Phi Sigma Pi

Andrew Pizzullo
Cody Mcilvaine
Hunter Ginther
James Grady

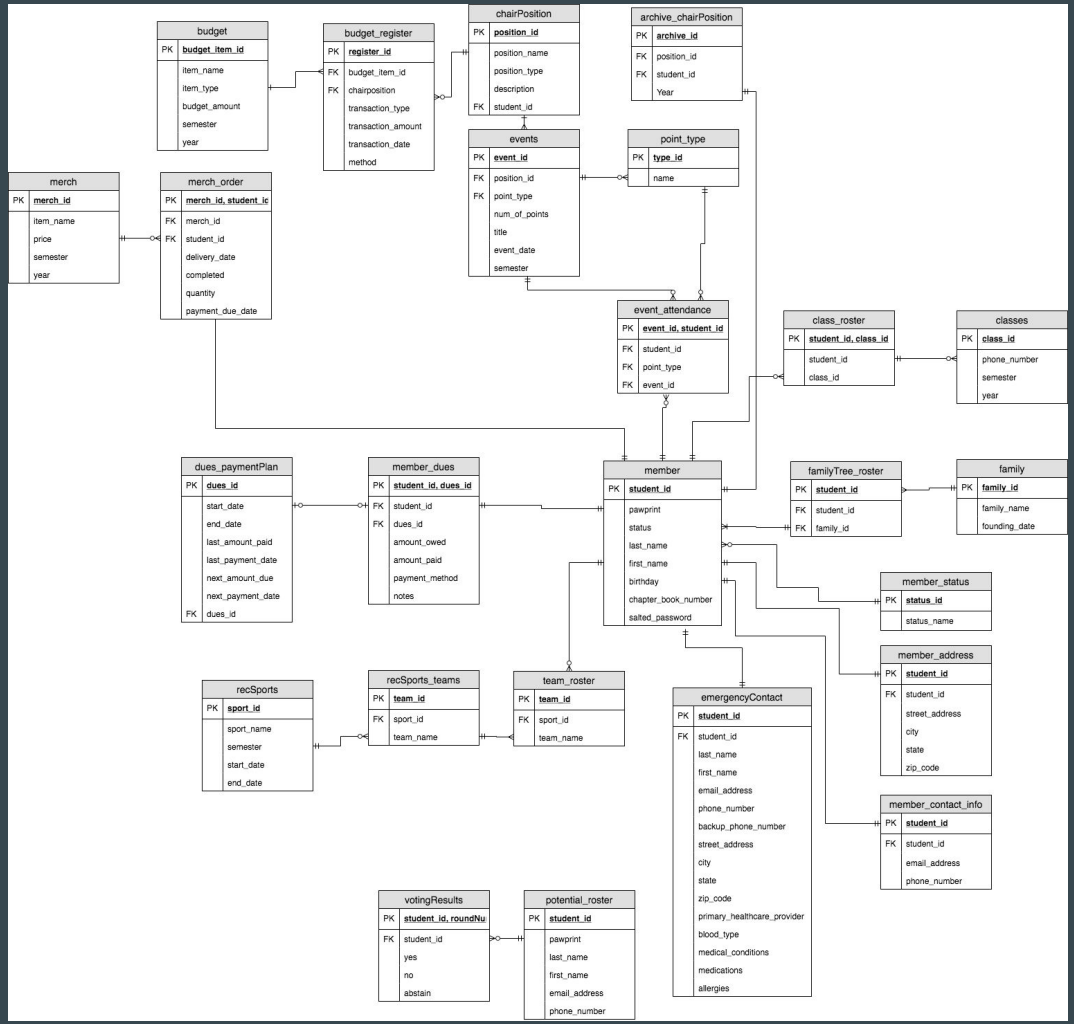
Our Client

Phi Sigma Pi National Honor Fraternity

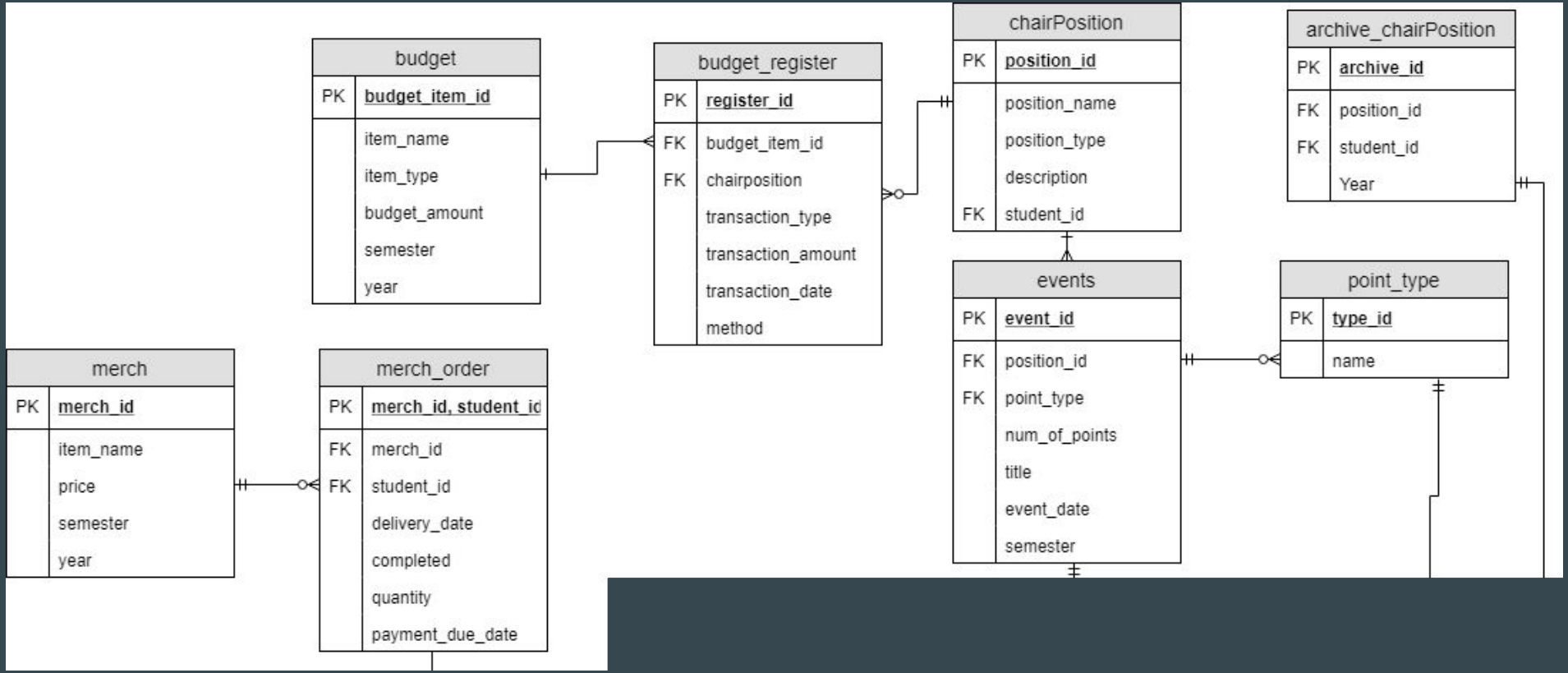
Client Goals/Requirements:

- Centralized/Standard place for all their data
  - Reporting based on different areas of activity (from members)
- Data centers around the members
- Easy to use dashboard for all members to view their own involvement

