Marist Men’s Crew Database Project

By Raymond Mattingly

12 - 7 - 2016

Table of Contents

Executive Summary

My goal was to create a database system for the men's rowing team here at Marist. It is designed to keep detailed notes of practices on things including rowers, their boat lineups, oars in use and by whom, coaches and to which launch they are assigned, to which rowers coaches are assigned, the winning boats of each practice, the shells in which rowers and coxswains are assigned, and all of the information that can be extrapolated from the aforementioned data. The system also keeps careful track of regatta schedules, event entries within regattas, and the rowers and coxswains assigned to each event. From the database one can extract many different reports and within the database there are various views, stored procedure, and triggers that are intended to ease the process with which accurate information can be added and retrieved.

Implementation of this system can ease the process of boat selection, help coaches to coordinate practice, assist the whole team to remain organized in the face of a regatta schedule, ease the organizational nightmare of boat loading for a regatta, and help ensure the safety of those practicing.

Entity Relationship Diagram

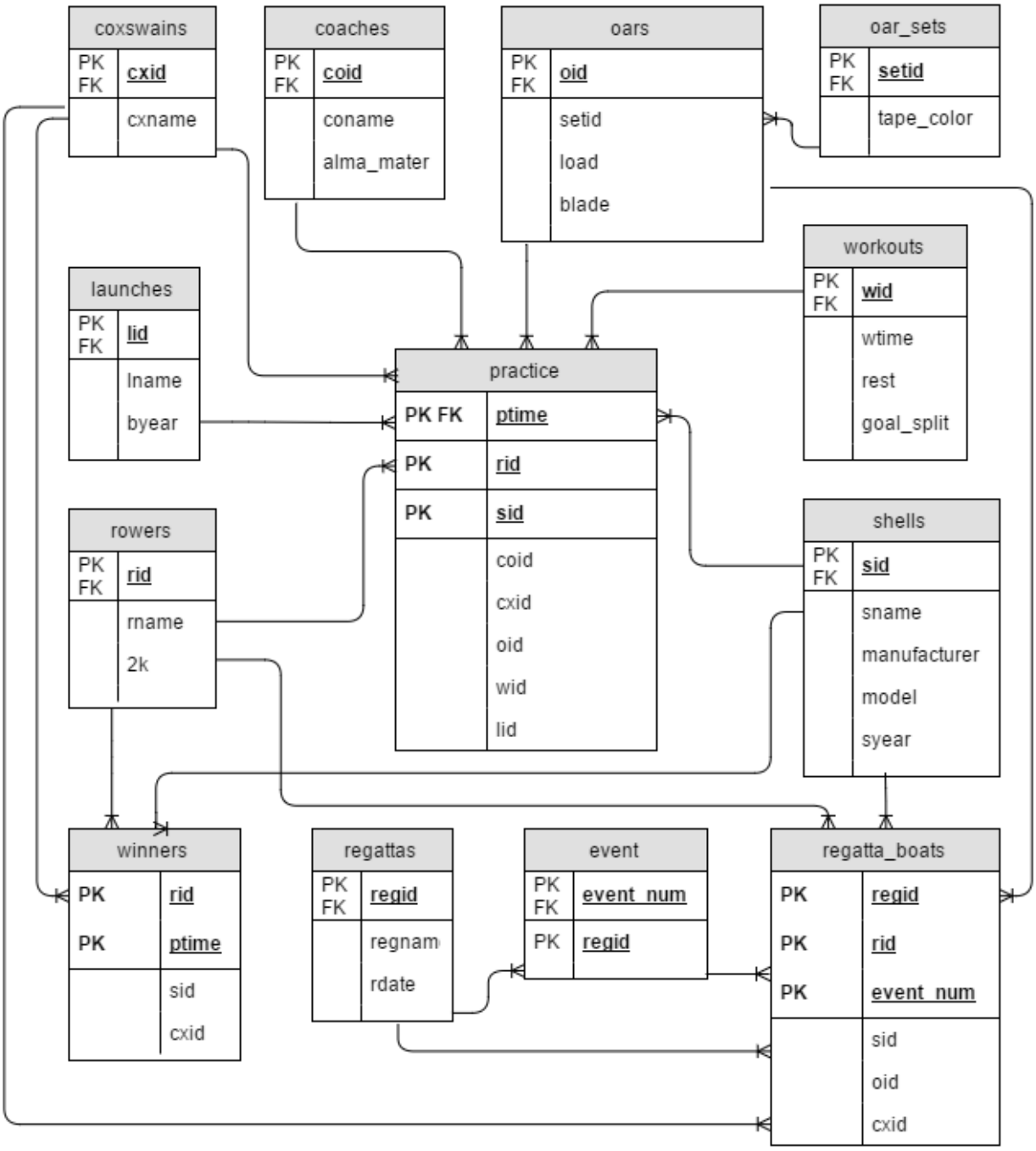

Table 1: rowers

This table works as a roster for all rowers on the team.

```
CREATE TABLE rowers
(
        rid SERIAL,
        rname varchar(255),
        twok varchar(10)
        PRIMARY KEY (rid)
);
```

Functional Dependencies:

rid→{rname, twok}

	rid integer	rname character varying(255)	twok character varying(10)	sixk character varying(10)
1	1	Raymond Mattingly	6:28	
2	2	Tadd Bindas	6:31	
3	3	Chris Carlson	6:12	
4	4	Chris Lazich	6:32	
5	5	Rami Awad	6:17	
6	6	Dan Arrato	6:25	
7	7	Joe Kohn	6:20	
8	8	Morgan Stippa	6:17	

Table 2: coaches

This table works as a roster for all coaches on the team.

```
CREATE TABLE coaches
(
        coid SERIAL,
        coname varchar(255),
        alma_mater varchar(255),
        PRIMARY KEY (coid)
);
```

Functional Dependencies:

coid→{coname, alma_mater}

	coid integer	coname character varying(255)	alma_mater character varying(255)
1	1	Campbell Woods	Oregon State
2	2	Reed Miller	UMass

