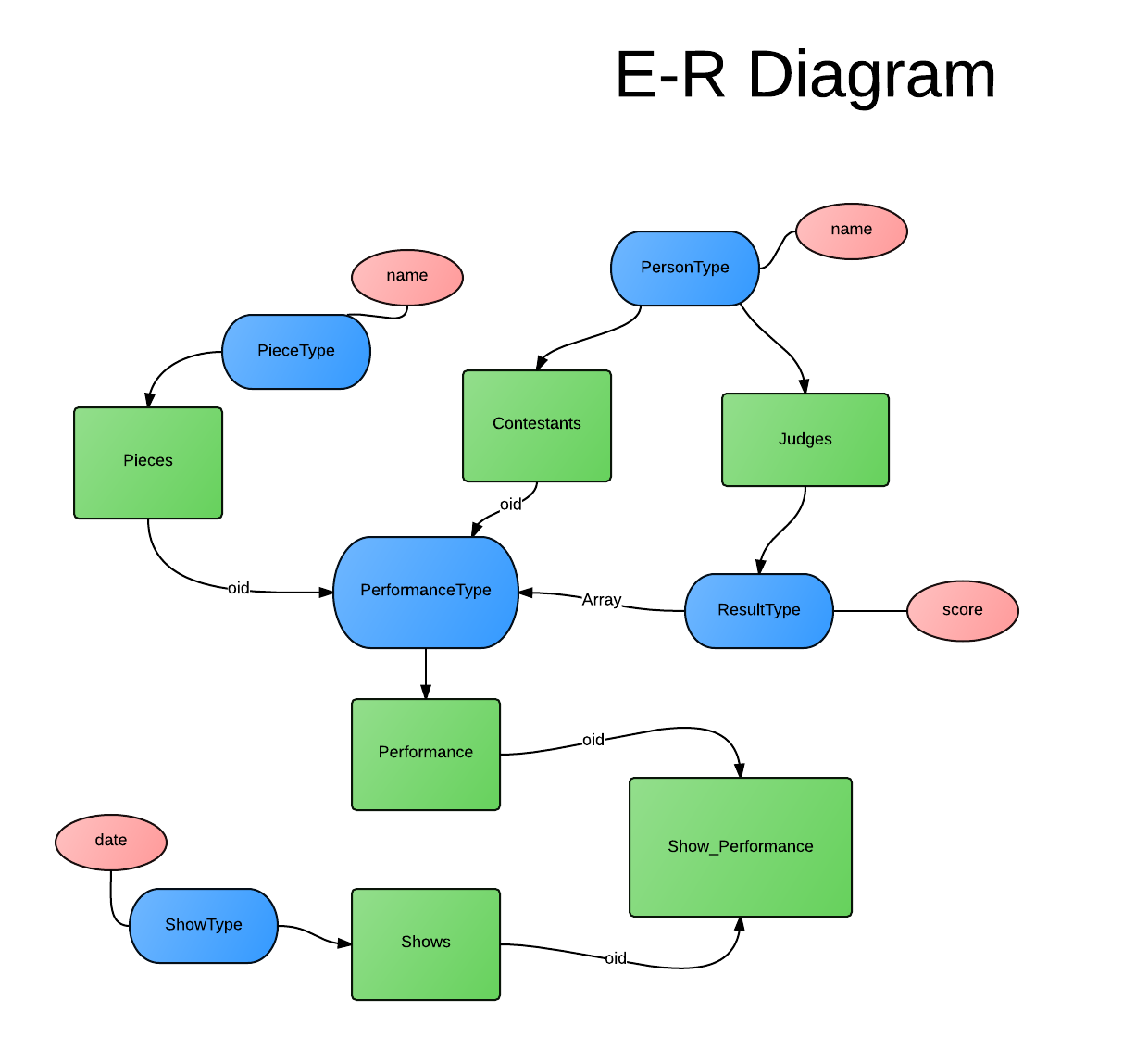
CSE 532 Project 2 Report

109892492 Hao-Tsung Yang

109247869 Kin Sum Liu

We, pledge our honor that all parts of this project were done by us alone and without collaboration with anybody else.

