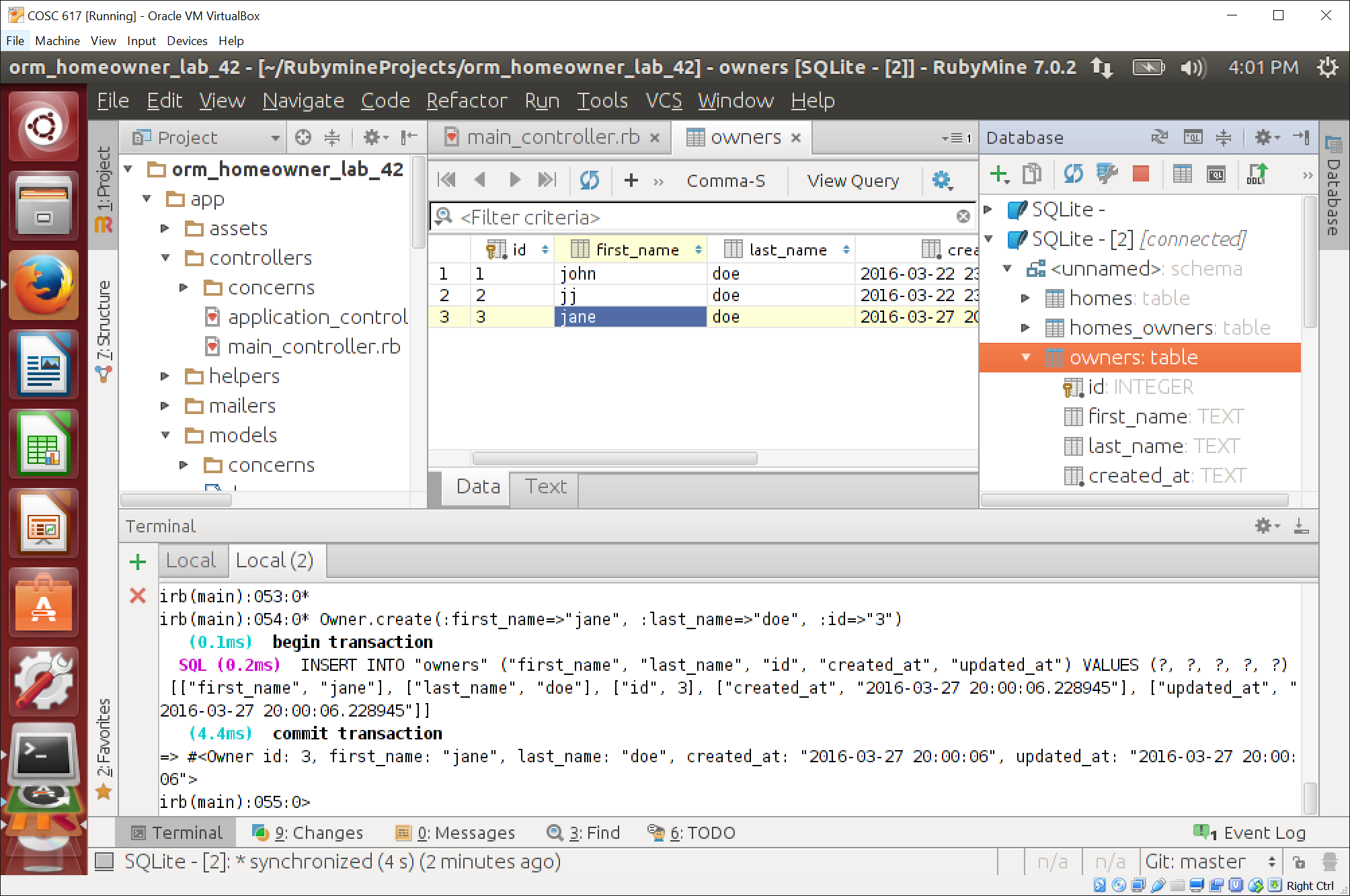
**Tables: owner, homes, and ownerships**