Table 3: oars

This table keeps track of individual oars.

```
CREATE TABLE oars
(
        oid SERIAL,
        setid int,
        load int,
        blade varchar(255),
        PRIMARY KEY (oid)
);
```

Functional Dependencies:

oid→{setid, load, blade}

	oid integer	setid integer	load integer	blade character varying(255)
1	1	1	135	fat2
2	2	1	135	fat2
3	3	1	135	fat2
4	4	1	135	fat2
5	5	2	135	hatchet
6	6	2	135	hatchet
7	7	2	135	hatchet
8	8	2	135	hatchet

Table 4: oar_sets

This table keeps track of the sets to which each oar belongs.

```
CREATE TABLE oar_sets
(
        setid SERIAL,
        tape_color varchar(255),
        PRIMARY KEY (setid)
);
```

Functional Dependencies:

setid→{tape_color}

	setid integer	tape_color character varying(255)
1	1	green
2	2	red

Table 5: launches

This table keeps track of all coach boats owned by the team.

```
CREATE TABLE launches
(
        lid SERIAL,
        lname varchar(255),
        byear varchar(255),
        PRIMARY KEY (lid)
);
```

Functional Dependencies:

lid→{lname, byear}

	lid integer	lname character varying(255)	byear character varying(255)
1	1	Sea Snapper	2015
2	2	Tom Sanford	2014

Table 6: practice

This table ties together each practice as a set of records corresponding to each rower.

```
CREATE TABLE practice
(
        rid int,
        sid int,
        ptime timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
        coid int,
        cxid int,
        oid int,
        wid int,
        lid int,
        PRIMARY KEY (rid, sid, ptime)
);
```

Functional Dependencies:

none

	rid integer	sid integer	ptime timestamp without time zone	coid integer	cxid integer	oid integer	wid integer	lid integer
1	1	1	2016-10-15 06:30:00	1	1	1	1	1
2	2	1	2016-10-15 06:30:00	1	1	2	1	1
3	3	1	2016-10-15 06:30:00	1	1	3	1	1
4	4	1	2016-10-15 06:30:00	1	1	4	1	1
5	5	2	2016-10-15 06:30:00	2	2	5	1	1
6	6	2	2016-10-15 06:30:00	2	2	6	1	1
7	7	2	2016-10-15 06:30:00	2	2	7	1	1
8	8	2	2016-10-15 06:30:00	2	2	8	1	1
9	1	1	2016-11-15 06:15:00	2	2	1	1	2
10	2	1	2016-11-15 06:15:00	2	2	2	1	2
11	3	1	2016-11-15 06:15:00	2	2	3	1	2
12	4	1	2016-11-15 06:15:00	2	2	4	1	2
13	5	2	2016-11-15 06:15:00	1	1	5	1	1
14	6	2	2016-11-15 06:15:00	1	1	6	1	1
15	7	2	2016-11-15 06:15:00	1	1	7	1	1
16	8	2	2016-11-15 06:15:00	1	1	8	1	1

