Final Report

December 8, 2017

Team Members:

Jacqueline Ali Cordoba, Kayla Brady, Samuel Pinheiro

Group Name:

AlicordobaBradyPinheiro

Part 1: README

Rails must be installed first. You can check if you have it with `rails --version`. If you don’t have it, you can install it from here: http://railsinstaller.org/en

Then, run the following in your terminal:

`cd aquarium_tycoon`

`bundle install`

`rake db:drop db:create db:migrate`
  -If this step doesn’t work correctly then run ‘bin/rails db:migrate RAILS_ENV=development’, and try running the step again
  - if your msql root is password protected, you may get an error on this step. If so -
  `cd config`
  Edit `database.yml` file - enter your password after “password:” on line 17

`cd ..`

`mysql -u root -p aquarium_tycoon_development < populateDatabase.sql`
  -If you get ‘mysql: command not found’, run ‘export PATH=${PATH}:/usr/local/mysql/bin/’ and run the previous step again

‘cd aquarium_tycoon’

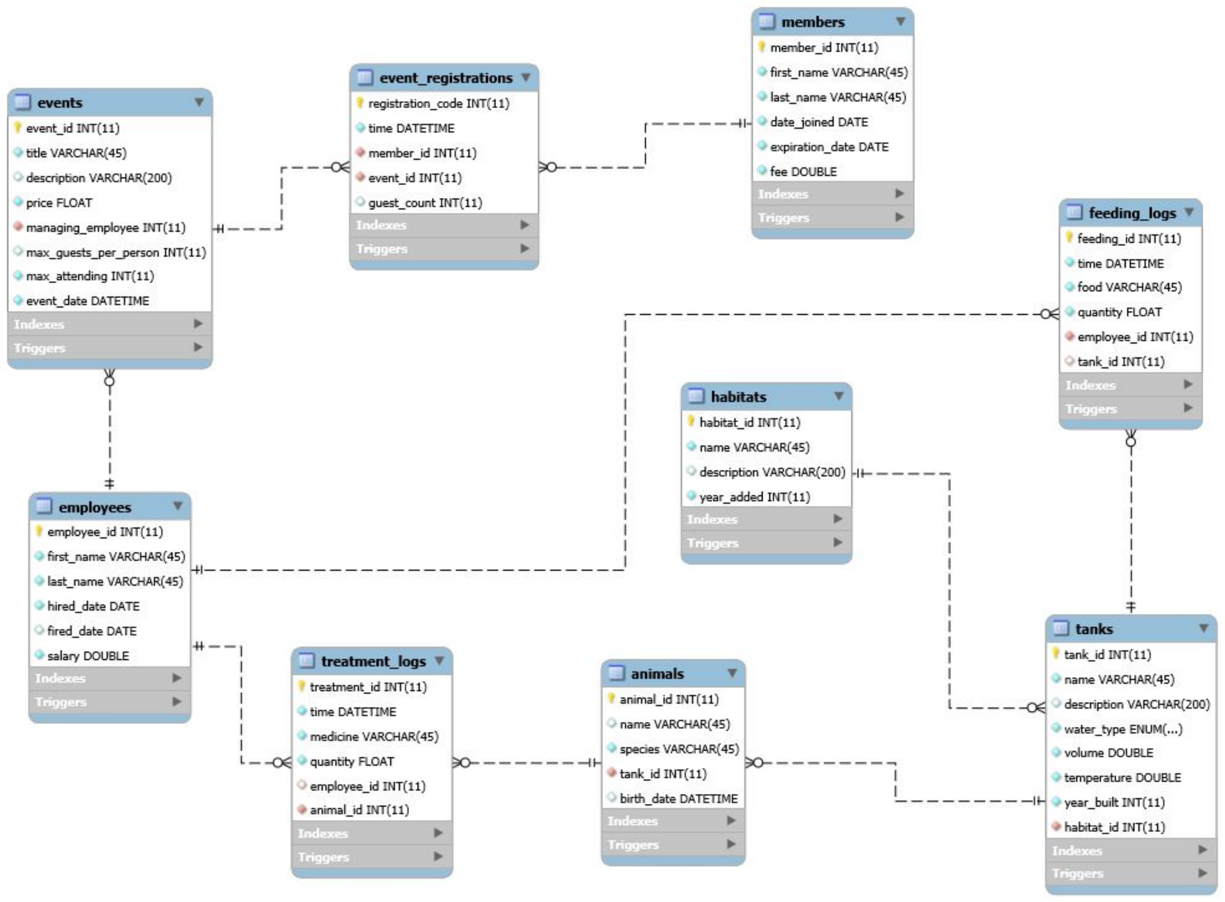
‘rails server’

Part 2: Technical Specifications

We’re using Ruby on Rails, a popular server-side framework for building web applications, to develop our API. We chose it because one member of our team already has production experience with the language, and its expressiveness and similarity to English makes it easy to read and learn.