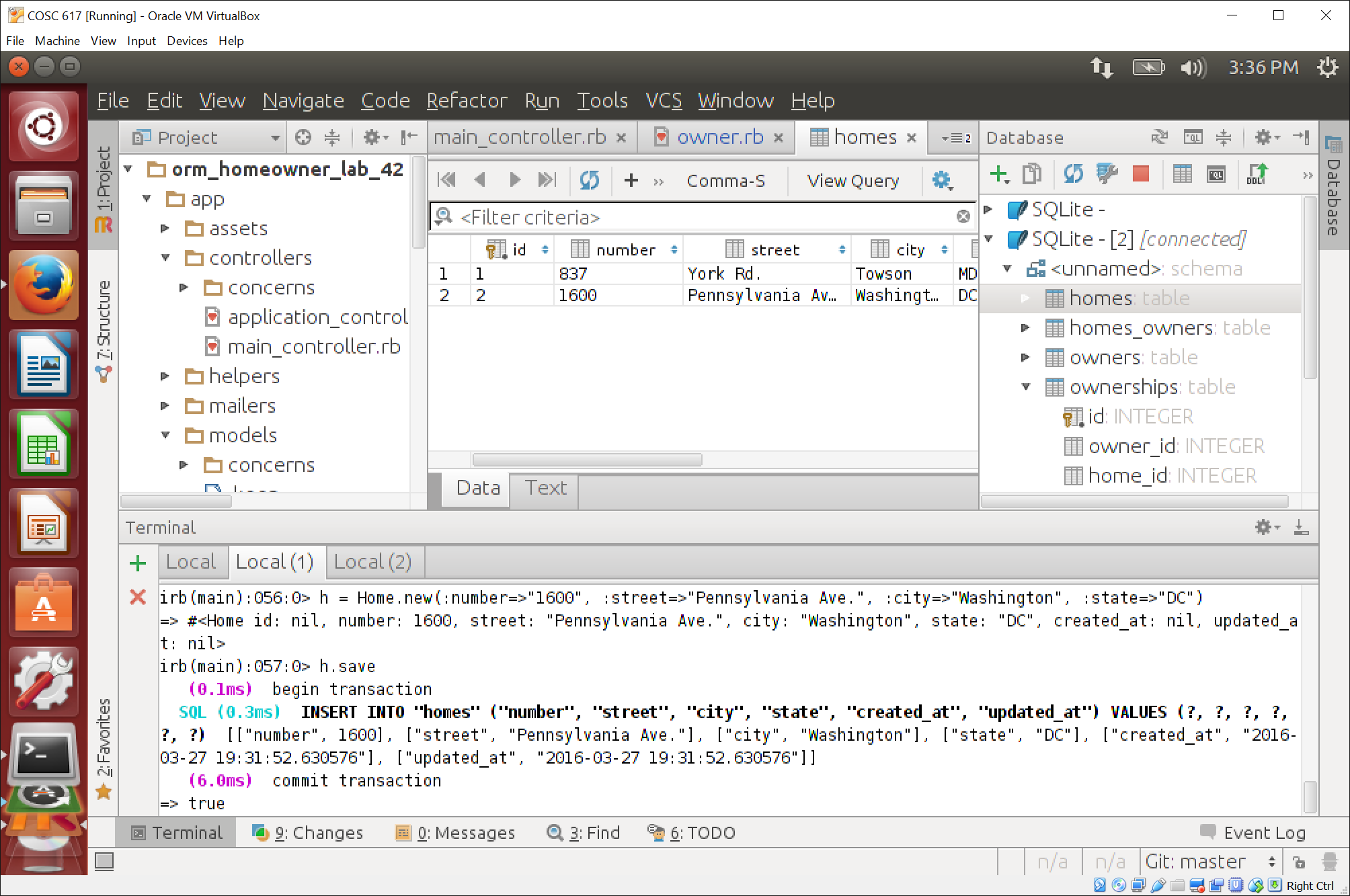
**Fields: owner\_id (corresponds to the id from the owner table), home\_id (corresponds to the id from the home table), and bought (string)**