Table 7: workouts

This table keeps track of the potential workouts for each practice.

```
CREATE TABLE workouts
(
        wid SERIAL,
        wtime varchar(255),
        rest varchar(255),
        goal_split varchar(255),
        PRIMARY KEY (wid)
);
```

Functional Dependencies:

wid→{wtime, rest, goal_split}

	wid integer	wtime character varying(255)	rest character varying(255)	goal_split character varying(255)
1	1	5	5	1:35
2	2	2	2	1:27
3	3	20	4	1:50
4	4	1	1	1:24

Table 8: shells

This table keeps record of all shells owned by the team.

```
CREATE TABLE shells
(
        sid SERIAL,
        sname varchar(255),
        manufacturer varchar(255),
        model varchar(255),
        syear varchar(255),
        PRIMARY KEY (sid)
);
```

Functional Dependencies:

sid→{sname, manufacturer, model, syear}

	sid integer	sname character varying(255)	manufacturer character varying(255)	model character varying(255)	syear character varying(255)
1	1	Austin	Vespoli	E	2015
2	2	Simmons	Vespoli	D	2014

Table 9: winners

This table records the winners of each practice.

```
CREATE TABLE winners
(
        rid int,
        ptime timestamp,
        sid int,
        cxid int,
        PRIMARY KEY (rid, ptime)
);
```

Functional Dependencies:

none

	rid integer	ptime timestamp without time zone	sid integer	cxid integer
1	1	2016-10-15 06:30:00	1	1
2	2	2016-10-15 06:30:00	1	1
3	3	2016-10-15 06:30:00	1	1
4	4	2016-10-15 06:30:00	1	1
5	5	2016-11-15 06:15:00	2	2
6	6	2016-11-15 06:15:00	2	2
7	7	2016-11-15 06:15:00	2	2
8	8	2016-11-15 06:15:00	2	2

Table 10: coxswains

This table works as a roster for the team's coxswains.

```
CREATE TABLE coxswains
(
        cxid SERIAL,
        cxname varchar(255),
        PRIMARY KEY (cxid)
);
```

Functional Dependencies:

cxid→{cxname}

	cxid integer	cxname character varying(255)
1	1	Ryan Lillis
2	2	Dylan Galimi

Table 11: regattas

This table keeps track of the regatta schedule.

```
CREATE TABLE regattas
(
        regid SERIAL,
        regname varchar(255),
        rdate date,
        PRIMARY KEY (regid)
);
```

Functional Dependencies:

regid→{regname, rdate}

	regid integer	regname character varying(255)	rdate date
1	1	Head of the Charles	2016-10-22
2	2	Head of the Fish	2016-10-29

Table 12: event

This table keeps track of event entries at each regatta.

```
CREATE TABLE event
(
        event_num int,
        regid int,
        PRIMARY KEY (event_num, regid)
);
```

Functional Dependencies:

none

	event_num integer	regid integer
1	63	1
2	6	2

Table 13: regatta_boats

This table keeps track of lineups that will be attending each regatta.

```
CREATE TABLE regatta_boats
(
        regid int,
        rid int,
        event_num int,
        sid int,
        oid int,
        cxid int,
        PRIMARY KEY (regid, rid, event_num)
);
```

Functional Dependencies:

none

	regid integer	rid integer	event_num integer	sid integer	oid integer	cxid integer
1	1	1	63	1	1	1
2	1	2	63	1	2	1
3	1	3	63	1	3	1
4	1	4	63	1	4	1
5	2	5	6	2	5	2
6	2	6	6	2	6	2
7	2	7	6	2	7	2
8	2	8	6	2	8	2

View 1: Rower Wins

```
create or replace view AllWins as
select practice.ptime, rowers.rname, shells.sname
from practice
left join rowers on rowers.rid=practice.rid
left join shells on shells.sid=practice.sid
left join winners on winners.ptime=practice.ptime
        AND winners.rid=practice.rid
where
    (
                winners.rid IS NOT NULL
    );
select *
from AllWins;
```

Description:

This view consists of the rowers who have won at each practice.