The Ruby on Rails API will handle endpoint routing to connect user inputs and requests on the front-end with desired actions in the database. The API calls SQL functions and some stored procedures/triggers that we've created. Some calls to the database are done through ActiveRecord instead as calling procedures directly in the SQL database isn't part of the "rails standard."

Part 3: EER/UML Diagram



Part 4: User Flow/Interactions

Each of the following instructions begin after the application has been launched, as described in the README section.

Animals

READ

1. Navigate to: http://localhost:3000/animals
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Animal #47 - Toady - will produce:

    Animal: 47
    Name: Toady
    Species: Toad Crab
    Tank: 5
    Birth date: 2016-05-06 00:00:00 UTC

CREATE

1. Navigate to http://localhost:3000/animals
2. Scroll to click the “New Animal” button
3. Enter an animal name - “Michelangelo”
4. Enter a species - “Loggerhead Turtle”
5. Select a tank - “Carribean Shallows”
6. Enter a birthdate - 2017 - November - 16 - 01 : 01
7. Click “Create Animal” - on success, the record will be shown with message “animal successfully added”
8. Click “Back” to navigate back to the page listing all animals

→ Trigger add_animal is called - any error message will be delivered at the top of the page. (ex: missing species will result in error “Must input a species”

UPDATE

1. Navigate to http://localhost:3000/animals
2. To update a single record, click the “edit” button next to that record.
3. Ex: Edit Animal #47 - Toady
4. Change name to “LevelUp”
5. Change tank to “Kids Tank”
6. Click “Update Animal”
7. The updated record will appear with message “Animal was successfully updated”

    Animal: 47
    Name: LevelUp
    Species: Toad Crab
    Tank: 8
    Birth date: 2016-05-06 00:00:00 UTC

→Trigger update_animal is called- any error message will be delivered at the top of the page. (ex: trying to change the species will result in error “Cannot change species”)

DELETE

1. Navigate to http://localhost:3000/animals
2. To remove an animal, select “destroy” next to that record.
3. Ex: Destroy Animal #47 - LevelUp (formerly Toady, changed during update)

4. An “Are you sure” box will pop up - click “okay”
5. At the top of the page, message “Animal was successfully removed” will appear.

Tanks

READ

1. Navigate to http://localhost:3000/tanks
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Tank # 1 - River will produce:

    Tank: 1
    Name: River
    Description: Meet the menacing inhabitants of the Amazon River!
    Water type: fresh
    Volume: 2000.0
    Temperature: 76.0
    Year built: 2015
    Habitat: 1

CREATE

1. Navigate to http://localhost:3000/tanks
2. Scroll to click the “New Tank” button
3. Enter an tankname - “Jellyfishes”
4. Enter a description - “Jellyfish of all shapes and sizes!”
5. Select a water type- “salt”
6. Enter a volume - 15000
7. Enter a temperature - 79.2
8. Select a habitat - “Coral Reef”
9. Click “Create Tank” - on success, the record will be shown with message “Tank successfully added”
10. Click “Back” to navigate back to the page listing all animals

→ Trigger add_tank is called - any error message will be delivered at the top of the page. (ex: volume of 150 will result in error “Volume must be between 500 and 100,000 gallons”

UPDATE

1. Navigate to http://localhost:3000/tanks/
2. To update a single record, click the “edit” button next to that record.
3. Ex: Edit Tank #1 - River
4. Increase the volume to 2500
5. Click “Update Tank”
6. The updated record will appear with message “Tank was successfully updated”

    Tank: 1

Name: River
Description: Meet the menacing inhabitants of the Amazon River!
Water type: fresh
Volume: 2500.0
Temperature: 76.0
Year built: 2017
Habitat: 1

→Trigger update_tank is called- any error message will be delivered at the top of the page. (ex: trying to decrease the volume will result in error “Cannot shrink a tank.”)