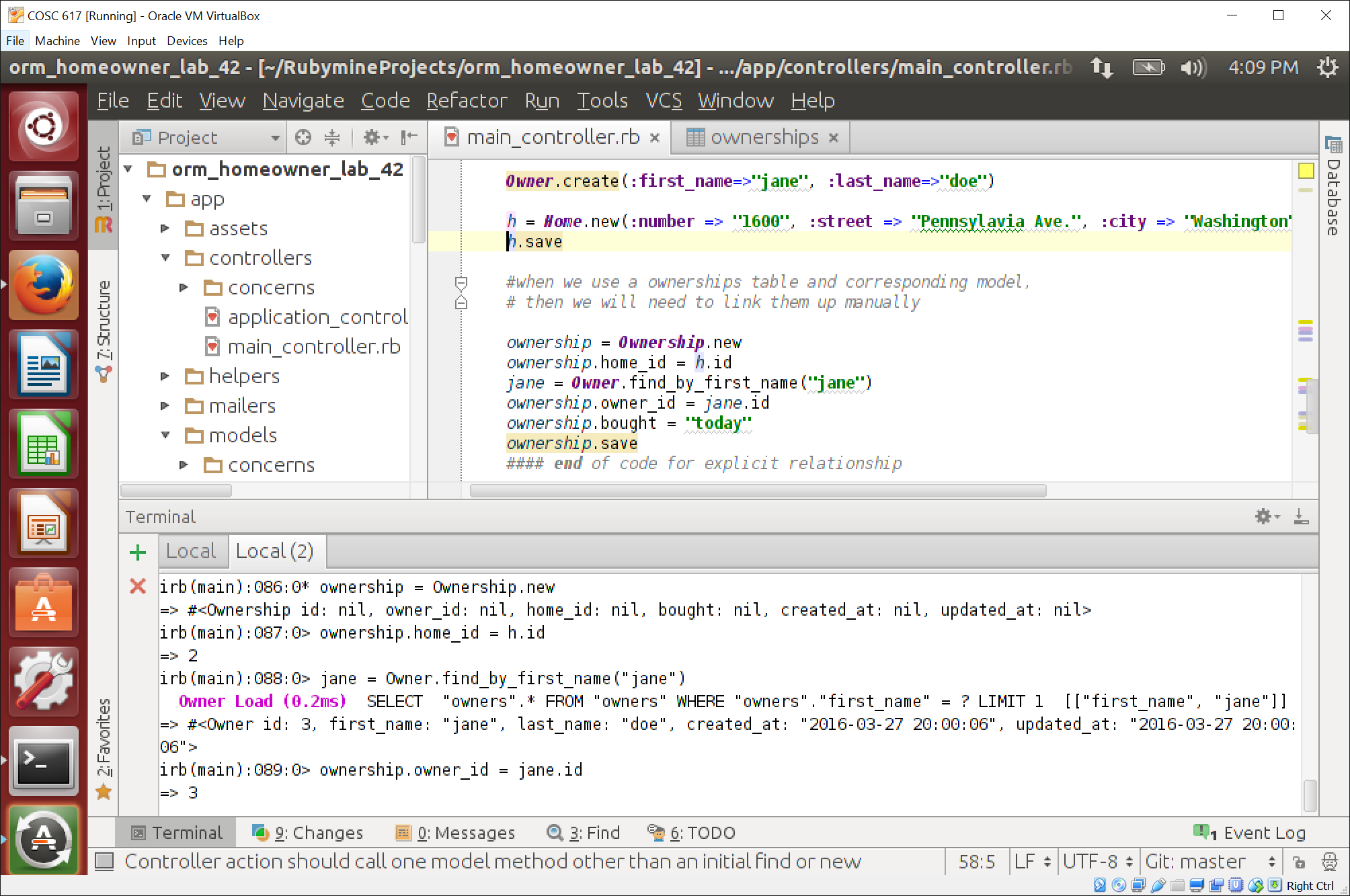
**The ownerships table stores the relationship between an owner and a home through their respective table ids and includes a string for purchase date (bought).**