Output is on the next page.

	ptime timestamp without time zone	rname character varying(255)
1	2016-10-15 06:30:00	Raymond Mattingly
2	2016-10-15 06:30:00	Tadd Bindas
3	2016-10-15 06:30:00	Chris Carlson
4	2016-10-15 06:30:00	Chris Lazich
5	2016-10-15 06:30:00	Rami Awad
6	2016-10-15 06:30:00	Dan Arrato
7	2016-10-15 06:30:00	Joe Kohn
8	2016-10-15 06:30:00	Morgan Stippa
9	2016-11-15 06:15:00	Raymond Mattingly
10	2016-11-15 06:15:00	Tadd Bindas
11	2016-11-15 06:15:00	Chris Carlson
12	2016-11-15 06:15:00	Chris Lazich
13	2016-11-15 06:15:00	Rami Awad
14	2016-11-15 06:15:00	Dan Arrato
15	2016-11-15 06:15:00	Joe Kohn
16	2016-11-15 06:15:00	Morgan Stippa

View 2: Coxswain Shells

```
create or replace view CoxswainShells as

select distinct practice.ptime, coxswains.cxname, shells.sname

from practice

left join coxswains on practice.cxid=coxswains.cxid

left join shells on shells.sid=practice.sid;

select *

from CoxswainShells;
```

Description:

This view consists of the coxswains and the shell with which they are in charge of for each practice.

	ptime timestamp without time zone	cxname character varying(255)	sname character varying(255)
1	2016-10-15 06:30:00	Dylan Galimi	Simmons
2	2016-11-15 06:15:00	Dylan Galimi	Austin
3	2016-11-15 06:15:00	Ryan Lillis	Simmons
4	2016-10-15 06:30:00	Ryan Lillis	Austin

View 3: Coach Launches

```
create or replace view CoachesLaunch as

select distinct practice.ptime, coaches.coname, launches.lname

from practice

left join launches on practice.lid=launches.lid

left join coaches on practice.coid=coaches.coid

order by practice.ptime;
```

Description:

This view consists of which coaches operated which launches at specific practices.

	ptime timestamp without time zone	coname character varying(255)	lname character varying(255)
1	2016-10-15 06:30:00	Campbell Woods	Sea Snapper
2	2016-10-15 06:30:00	Reed Miller	Sea Snapper
3	2016-11-15 06:15:00	Campbell Woods	Sea Snapper
4	2016-11-15 06:15:00	Reed Miller	Tom Sanford

Report 1: Shells on the Water

```
select distinct practice.ptime, shells.sname

from practice

left join shells on practice.sid=shells.sid

order by practice.ptime;
```

Description:

This report returns the shells on the water for each practice, for safety reasons.

	ptime timestamp without time zone	sname character varying(255)
1	2016-10-15 06:30:00	Austin
2	2016-10-15 06:30:00	Simmons
3	2016-11-15 06:15:00	Austin
4	2016-11-15 06:15:00	Simmons

Report 2: What Shells to Bring to Regattas

```
select distinct regattas.rdate, regattas.regname, shells.sname
from regattas, shells
left join regatta_boats on regatta_boats.sid=shells.sid
order by regattas.rdate
```

Description:

This report returns the shells that one must bring to each regatta, for organizational purposes.

	rdate date	regname character varying(255)	sname character varying(255)
1	2016-10-22	Head of the Charles	Austin
2	2016-10-22	Head of the Charles	Simmons
3	2016-10-29	Head of the Fish	Austin
4	2016-10-29	Head of the Fish	Simmons

Report 3: What Rowers are on the Water

```
select practice.ptime, rowers.rname

from practice

left join rowers on practice.rid=rowers.rid

order by practice.ptime;
```

Description:

This report returns the rowers on the water at each practice, for safety reasons.