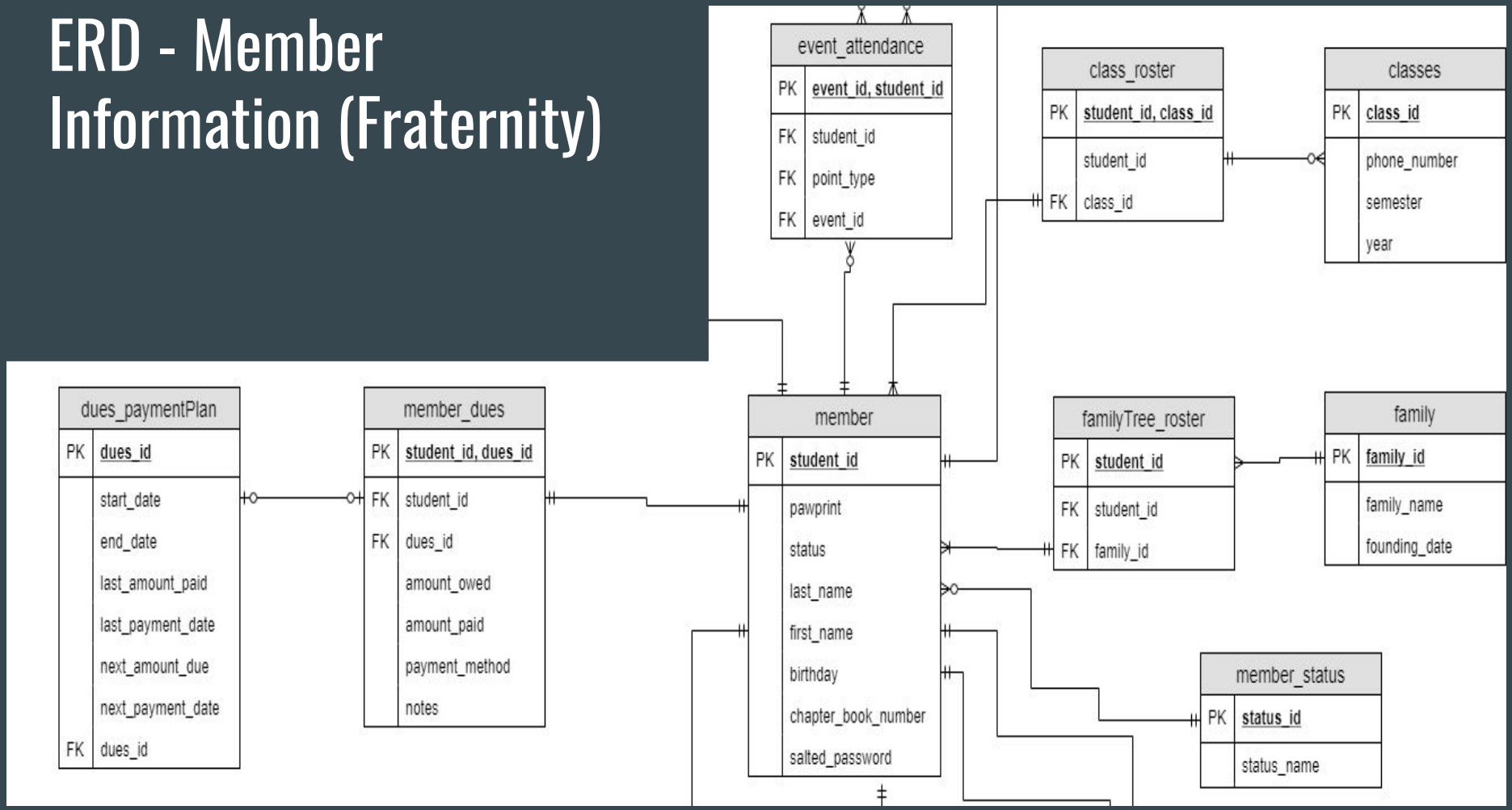
ERD


budget
PK budget_item_id
item_name
item_type
budget_amount
semester
year
budget_register
PK register_id
FK budget_item_id
FK chairposition
transaction_type
transaction_amount
transaction_date
method
chairPosition
PK position_id
position_name
position_type
description
FK student_id
archive_chairPosition
PK archive_id
FK position_id
FK student_id
Year
events
PK event_id
FK position_id
FK point_type
num_of_points
title
event_date
semester
point_type
PK type_id
name
merch
PK merch_id
item_name
price
semester
year
merch_order
PK merch_id, student_ic
FK merch_id
FK student_id
delivery_date
completed
quantity
payment_due_date
event_attendance
PK event_id, student_id
FK student_id
FK point_type
FK event_id
class_roster
PK student_id, class_id
student_id
class_id
classes
PK class_id
phone_number
semester
year
dues_paymentPlan
PK dues_id
start_date
end_date
last_amount_paid
last_payment_date
next_amount_due
next_payment_date
FK dues_id
member_dues
PK student_id, dues_id
FK student_id
FK dues_id
amount_owed
amount_paid
payment_method
notes
member
PK student_id
pawprint
status
last_name
first_name
birthday
chapter_book_number
salted_password
familyTree_roster
PK student_id
FK student_id
FK family_id
family
PK family_id
family_name
founding_date
member_status
PK status_id
status_name
member_address
PK student_id
FK student_id
street_address
city
state
zip_code
recSports
PK sport_id
sport_name
semester
start_date
end_date
recSports_teams
PK team_id
FK sport_id
team_name
team_roster
PK team_id
FK sport_id
team_name
emergencyContact
PK student_id
FK student_id
last_name
first_name
email_address
phone_number
backup_phone_number
street_address
city
state
zip_code
primary_healthcare_provider
blood_type
medical_conditions
medications
allergies
member_contact_info
PK student_id
FK student_id
email_address
phone_number
votingResults
PK student_id, roundNu
FK student_id
yes
no
abstain
potential_roster
PK student_id
pawprint
last_name
first_name
email_address
phone_number