DELETE

1. Navigate to http://localhost:3000/tanks
2. To remove a tank, select “destroy” next to that record.
3. Ex: Destroy Tank #1 - River
4. An “Are you sure” box will pop up - click “okay”
5. At the top of the page, message “Tank was successfully removed” will appear.
6. Navigate to http://localhost:3000/animals - all animals in Tank 1 have been removed.

Habitats

READ

1. Navigate to http://localhost:3000/habitats
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Habitat # 1 - Amazon Rainforest will produce:

   Habitat: 1
   Name: Amazon Rainforest
   Description: Explore the exotic aquatic animals of the Amazon Rainforest. Located in South America, the Amazon Rainforest is home to thousands of diverse species.
   Year Added: 2015

CREATE

1. Navigate to http://localhost:3000/habitats
2. Scroll to click the “New Habitat” button
3. Enter an name - “Subterranean depths”
4. Enter a description - “The deepest depths of the ocean”
5. Click “Create Habitat” - on success, the record will be shown with message “Habitat successfully added”
6. Click “Back” to navigate back to the page listing all habitats

→ Trigger add_habitat is called - any error message will be delivered at the top of the page

UPDATE

7. Navigate to http://localhost:3000/habitats
8. To update a single record, click the “edit” button next to that record.
9. Ex: Edit Habitat #1 - Amazon Rainforest
10. Change description to “Anything”
11. Click “Update Habitat”
12. The updated record will appear with message “Habitat was successfully updated”

    Habitat: 1
    Name: Amazon Rainforest
    Description: Anything
    Year Added: 2017

→Trigger update_habitat is called- any error message will be delivered at the top of the page

DELETE

7. Navigate to http://localhost:3000/habitats
8. To remove a habitat, select “destroy” next to that record.
9. Ex: Destroy Habitat #1 - Amazon Rainforest
10. An “Are you sure” box will pop up - click “okay”
11. At the top of the page, message “Habitat was successfully destroyed” will appear.
12. Navigate to http://localhost:3000/tanks - all tanks in habitat 1 are removed
13. Navigate to http://localhost:3000/animals - all animals in the corresponding tanks are removed

Feeding_Logs

READ

1. Navigate to: http://localhost:3000/feeding_logs
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Feeding Log #1 - 11/01/2017 - will produce:

    Feeding: 1
    Time: 2017-11-01 00:00:00 UTC
    Food: sardines
    Quantity: 60.0
    Employee: 6
    Tank: 12

CREATE

1. Navigate to http://localhost:3000/feeding_logs
2. Click the “New Feeding Log” button
3. Enter a time - 01: 29
4. Enter a food - “sardines”

5. Enter a quantity - 15
6. Select an employee - Edleman, Alice
7. Select a tank - Chinstrap Penguins
8. Click “Create Feeding log” - on success, the record will be shown with message “Feeding Log was successfully added”
9. Click “Back” to navigate back to the page listing all animals

→ Trigger add_feeding is called - any error message will be delivered at the top of the page. (ex: missing food will result in error “Must input food type.”
→ Procedure current_employees() is called to limit employee selection to only current employees

UPDATE
This is not an option for feeding logs - past feeding logs should never be updated.
DELETE
This is not an option for feeding logs - past feeding logs should never be deleted.

Treatment_Logs

4. Navigate to: http://localhost:3000/treatment_logs
5. To view a page for just a single record, click the “show” button next to that record.
6. Ex: Show for Treatment Log #1 - 01/01/2017 - will produce:

    Treatment: 1
    Time: 2017-01-01 00:00:00 UTC
    Medicine: Cophicin
    Quantity: 20.0
    Employee: 1
    Animal: 150

CREATE

10. Navigate to http://localhost:3000/treatment_logs
11. Click the “New Treatment Log” button
12. Enter a time - 01: 29
13. Enter a medicine - “advil”
14. Enter a quantity - 2
15. Select an employee - Edleman, Alice
16. Select an animal - 154 Handlebar
17. Click “Create Treatment log” - on success, the record will be shown with message “Treatment log was successfully added”
18. Click “Back” to navigate back to the page listing all animals

→ Trigger add_feeding is called - any error message will be delivered at the top of the page. (ex: missing food will result in error “Must input a medicine”

→ Procedure current_employees() is called to limit employee selection to only current employees

UPDATE
This is not an option for feeding logs - past feeding logs should never be updated.
DELETE
This is not an option for feeding logs - past feeding logs should never be deleted.