	ptime timestamp without time zone	rname character varying(255)
1	2016-10-15 06:30:00	Raymond Mattingly
2	2016-10-15 06:30:00	Tadd Bindas
3	2016-10-15 06:30:00	Chris Carlson
4	2016-10-15 06:30:00	Chris Lazich
5	2016-10-15 06:30:00	Rami Awad
6	2016-10-15 06:30:00	Dan Arrato
7	2016-10-15 06:30:00	Joe Kohn
8	2016-10-15 06:30:00	Morgan Stippa
9	2016-11-15 06:15:00	Raymond Mattingly
10	2016-11-15 06:15:00	Tadd Bindas
11	2016-11-15 06:15:00	Chris Carlson
12	2016-11-15 06:15:00	Chris Lazich
13	2016-11-15 06:15:00	Rami Awad
14	2016-11-15 06:15:00	Dan Arrato
15	2016-11-15 06:15:00	Joe Kohn
16	2016-11-15 06:15:00	Morgan Stippa

Stored Procedure 1: Wins2(name)

```
create or replace function Wins2(varchar, REFCURSOR) returns refcursor as
$$
declare
        eName varchar           := $1;
        resSet REFCURSOR        := $2;
begin
        open resSet for
                select ptime, rname, sname
                from AllWins
                where rname=eName;
        return resSet;
end;
$$
language plpgsql;
select Wins2('Raymond Mattingly', 'results2');
Fetch all from results2;
```

Description:

This function returns the wins for a specific rower.

	ptime timestamp without time zone	rname character varying(255)	sname character varying(255)
1	2016-10-15 06:30:00	Raymond Mattingly	Austin

Stored Procedure 2: CoachLaunchesFxn(name)

```
create or replace function CoachLaunchesFxn(varchar, REFCURSOR) returns refcursor
as
$$
declare
        cName varchar           := $1;
        resSet REFCURSOR        := $2;
begin
        open resSet for
                select ptime, coname, lname
                from CoachesLaunch
                where coname=cName;
        return resSet;
end;
$$
language plpgsql;
select CoachLaunchesFxn('Reed Miller', 'clresults');
Fetch all from clresults;
```

Description:

This function returns the launches operated by a specific coach.

	ptime timestamp without time zone	coname character varying(255)	lname character varying(255)
1	2016-10-15 06:30:00	Reed Miller	Sea Snapper
2	2016-11-15 06:15:00	Reed Miller	Tom Sanford

Stored Procedure 3: CoxswainShellsFxn(name)

```
create or replace function CoxswainShellsFxn(varchar, REFCURSOR) returns refcursor
as
$$
declare
        cName2 varchar          := $1;
        resSet REFCURSOR        := $2;
begin
        open resSet for
                select ptime, cxname, sname
                from CoxswainShells
                where cxname=cName2;
        return resSet;
end;
$$
language plpgsql;
```

Description:

This function returns the shells operated by specific coxswains.

	ptime timestamp without time zone	cxname character varying(255)	sname character varying(255)
1	2016-11-15 06:15:00	Ryan Lillis	Simmons
2	2016-10-15 06:30:00	Ryan Lillis	Austin

Trigger 1: valid_rower

```
create or replace function ValidateRowerFxn() returns trigger as
$$
begin
        if exists
        (
                select 1
                from rowers
                where rid=new.rid
        ) then
        return null;
        end if;
return new;
end
$$
language plpgsql;
CREATE TRIGGER valid_rower
BEFORE INSERT ON rowers
FOR EACH ROW
EXECUTE PROCEDURE ValidateRowerFxn();
```

Description:

This trigger and function prevent duplicate entries within the rowers table.

```
INSERT INTO rowers
        VALUES ('1', 'Raymond Mattingly2', '6:28');
```

Output pane

Data Output | Explain | **Messages** | History

```
Query returned successfully: 0 rows affected, 21 msec execution time.
```

Trigger 2: valid_coach

```
create or replace function ValidateCoachFxn() returns trigger as
$$
begin
	if exists
	(
		select 1
		from coaches
		where coid=new.coid
	) then
	return null;
	end if;
return new;
end
$$
language plpgsql;
CREATE TRIGGER valid_coach
BEFORE INSERT ON coaches
FOR EACH ROW
EXECUTE PROCEDURE ValidateCoachFxn();
```