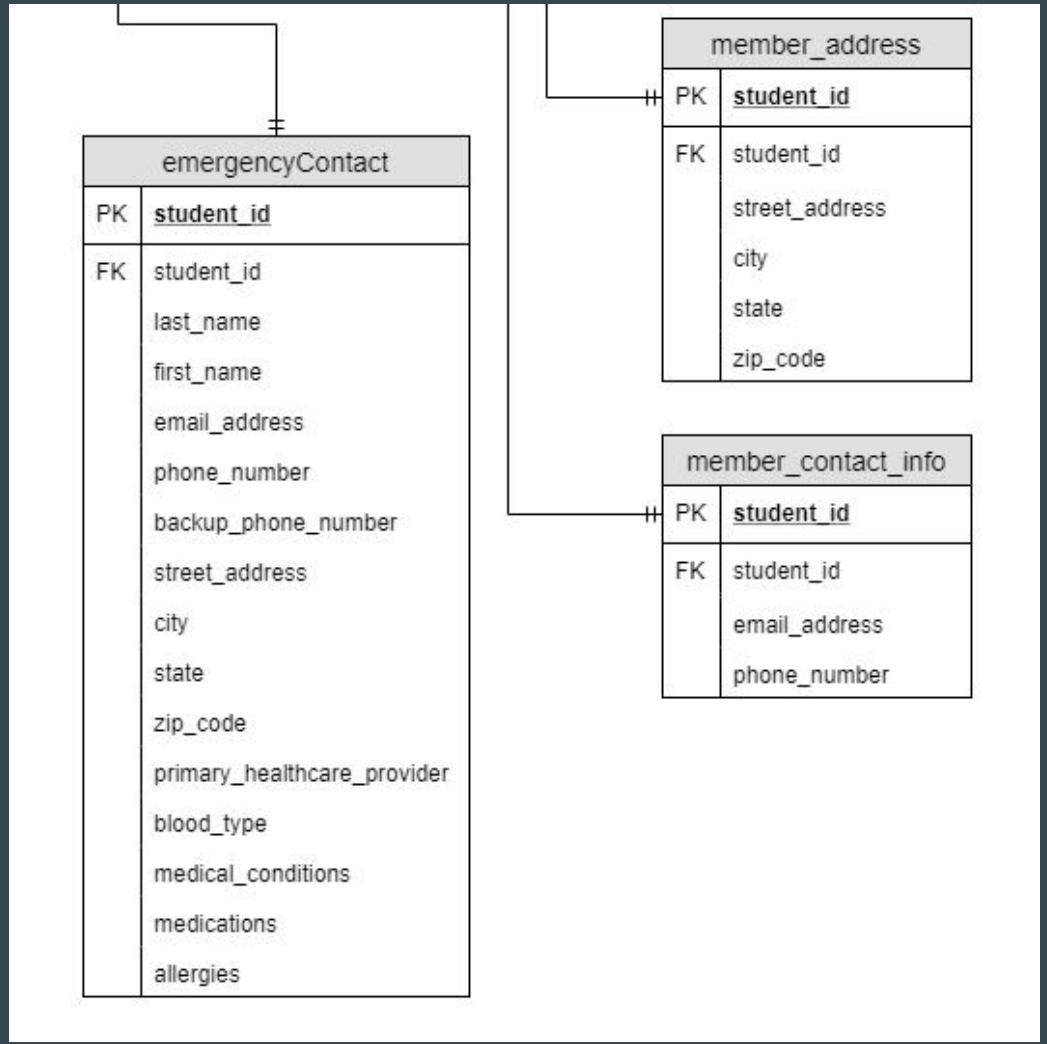
ERD - Fraternity Administration



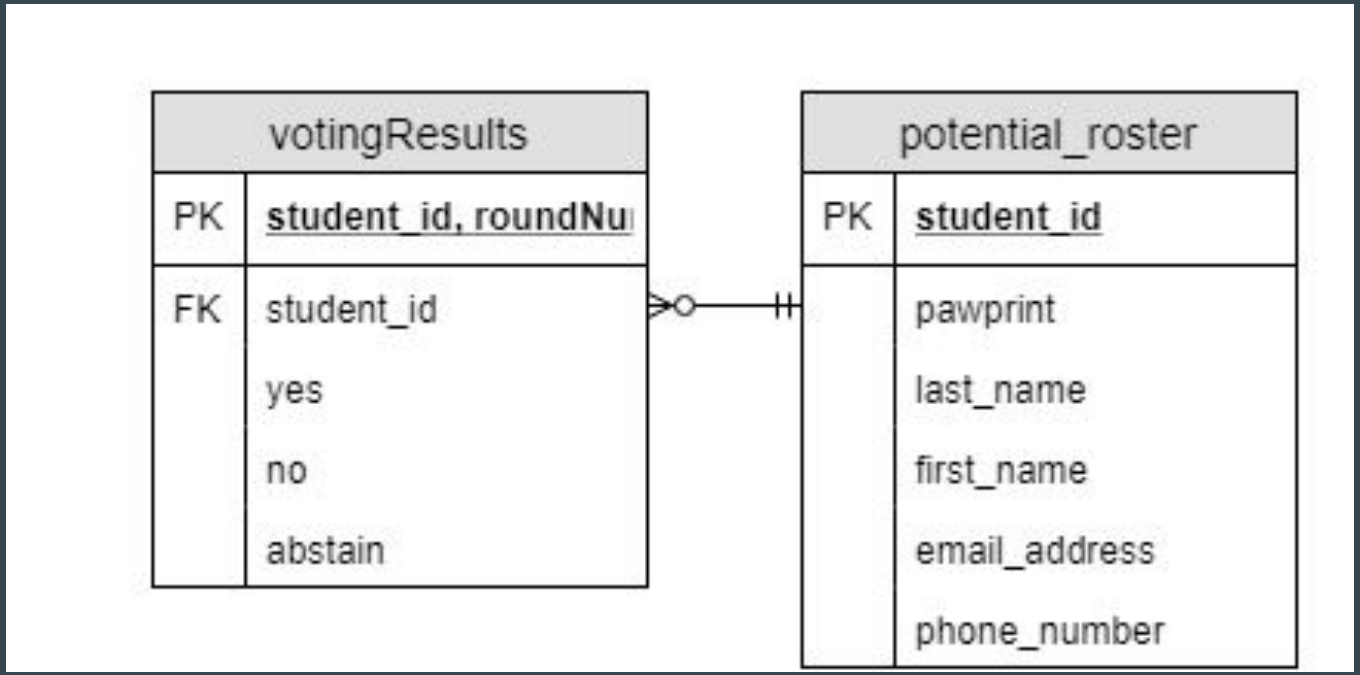