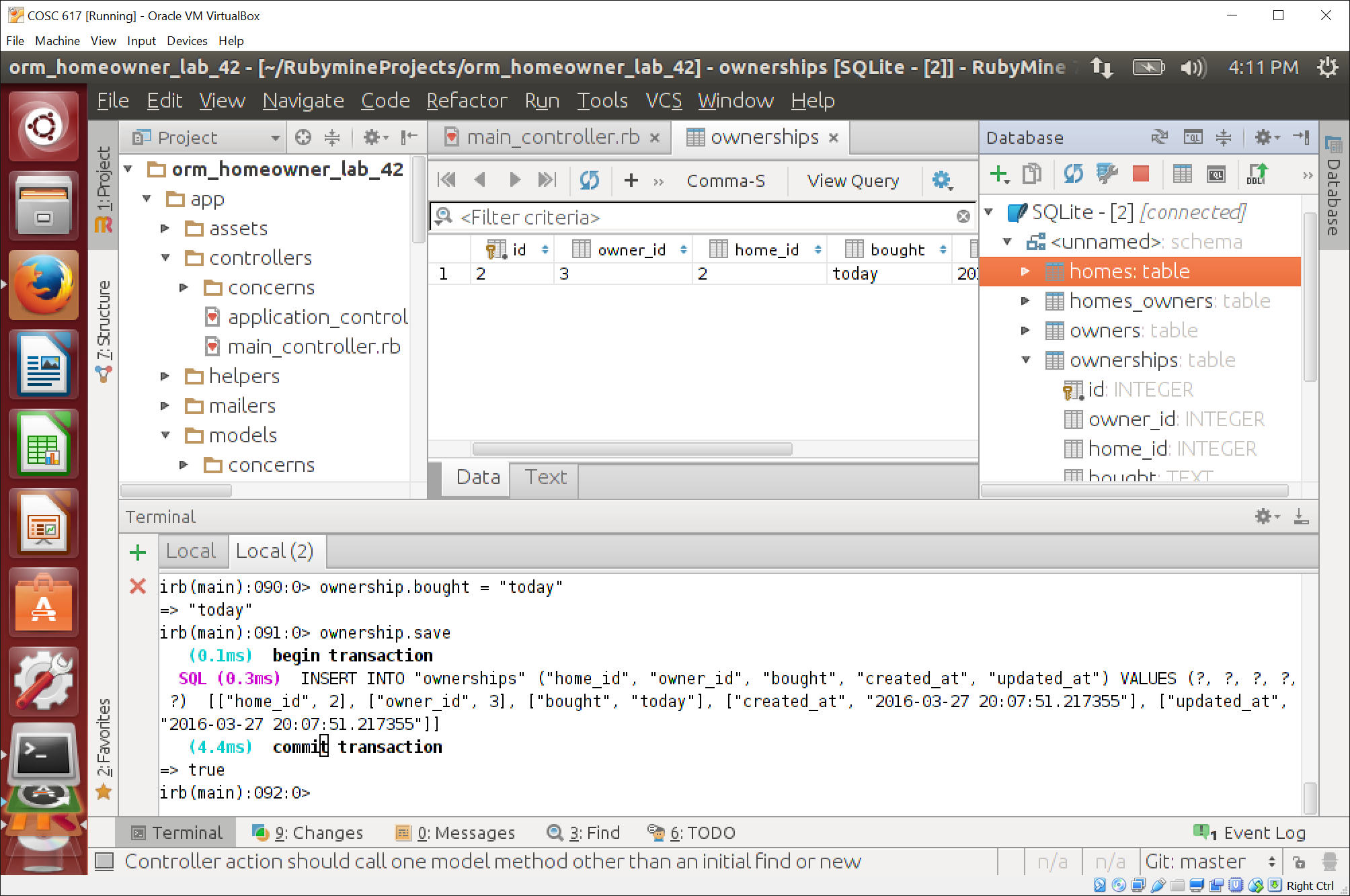


1. **Restart rails console first to refresh the code it uses for the models.** Enter the statements in the add\_homes method in the main controller that correspond to the explicit relationship in the rails console **one-by-one**.
2. See the result after each statement.









1. What is the net effect on the database after all statements are run? Are the owner\_id and home\_id fields in the relationship table updating correctly.

**After all the statements are run, there is a record in the ownerships table with the owner\_id for “jane” and the home\_id for the newly added home (1600 Pennsylvania).**

* 1. Write the code (in rails console) to create a second owner for the home using the explicit join table. Paste the code here.

**Owner.create(:first\_name=>”jay”, :last\_name=>”doe”)**

**ownership = Ownership.new**

**ownership.home\_id = h.id [where h is the newly added home with id 2]**

**jay = Owner.find\_by\_first\_name(“jay”)**

**ownership.owner\_id = jay.id**

**ownership.bought = “today”**

**ownership.save**

