Channel Islands / COMPUTER SCIENCE PROGRAM

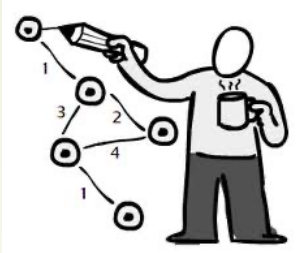
Middle School Citizenship Tracker

By: Eric Gentry

COMP520 - Advanced Database Systems

Spring 2017

Project Overview



- Keep track of citizenship behavior of middle school students
- Teachers are the main users
- Daily tracking of behaviors:
  - Participating (positive)
  - Helpful (positive)
  - Disruptive (negative)
  - Not Attentive (negative)
- Reports and emailing to keep teacher informed

Problem to solve

- Teachers do not have a simple way of tracking middle school citizenship points
- Each student starts with 100 citizenship points per semester and points and are adjusted based on the tracking tool feedback
- Teachers need a data-centric solution to keep track of and show (if necessary)

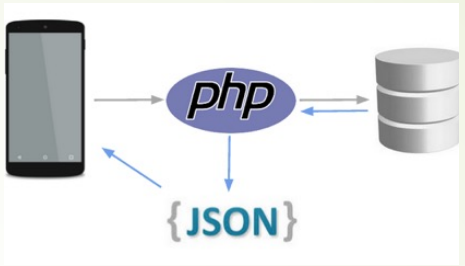
Data source

- Completely made up, but based on:
- Medea Creek Middle School
- Science, 7th grade
- Math 7th grade
- Physical Education 7th grade
- Student's names and photos are mocked up

Business rules

- Each school year must be unique
- Each class must be unique
- Students must have a first name, last name, and a guardian
- Teachers must have a first name, last name, and email
- Behaviors are kept track of on a daily basis and always start at 0
- Seating Charts cannot have more than 10 seats across and 10 seats back for students

Technologies



- mysql for the database (through cikeys)
- PHP web service
  - custom RESTful API
  - accessed via and stored on cikeys
  - direct access to mysql database
  - JSON results
- iOS designed for the iPad