# E-R design



Pinks are basic types, blues are UDTs, greens are tables.

# Database Scheme

In this projects, we have 5 UDTs:

1. Person type: Save the person name attribute only.
2. Piece type: Save the piece name attribute only.
3. Result type: Have the score attribute and judge person.
4. Performance type: Have the oid of Contestant table, oid of Piece table and the arrays that save multiple result type objects.
5. Show types: Have the date attribute.

And 6 typed tables:

1. Contestants: Contains Person Type w/ unique name. Save all the contestants.
2. Judges: Contains Person Type w/ unique name. Save all the judges.
3. Pieces: Contains Piece Type w/ unique name. Save all the pieces .
4. Shows: Save the date attribute only. Remember the date when the show start.
5. Performances: Contains Performance Type w/ one oid in contestant, one oid in pieces, and several results that judges judge for the performances.
6. Show\_performace: Contains one oid in Shows and one oid in performances. Present when the show start and several performances at that show.

# Integrity Constraints

Consider the tables we have:

1. Contestants: Unique Key (name). Check length(name) > 0.
2. Judges: Unique Key(name). Check length(name) > 0.
3. Pieces: Unique Key(name). Check length(name) > 0.
4. Performances: Foreign Key (Contestant) REF (Contestants).

Foreign Key (Piece) REF (Pieces).

1. Show\_performace: Foreign Key (show) REF(Shows)

Foreign Key (performance) REF(Performances).

# Database Schema

CREATE TYPE PersonType AS (

name text

);

CREATE TYPE PieceType AS (

name text

);

CREATE TYPE ResultType AS (

judge text,

score integer

);

CREATE TYPE PerformanceType AS (

Contestant oid,

Piece oid,

results ResultType[]

);

CREATE TYPE ShowType AS (

showdate date

);

CREATE TABLE Contestants OF PersonType (

UNIQUE(name),

PRIMARY KEY (oid),

name WITH OPTIONS CHECK (char\_length(name) > 0)

) WITH (OIDS);

CREATE TABLE Judges OF PersonType (

UNIQUE(name),

PRIMARY KEY (oid),

name WITH OPTIONS CHECK (char\_length(name) > 0)

) WITH (OIDS);

CREATE TABLE Pieces OF PieceType(

UNIQUE(name),

PRIMARY KEY (oid),

name WITH OPTIONS CHECK (char\_length(name) > 0)

) WITH (OIDS);

CREATE TABLE Performances OF PerformanceType (

Contestant WITH OPTIONS REFERENCES Contestants(oid),

Piece WITH OPTIONS REFERENCES Pieces(oid),

PRIMARY KEY (oid)

) WITH (OIDS);

CREATE TABLE Shows OF ShowType(

UNIQUE(showdate),

PRIMARY KEY (oid)

) WITH (OIDS);

CREATE TABLE Show\_Performances (

show oid REFERENCES Shows(oid),

performance oid REFERENCES Performances(oid)

);

CREATE VIEW Shows3Judges AS //find shows with 3 judges (for Q3)

SELECT oid, showdate // Select the show’s oid that

FROM Shows S

WHERE EXISTS( // exists some of the contestants

SELECT P.contestant

FROM Show\_Performances SP, Performances P

WHERE S.oid = SP.show AND

SP.performance = P.oid AND

array\_length(P.results, 1) > 2 // that have more than 2 results (means there are at least 3 judges)

);

CREATE VIEW Direct\_Coaudition AS ( //this view can be used in Q1 and Q5

SELECT C1.name AS name1, C2.name AS name2

FROM Contestants C1, Contestants C2,

Show\_Performances SP1, Show\_Performances SP2,

Performances P1, Performances P2

WHERE SP1.show = SP2.show AND // select the same show of different performances

SP1.performance = P1.oid AND

SP2.performance = P2.oid AND

P1.piece = P2.piece AND //with same piece

P1.contestant = C1.oid AND

P2.contestant = C2.oid AND

C1.name != C2.name AND // but different contestants

P1.results && P2.results // w/ same result

);

CREATE RECURSIVE VIEW Coaudition (name1, name2) AS ( //build a recursive view

SELECT \*

FROM Direct\_Coaudition //select all the stuff from Direct\_Coaudition first.