ERD - Member Information (Fraternity)



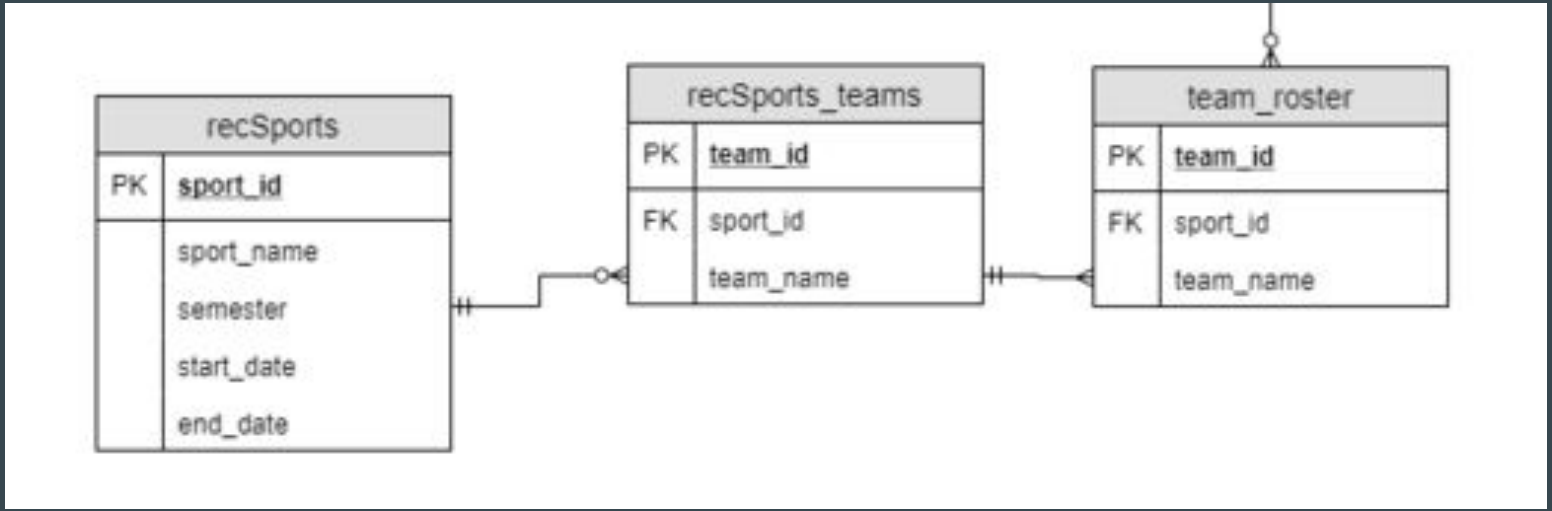
ERD - Member Information (Personal)