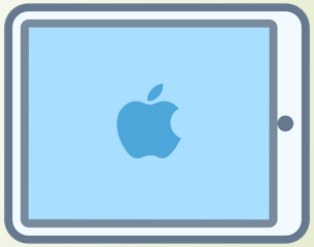
CI keys

MySQL

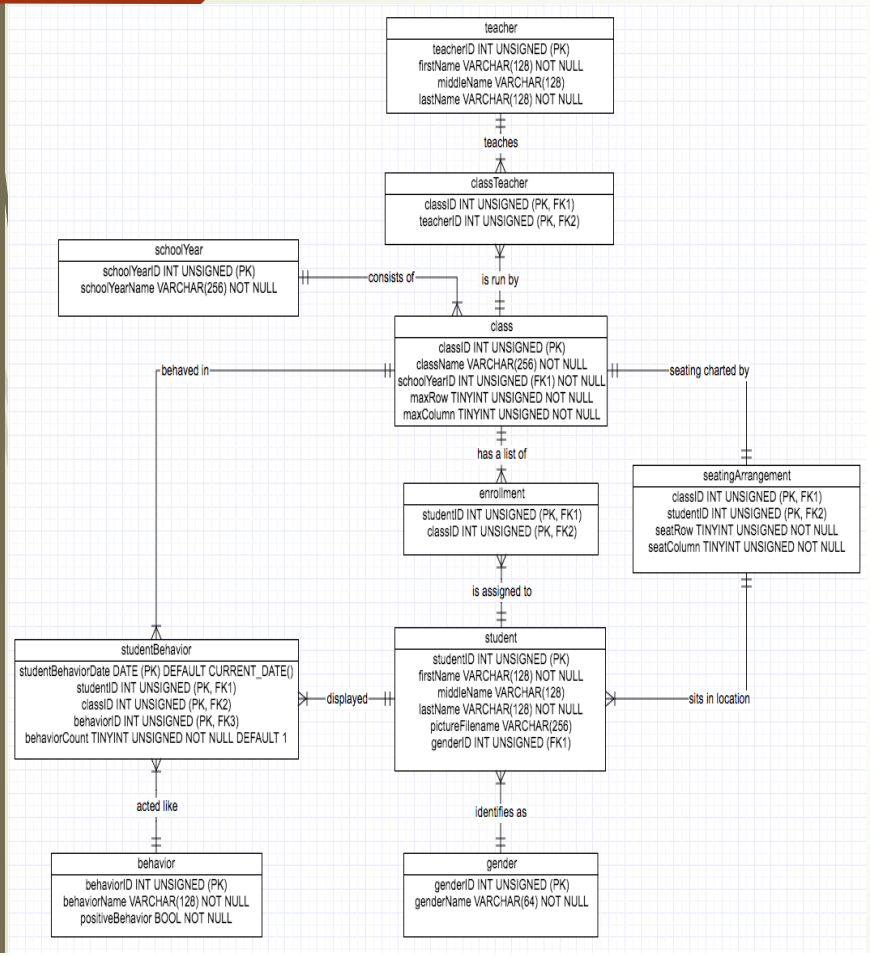
php

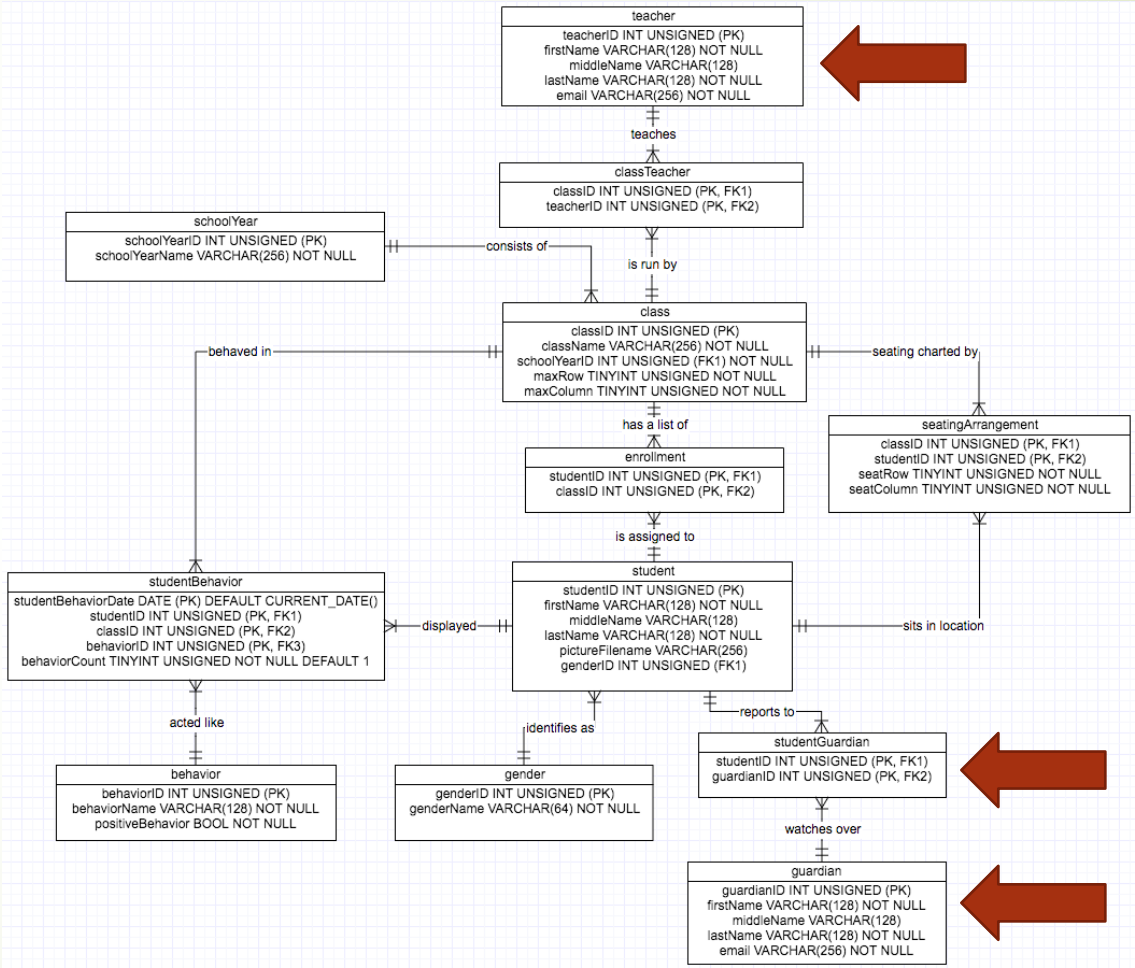
JSON

iOS



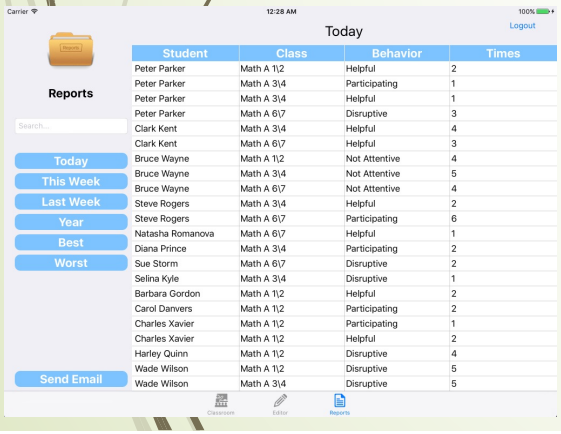
ERDs (old and new)

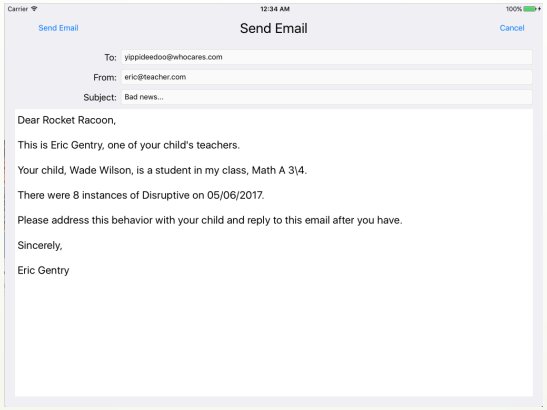




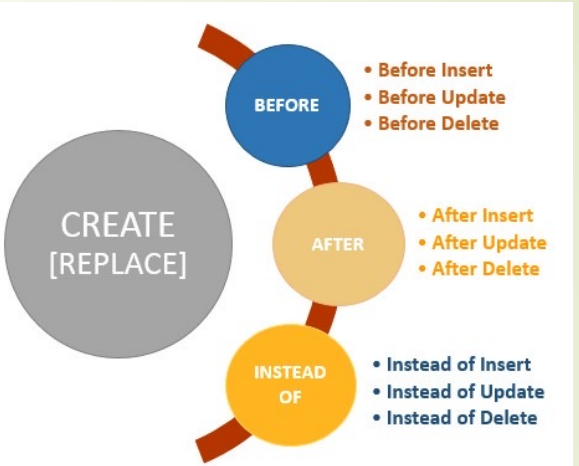
User Interface

- Teacher classroom
- Editor for modifications
  - Teacher, Student, School Year, Class, Classroom Editors
- Report viewer
  - Email sender





Triggers




```
DELIMITER //
CREATE TRIGGER `enrollment_AFTER_INSERT` AFTER INSERT ON `enrollment` FOR EACH ROW
   BEGIN
      -- variable declaration section
      DECLARE biggestRow INT;
      DECLARE biggestColumn INT;
      DECLARE currentRow INT;
      DECLARE currentColumn INT;
      DECLARE nextRow INT;
      DECLARE nextColumn INT;

      -- put the new student into a valid seat
      IF (NOT EXISTS (SELECT MAX(seatRow) FROM seatingArrangement WHERE classID=NEW.classID GROUP BY classID)) THEN
         -- this is for when this is the first student added to a class
         INSERT INTO seatingArrangement VALUES (NEW.classID, new.studentID, 0, 0);
      ELSE
         -- find the next available seat
         SET biggestRow = (SELECT maxRow FROM class WHERE classID=NEW.classID) -1;  -- zero based index
         SET biggestColumn = (SELECT maxColumn FROM class WHERE classID=NEW.classID) -1;  -- zero based index
         SET currentRow = (SELECT MAX(seatRow) FROM seatingArrangement WHERE classID=NEW.classID);
         SET currentColumn = (SELECT MAX(seatColumn) FROM seatingArrangement WHERE classID=NEW.classID AND seatRow=currentRow GROUP BY classID);
         IF (currentColumn = biggestColumn) THEN
            SET nextRow = currentRow + 1;
            SET nextColumn = 0;
         ELSE
            SET nextRow = currentRow;
            SET nextColumn = currentColumn + 1;
         END IF;
         -- now put the student in the next available seat
         INSERT INTO seatingArrangement VALUES (NEW.classID, new.studentID, nextRow, nextColumn);
      END IF;
   END;
//
DELIMITER ;
```