Employees

READ

1. Navigate to: http://localhost:3000/employees
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Employee #2 - Bob - will produce:

    Employee: 2
    First name: Bob
    Last name: Franklin
    Hired date: 2015-01-01
    Fired date:
    Salary: 67000.0

CREATE

1. Navigate to http://localhost:3000/employees
2. Scroll to click the “New Employee” button
3. Enter an employee first name - “Avon”
4. Enter an employee last name - “Barksdale”
5. Enter an employee salary >= 40,000 - 40,000
9. Click “Create Employee” - on success, the record will be shown with message “Employee successfully added”
10. Click “Back” to navigate back to the page listing all employees

→ Trigger add_employee is called - any error message will be delivered at the top of the page. (ex: salary lower than 40,000 will result in error “40000 is the minimum salary”)
UPDATE

1. Navigate to http://localhost:3000/employees
2. To update a single record, click the “edit” button next to that record.
3. Ex: Edit Employee #2 - Bob
4. Change first name to “Robert”
5. Change salary to 69000
6. Click “Update Employee”
7. The updated record will appear with message “Employee updated”

    Animal: 2
    First name: Robert

Last name: Franklin
Hired date: 2017-12-07
Fired date:
Salary: 69000.0

→Trigger update_employee is called- any error message will be delivered at the top of the page. (ex: salary lower than 40,000 will result in error “40000 is the minimum salary”)

DELETE

1. Navigate to http://localhost:3000/employees
2. To fire an employee, select “fire” next to that record.
3. Ex: Fire Employee #9 - Maria Richard
4. An “Are you sure” box will pop up - click “okay”
5. At the top of the page, message “employee fired” will appear. Today’s date should appear as the fired date.
6. Firing an employee associated with the event will display error message on top of page: “Cannot fire this employee until all events under their management are transferred to another employee”

→ Trigger update_employee is called, and uses the function events_managed(emp int) to determine if an employee is still listed as an event manager and therefore cannot yet be fired.

Events

READ

1. Navigate to: http://localhost:3000/events
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Event #2 - New England Shore Lecture - will produce:

Event: 2
Title: New England Shore Lecture
Price: 15.0
Managing Employee: 3
Max Guests per Person: 4
Max attending: 60
Event date: 2018-03-01 19:00:00 UTC

CREATE

1. Navigate to http://localhost:3000/events
2. Scroll to click the “New Event” button
3. Enter a title - “Shark Feeding”
4. Enter a price - 20
5. Click a managing employee from the dropdown - Quinn, Laura
6. Enter max guests per person - 2
7. Enter max attending - 100
8. Enter event date - 2017 December 8 20:30

9. Click “Create Event” - on success, the record will be shown with message “Event was successfully updated”
10. Click “Back” to navigate back to the page listing all events

→Trigger invalid_event_insert is called- any error message will be delivered at the top of the page

→ Procedure current_employees() is called to limit manager selection to only current employees

UPDATE

1. Navigate to http://localhost:3000/events
2. To update a single record, click the “edit” button next to that record.
3. Ex: Edit Event #3 - Octopus Night
4. Change price to 25.0
5. Click “Update Event”
6. The updated record will appear with message “Event was successfully updated”

    Event: 3
    Title: Octopus Night
    Price: 25.0
    Managing Employee: 3
    Max Guests per Person: 4
    Max attending: 60
    Event date: 2018-06-01 19:00:00 UTC

→Trigger invalid_event_update is called- any error message will be delivered at the top of the page

→ Procedure current_employees() is called to limit manager selection to only current employees

DELETE

1. Navigate to http://localhost:3000/events
2. To delete an event, select “cancel” next to that record
3. Ex: Cancel Event #10 - Behind the scenes - penguins
4. An “Are you sure” box will pop up - click “okay”
5. At the top of the page, message “Event was successfully destroyed.” will appear. The event should disappear from records

Members

READ

1. Navigate to: http://localhost:3000/members
2. To view a page for just a single record, click the “show” button next to that record.
3. Ex: Show for Member #2 - Barry - will produce:

    Member: 2

First name: Barry
Last name: Yelp
Date joined: 2011-01-01
Expiration date: 2017-04-04
Fee: 400.0

CREATE

1. Navigate to http://localhost:3000/members
2. Scroll to click the “New Member” button
3. Enter an employee first name - “Edward”
4. Enter an employee last name - “Elric”
5. Enter a fee from the dropdown - 100
6. Click “Create Member” - on success, the record will be shown with message “Member successfully added”
7. Click “Back” to navigate back to the page listing all members