UNION ALL

SELECT C1.name1,C2.name2

FROM Coaudition C1, Direct\_Coaudition C2

WHERE C1.name2 = C2.name1 AND C1.name1 != C2.name2 AND NOT EXISTS( //select C1 and C2 if C1.name2 is same as C2.name1

SELECT C.name1, C.name2

FROM Direct\_Coaudition C

WHERE C.name1 = C1.name1 and C.name2 = C2.name2

)

);

# SQL queries

**-- Query 1:**

Find all pairs of contestants who happened to audition the same piece during the same show and got the same score from at least one judge

SELECT \* FROM Direct\_Coaudition WHERE name1 < name2; //show the tuples in Direct\_Coaudition but no duplicated data

**-- Query 2:**

Find all pairs of contestants who happened to audition the same piece (in possibly different shows) and got the same average score for that piece.

SELECT C1.name AS name1, C2.name AS name2

FROM Contestants C1, Contestants C2,

Performances P1, Performances P2

WHERE P1.piece = P2.piece AND //2 contestants w/ same piece

P1.contestant = C1.oid AND

P2.contestant = C2.oid AND

C1.name < C2.name AND // but different contestants

( SELECT avg(r.score) FROM unnest(P1.results) r) = ( SELECT avg(r.score) FROM unnest(P2.results) r) // have same average scores.

;

**-- Query 3:**

Find all pairs of contestants who auditioned the same piece in (possibly different) shows that had at least 3 judges and the two contestants got the same highest score.

SELECT C1.name AS name1, C2.name AS name2

FROM Contestants C1, Contestants C2, //select 2 contestants

Shows3Judges S1, Shows3Judges S2,

Show\_Performances SP1, Show\_Performances SP2,

Performances P1, Performances P2

WHERE S1.oid = SP1.show AND // that perform in the show of show3judge

S2.oid = SP2.show AND

SP1.performance = P1.oid AND

SP2.performance = P2.oid AND

P1.piece = P2.piece AND //with same piece

P1.contestant = C1.oid AND

P2.contestant = C2.oid AND

C1.name < C2.name AND

( SELECT max(r.score) FROM unnest(P1.results) r) = ( SELECT max(r.score) FROM unnest(P2.results) r)

; // and same max score

**-- Query 4:**

Find all pairs of contestants such that the first contestants has performed all the pieces of the second contestant (possibly in different shows) C1 dominates C2

SELECT C1.name AS name1, C2.name AS name2

FROM Contestants C1, Contestants C2 //select contestants C1 and C2 such that

WHERE

EXISTS( //exist a piece performed by C1

(

SELECT piece

FROM Performances P

WHERE P.contestant = C1.oid

)

EXCEPT //but not performed by C2.

(

SELECT piece

FROM Performances P

WHERE P.contestant = C2.oid

)

) AND NOT EXISTS( // and not exist

(

SELECT piece // a piece performed by C2

FROM Performances P

WHERE P.contestant = C2.oid

)

EXCEPT //but not performed by C1

(

SELECT piece

FROM Performances P

WHERE P.contestant = C1.oid

)

)

;

**-- Query 5:**

Find all chained co-auditions that

-- X and Y (directly) co-auditioned iff they both performed the same piece in the same show and got the same score from at least one (same) judge.

SELECT \* FROM Coaudition C WHERE C.name1<C.name2;