ERD - Voting


votingResults
PK
student_id, roundNu
FK
student_id
yes
no
abstain
potential_roster
PK
student_id
pawprint
last_name
first_name
email_address
phone_number

ERD - Sports


recSports
PK
sport_id
sport_name
semester
start_date
end_date
recSports_teams
PK
team_id
FK
sport_id
team_name
team_roster
PK
team_id
FK
sport_id
team_name

Queries

Many Queries center around ***Member Table*** (21 queries in total)

Example Queries:

- All *Member* info based on id
- All *Member* points based on id
- Grab all members and basic contact info
- Number of members that attended each event
- Display budget info for a given chairPosition
- etc...

All Potential Members that passed a specific Round (got more than 50% yes)

```
%%sql
WITH percentage AS (
    SELECT
        VR.student_id,
        VR.roundNum,
        round((VR.yes::decimal  / (VR.yes + VR.no + VR.abstain)::decimal), 2) * 100 AS yes_percentage
    FROM
        votingResults VR
)
SELECT
    VR.roundNum,
    VR.student_id,
    M.first_name,
    M.last_name,
    P.yes_percentage
FROM
    votingResults VR,
    percentage P,
    member M
WHERE
    M.student_id = VR.student_id AND
    P.student_id = VR.student_id AND
    P.roundNum = VR.roundNum AND
    /* roundNum and yes_percentage selections should be user inputed */
    VR.roundNum = 1 AND
    P.yes_percentage >= 50
```

Indexes

4 indexes

- focus on bigger, more queried tables

1. member Table *(student_id, first_name, last_name)*
    a. CREATE INDEX member_name ON member (student_id, first_name, last_name);
2. memberDues Table *(amount_owed, amount_paid)*
    a. *CREATE INDEX member_dues_amount ON memberDues (amount_owed, amount_paid);*
3. budget Table *(budget_item_id, semester, year)*
    a. *CREATE INDEX budget_index ON budget (budget_item_id, item_type, semester, year);*
4. Potential_roster Table *(student_id, first_name, last_name)*
    a. *CREATE INDEX potential_roster_name ON potential_roster (student_id, first_name, last_name);*