Description:

This trigger and function prevent duplicates within the coaches table.

```
INSERT INTO coaches
       VALUES ('1', 'CampbellWoods2', 'Go Ducks, Baby');
```

Output pane

Data Output | Explain | **Messages** | History

```
Query returned successfully: 0 rows affected, 24 msec execution time.
```

Security

```
CREATE ROLE Admin

CREATE ROLE Coach

--admin privileges below

GRANT SELECT, INSERT, UPDATE, DELETE ON coxswains TO Admin;

GRANT SELECT, INSERT, UPDATE, DELETE ON coaches TO Admin; GRANT

SELECT, INSERT, UPDATE, DELETE ON oars TO Admin; GRANT SELECT,

INSERT, UPDATE, DELETE ON oar_sets TO Admin; GRANT SELECT,

INSERT, UPDATE, DELETE ON launches TO Admin; GRANT SELECT,

INSERT, UPDATE, DELETE ON practice TO Admin; GRANT SELECT, INSERT,

UPDATE, DELETE ON workouts TO Admin; GRANT SELECT, INSERT,

UPDATE, DELETE ON rowers TO Admin; GRANT SELECT, INSERT, UPDATE,

DELETE ON shells TO Admin; GRANT SELECT, INSERT, UPDATE, DELETE

ON winners TO Admin; GRANT SELECT, INSERT, UPDATE, DELETE ON

regattas TO Admin; GRANT SELECT, INSERT, UPDATE, DELETE ON event

TO Admin; GRANT SELECT, INSERT, UPDATE, DELETE ON regatta_boats TO

Admin;

--revoke privileges from Coach

REVOKE ALL PRIVILEGES ON coxswains FROM Coach; REVOKE ALL

PRIVILEGES ON coaches FROM Coach; REVOKE ALL PRIVILEGES ON oars

FROM Coach; REVOKE ALL PRIVILEGES ON oar_sets FROM Coach;

REVOKE ALL PRIVILEGES ON launches FROM Coach; REVOKE ALL
```

```
PRIVILEGES ON practice FROM Coach; REVOKE ALL PRIVILEGES ON
workouts FROM Coach; REVOKE ALL PRIVILEGES ON rowers FROM Coach;
REVOKE ALL PRIVILEGES ON shells FROM Coach; REVOKE ALL
PRIVILEGES ON winners FROM Coach; REVOKE ALL PRIVILEGES ON
regattas FROM Coach; REVOKE ALL PRIVILEGES ON event FROM Coach;
REVOKE ALL PRIVILEGES ON regatta_boats FROM Coach;
--grant Coach privileges
GRANT SELECT, INSERT, UPDATE, DELETE ON practice TO Coach; GRANT
SELECT, INSERT, UPDATE, DELETE ON regattas TO Coach; GRANT
SELECT, INSERT, UPDATE, DELETE ON event TO Coach; GRANT SELECT,
INSERT, UPDATE, DELETE ON regatta_boats TO Coach;
```

Implementation Notes

- Use the premade or create new views when extrapolating information from the practice table -- depending on the desired information it can save many lines.
- The admin user will need to be used to update roster and equipment information; the coach user can be used to update practice records, the list of intended regattas, and regatta lineups.

Known Problems

- Due to the nature of this database system and its method of tying many strong entities into a weak entity via primary key (as is done in the practice table), one may need to incorporate many tedious joins in order to extrapolate worthwhile information from the data.
- Often times the tedious joins needed to extrapolate information can result in quickly made reports containing many duplicates or useless information.
- I am not really happy with the way that the winners are recorded. It currently records an entry for each winning rower, though by doing that it records an unnecessarily bulky load of information. A fix for this problem is discussed on the next page.

Future Enhancements

- While keeping track of practice results in important, regatta results are ultimately what matters and this database system currently does not support the tracking of regatta results.
- One could add a functionality that allows for keeping track of which shells are in need of or currently undergoing repair. If one were being truly cut throat, they could also create a way of tracking which coxswains most frequently damage their shell's hull.
- I could fix the issue with the winners being recorded in an unnecessarily bulky and almost duplicative way by recording winning shell ID for each practice and creating a view while pulls the people from each practice that were a part of the winning shell.