→ Trigger invalid_member_insert is called - any error message will be delivered at the top of the page

UPDATE

1. Navigate to http://localhost:3000/members
2. To update a single record, click the “renew” button next to that record.
3. Ex: Edit Member #2 - Barry
4. Change fee to 200
5. Click “Update Member”
6. The updated record will appear with message “Member was successfully updated”

Member: 2
First name: Barry
Last name: Yelp
Date joined: 2011-01-01
Expiration date: 2017-04-04
Fee: 200.0

→Trigger invalid_member_update is called- any error message will be delivered at the top of the page

DELETE

1. Navigate to http://localhost:3000/members
2. To delete a member, select “cancel” next to that record
3. Ex: Cancel Event #2 - Barry
4. An “Are you sure” box will pop up - click “okay”
5. At the top of the page, message “Member was successfully destroyed.” will appear. The member should disappear from records

Event_Registrations

READ

1. Navigate to http://localhost:3000/event_registrations
4. To view a page for just a single record, click the “show” button next to that record.
5. Ex: Show for Registration # 1 will produce:

    Registration code: 1
    Time: 2017-10-15 00:00:00 UTC
    Member: 8
    Event: 1
    Guest count: 3

CREATE

11. Navigate to http://localhost:3000/event_registrations
12. Click the “New Event Registration” button
13. Select which member to register - “Apple, Zach”
14. Select which event - “Amazon River Lecture: 2018-01-01
15. Click “Create Event Registration” - on success, the record will be shown with message “Successfully registered for event”
16. Click “Back” to navigate back to the page listing all animals

→ Trigger add_registration_guests is called - any error message will be delivered at the top of the page. (ex: registering member “Ulper, Fanni” for the Amazon River Lecture will result in error “This member is already registered for this event”

→ Procedure current_members() is used to limit the dropdown for members to only those with active membership

→ Procedure upcoming_events() is used to populate the dropdown for events with only upcoming ones

UPDATE

13. Navigate to http://localhost:3000/event_registrations
14. To update a single record, click the “edit” button next to that record.
15. Ex: Edit Registration #1
16. Decrease the guest count to 2
17. Click “Update Event Registration”
18. The updated record will appear with message “Registration successfully updated”

    Registration code: 1
    Time: 2017-12-07 20:54:43 UTC
    Member: 8
    Event: 1
    Guest count: 2

→Trigger update_registration_guests iis called- any error message will be delivered at the top of the page. (ex: trying to change the event will result in error “To register for a different event, cancel this registration and start a new one.
”)

DELETE

14. Navigate to http://localhost:3000/event_registrations
15. To cancel a registration, select “cancel” next to that record.
16. Ex: Cancel Registration #1
17. An “Are you sure” box will pop up - click “okay”
18. At the top of the page, message “Event registration was successfully destroyed” will appear.

Part 5: Lessons Learned

We learned how to better conceptually design databases, keeping in mind the different constraints that would have to be upheld and problems that could occur in the future with every design revision. We now better understand basic backend-frontend connection, and were able to learn ruby on rails to easily do this.

Some alternative design decisions we had thought of were deleting any employees that got fired/left the company. This was quickly discarded when we realized that deleting every record of an employee would have bad consequences on the rest of the database, like affecting the feedings and treatments the employee was associated with. We also considered setting an employee’s ID to null to represent a fire employee, but also discarded this as we figured it was bad practice to make a primary key be able to be set to a null field. We also briefly considered keeping all fired employees in a “formerEmployee” table, but this didn’t end up being as strong as our final design--having an empty “firedDate” field for an employee that could be set properly if an employee was fired or leaves the company.

Part 6: Future Work

People can expect to use this database for an aquarium managing simulator--keeping track of the entities represented in our schema, seeing the various animals, which events are planned, which employees should be fired, etc.

Additionally, this can also be used as a teaching tool to show how different database tables are related to each other. For example, deleting a tank will also delete all the animals associated with that tank so that’s a clear demonstration of how foreign key constraints can be upheld for somebody trying to understand databases.

As for added functionality, our database was made to be fairly easily future-proof. We could add food and medicine tables and have their IDs be foreign keys in the feeding logs and treatment logs--rather than representing the feeding and treatment logs with varchars and float quantities. Additionally, we could also use this to represent multiple aquariums instead of one, having it hold all the necessary entities it's associated with as foreign keys in its table. Switching between aquariums would be simply done through a dropdown in the UI, making it a true *all*-in-one aquarium manager.