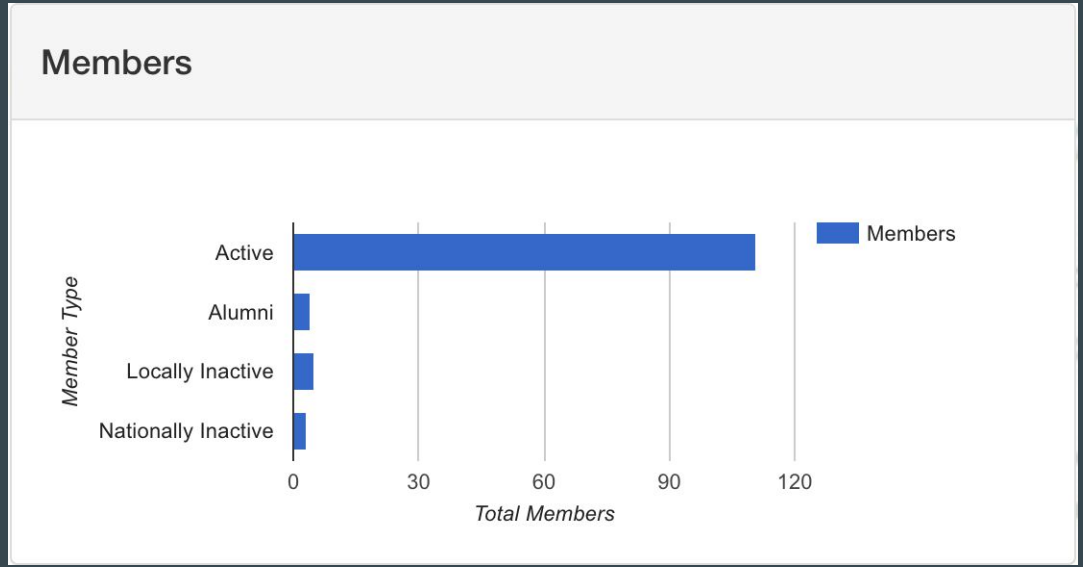
Analytics

- Grab percentages of budget items compared to total budget (pie chart)
  - Expenses in the budget
  - Revenues in the budget
- Number of Active, Inactive, Nationally Inactive, Alumni members (bar graph)
- Number of each Event DAS/fellowship/etc. (bar graph)
- Most popular merch items by number of orders (top 5) (bar graph)
- Averages of Yes, No, Abstain in votingResults (round 1, 2, 3, or all) (line/pie chart)
  - Average of each Round
  - Average of all Rounds combined

Sample Analytics

Number of Active, Inactive, etc. members

```
%%sql
SELECT
    MS.status_name,
    COUNT(M.status)
FROM
    member M,
    member_status MS
WHERE
    MS.status_id = M.status
GROUP BY
    M.status, MS.status_name
ORDER BY
    MS.status_name
```



Triggers

- When a member's Payment Plan is updated, the memberDues Table should reflect that change
  - AFTER UPDATE OF last_amount_paid ON dues_paymentPlan -> memberDues (amount_paid) updated
- When an event's point type is updated the event_attendance Table should be updated as well
  - AFTER UPDATE OF point_type ON events -> event_attendance (point_type) updated
- When a new member takes control of a position the old member in that position should be added to the archive_chairPosition Table
  - AFTER UPDATE OF student_id ON chairPosition -> INSERT new tuple INTO archive_chairPosition

Sample Trigger

```
%%sql
CREATE TRIGGER archive_chairPosition
  AFTER UPDATE OF student_id
  ON chairPosition
  FOR EACH ROW
  EXECUTE PROCEDURE position_archive();
```

Archive chairPosition Trigger

```
%%sql
CREATE OR REPLACE FUNCTION position_archive()
 RETURNS trigger AS
$$
BEGIN
    INSERT INTO archive_chairPosition(position_id,student_id,year) VALUES
        (
            OLD.position_id,
            OLD.student_id,
            date_part('year',current_date)
        );

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

Security

- Decided to use a Mandatory Access Control like scheme in the front-end
  - Didn't think necessary to implement it backend/database side, website sessions do it for us
  - Website pretty much the only way to access database
  - Database only has one login (Administrator only)
- Controlled/Utilized by chairPosition's
  - Each chair position has their own access levels
  - Members who aren't chair positions are all grouped into one security group

position_name	position_type
Vice President	EBOARD
Recording Secretary	EBOARD
Corresponding Secretary	EBOARD
Treasurer	EBOARD
Recruitment Chair	EBOARD
Initiate Advisor	EBOARD
Historian	EBOARD
Parlimentarian	EBOARD
Bro At Large	EBOARD
Fundraising Chair	EBEC
Alumni Chair	EBEC
Philanthropy Chair	EBEC
Service Chair	EBEC
DAS Chair	EBEC
Fellowship Chair	EBEC
Risk Management Chair	EBEC
Rec Sports Chair (1)	EBEC
Rec Sports Chair (2)	EBEC
PR Chair (1)	EBEC
PR Chair (2)	EBEC
Campus Liason	EBEC
President	EBOARD

DEMO

Future Work

- Add more specialized pages
  - Scope for the project was too big given time frame
  - One page with a lot of queries and analytics will branch out to singular pages for each chair position to view (22 in total)
- Implement JavaScript modals for updating/inserting data
- Revise Indexes and possibly add Views as system is used
- Support and Maintenance for initial start-up for PSP
  - Clean Database and start inputting correct production data
  - Troubleshoot any unforeseen errors
  - Fine-tune DBMS to PSP's needs (could change)

Thank You
Questions?