Stored Procedures

```
DROP PROCEDURE IF EXISTS decreaseBehaviorCount;
DELIMITER //
   CREATE PROCEDURE decreaseBehaviorCount(IN theStudentID INT UNSIGNED, IN theClassID INT UNSIGNED, IN theBehaviorID INT UNSIGNED, IN theBehaviorCount INT UNSIGNED)
   BEGIN
      IF (theBehaviorID < 1) THEN
         DELETE FROM studentBehavior
         WHERE studentBehaviorDate=CURRENT_DATE()
         AND studentID=theStudentID
         AND classID=theClassID
         AND behaviorID=theBehaviorID;
      ELSE
         UPDATE studentBehavior
         SET behaviorCount=theBehaviorCount
         WHERE studentBehaviorDate=CURRENT_DATE()
         AND studentID=theStudentID
         AND classID=theClassID
         AND behaviorID=theBehaviorID;
      END IF;
   END//
DELIMITER ;
```

```
DROP PROCEDURE IF EXISTS deleteStudent;
DELIMITER //
   CREATE PROCEDURE deleteStudent(IN theStudentID INT UNSIGNED)
   BEGIN
      -- need setup a "handler" that will rollback any errors in the procedure
      DECLARE exit handler for sqlexception,sqlwarning
      BEGIN
         ROLLBACK;
      END;

      START TRANSACTION;
         -- must delete the places where there are foreign keys first!
         DELETE FROM enrollment WHERE studentID=theStudentID AND classID>0;
         DELETE FROM studentBehavior WHERE studentID=theStudentID AND studentBehaviorDate>0 AND classID>0 AND behaviorID>0;
         DELETE FROM guardian WHERE guardianID IN (SELECT guardianID FROM studentGuardian WHERE studentID=theStudentID);
         DELETE FROM studentGuardian WHERE studentID=theStudentID AND guardianID>0;
         DELETE FROM seatingArrangement WHERE studentID=theStudentID AND classID>0;
         DELETE FROM student WHERE studentID=theStudentID;
      COMMIT;
   END//
DELIMITER ;
```

Code

```
- (IBAction)SaveButtonPush:(UIButton *)sender {

    NSString *whereClause = [NSString stringWithFormat:@"studentID>0 AND (classID=%@", classroomClassIDs[0]];
    for (int i = 1; i < classroomClassIDs.count; i++) {
        whereClause = [NSString stringWithFormat:@"%@ OR classID=%@", whereClause, classroomClassIDs[i]];
    }
    whereClause = [NSString stringWithFormat:@"%@)", whereClause];

    // delete everything
    NSString *sql = [NSString stringWithFormat:@"%@%@%@%@",
                     @"D.W.",
                     @"&0=seatingArrangement",
                     @"&1=", whereClause];
    [db GetDataResults:@"MSCT.php?Z=" :sql];

    NSString *values = @"";
    for (int i = 0; i < classroomStudentInfo.count; i++) {
        NSString *classID = classroomClassIDs[i];
        NSInteger maxCol = [classroomMaxColumn[i] integerValue];
        NSMutableArray *studentsInfo = classroomStudentInfo[classID];
        for (int j = 0; j < studentsInfo.count; j++) {
            NSInteger studentRow = j/maxCol;
            NSInteger studentColumn = j%maxCol;
            NSMutableDictionary* studentInfo = studentsInfo[j];
            values = [NSString stringWithFormat:@"%@(%@,%@,%ld,%ld),",
                      values,
                      classID,
                      studentInfo[@"studentID"],
                      (long)studentRow,
                      (long)studentColumn];
        }
    }
    values = [values substringToIndex:[values length] - 1];  // remove last comma

    // record everything
    sql = [NSString stringWithFormat:@"%@%@%@%@",
           @"I.V.",
           @"&0=seatingArrangement",
           @"&1=", values];
    [db GetDataResults:@"MSCT.php?Z=" :sql];

    [self dismissViewControllerAnimated:YES completion:nil];
}
```

Demo time!