Part 7: Group Contributions

Jackie: Proposal, progress report, ruby project setup, migrations, procedures, bulk of rails/frontend development
Kayla: Proposal, progress report, final report, schema/data, error handling & triggers
Sam: Proposal, progress report, final report, schema, use cases, procedures, functions, migrating code to rails

Part 8: Triggers, Procedures, and Functions

Animals

Trigger: update_animal
This trigger checks if an update on in animal is valid before it is completed in the database, otherwise, it throws an error. An invalid animal update is one that changes an animal's species or birth date.
Trigger: add_animal
This trigger checks to make sure an animal is valid before it is added to the database. An invalid animal does not have a species listed, or has a birth date after today.

Tanks

Trigger: update_tank
This trigger checks to make sure an update to a tank is valid before it is completed in the database. An error will be thrown if the tank name or description is removed, the volume is less than 500 or more than 100,000, or if the volume of the tank is decreased. There will also be an error if the volume or temperature is removed, or the temperature is not between 0 and 100 degrees Fahrenheit.
Trigger: add_tank
This trigger checks to make sure a tank is valid before it is added to the database. An error will be thrown if the name, description, volume, or temperature is omitted. An error will also be thrown if the volume is not between 500 and 100,000, or the temperature is not between 0 and 100.

Habitats

Trigger: udpate_habitat
This trigger checks if an update to a habitat is valid before it is saved in the database. An error will be thrown if there is no longer a habitat name or description.
Trigger: add_habitat
This trigger checks if a habitat is valid before it is added to the database. An error will be thrown if it is missing a name or description.

Feeding_Logs

Trigger: add_feeding
This trigger checks if a feeding is valid before it is added to the database. An error will be thrown if a food type is not listed, the quantity is missing, or the quantity is less than zero.

Treatment_Logs

Trigger: add_treatment
This trigger checks if a treatment is valid before it is added to the database. AN error will be thrown if the medicine is not listed, the quantity is missing, or the quantity is less than zero.

Employees

Trigger: update_employee
This trigger checks if updates on an employee are valid before they are made in the database. An invalid employee update is one that changes the salary to less than $40,000 ,fires an already fired employee, and fires an employee that is managing an upcoming event (the manager of those events must be changed before firing the employee).
Function: events_managed(emp int)
This procedure gets the number of upcoming events (events today or later) managed by the employee with the given id.
Trigger: add_employee
This trigger checks if an employee is valid before it is added to the database. An invalid employee has a salary less than $40,000 , has no first name, or no last name.
Procedure: current_employees()
This procedure gets all employees currently working at the aquarium - ones that have a null fired_date.

Events

Trigger: invalid_event_update

This trigger checks if an update to an event record is valid before it is added to the database. The following updates to an event are invalid and will result in an error:

- Removing an event title
- Negative number of guests per person
- Negative price
- Changing the event date to a date that has already passed
- Decreasing the capacity of an event
- Moving an event to a day on which another event is already scheduled
- Moving an event to an earlier day

Trigger: invalid_event_insert
This trigger checks if an event is valid before it is added to the database. An event is considered invalid for any of the following reasons, which will result in an error:

- Missing title
- Already an event scheduled on the date of this event
- Negative guests per person
- Negative price
- Event on a date that has already passed
- Event capacity lower than the guests allowed per person

Procedure: upcoming_events()
This procedure retrieves all upcoming events - ones that take place either today or after. They are returned in ascending order, from the soonest event to the one furthest away.

Members

Procedure: current_members()
This procedure retrieves all current aquarium members - those whose memberships are not expired as of today.
Trigger: invalid_member_update
This trigger checks if an update to a member is valid before it is saved to the database. An update to a member is invalid if their fee is less than $100, or their first or last name is removed.
Trigger: invalid_member_insert
This trigger checks if a member is valid before it is added to the database. A member is invalid if their first or last name is missing, or their fee is less than the $100 minimum.

Event_Registrations

Trigger: update_registration_guests
This trigger checks if an update to an event registration is valid before it is saved in the database. The only thing that can be changed about a registration is the number of guests. An error will be thrown if the event or member associated with the registration is changed. An error will also be thrown if the new guest count is greater than the number of guests allowed at that event.
Trigger: add_registration_guests

This trigger checks if a registration is valid before it is saved in the database. A registration is invalid if the member is already registered for the event, if their guest count exceeds the maximum allotted for the event, or there are not enough tickets left.