CBS2 DATABASE

A Database Design Proposal

By Mitchell Xanders

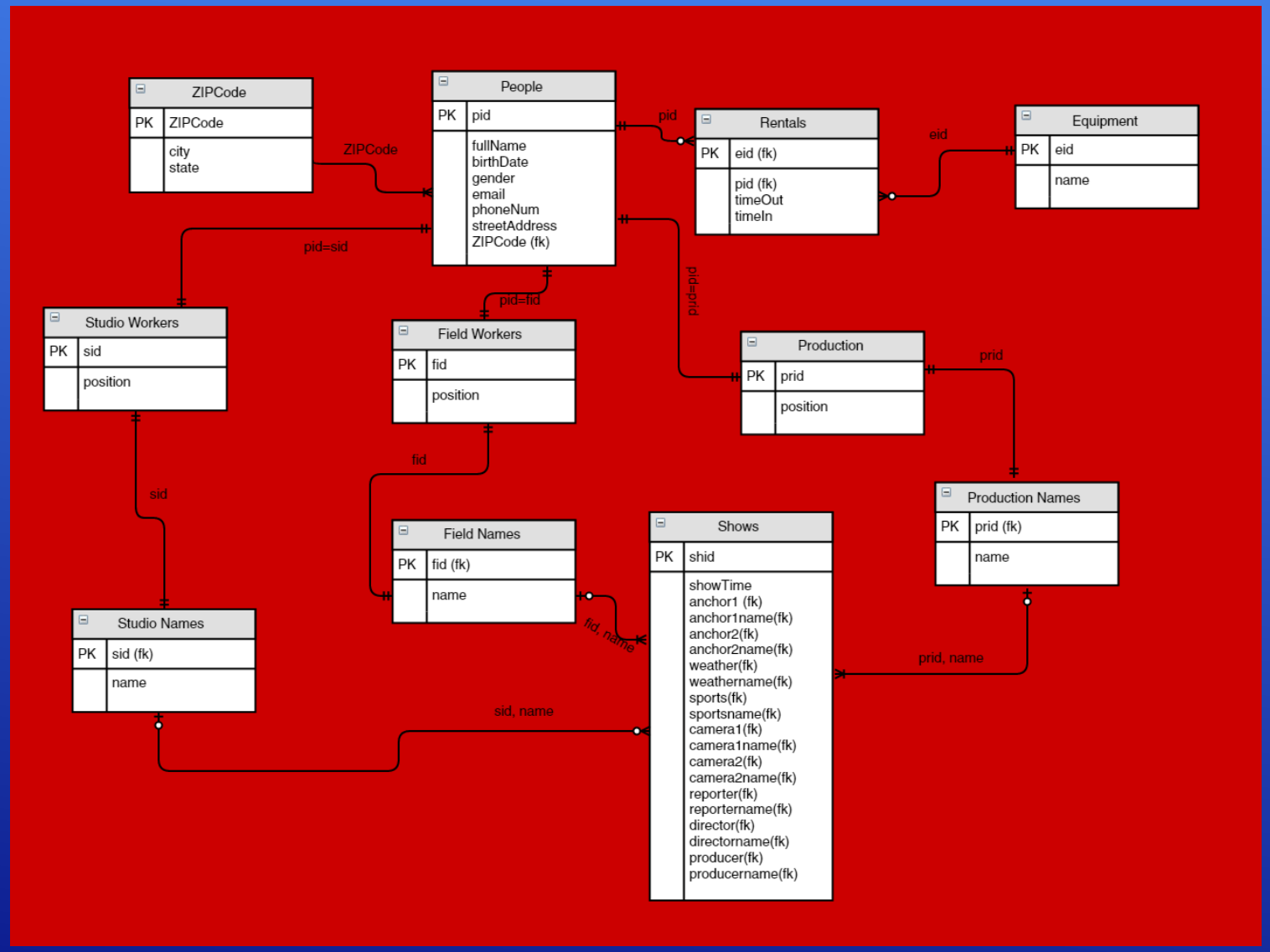
TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document serves as a basis for outlining the relationships between employees, shifts, jobs, and equipment. It's primary aim is to keep every employee and every item accounted for while being easy to update on a day-to-day basis. With further funding and further development, the strongest database can be attained to keep the news station operating smoothly.

This document provides a basic overview of the database and its specific relations, along with samples of each relation with data. It will also explore stored procedures and triggers meant to increase reliability and efficiency in the database.

In the end, this plan hopes to lead to the development of a strong database for the betterment of the organization.



ENTITY-RELATIONSHIP DIAGRAM

```
--Create the ZIPCode table
CREATE TABLE IF NOT EXISTS ZIPCode (
        ZIPCode         integer         not null unique,
        city            text            not null,
        state           char(2)         not null,
        PRIMARY KEY(ZIPCode)
);
```

Functional Dependencies:
ZIPCode → city,state

zipcode integer	city text	state character(2)
12601	Poughkeepsie	NY
92253	La Quinta	CA
92260	Palm Desert	CA
92262	Palm Springs	CA
92276	Thousand Palms	CA
90052	Los Angeles	CA
94114	San Francisco	CA
92111	San Diego	CA
90210	Beverly Hills	CA

ZIP CODE TABLE

List all cities and states associated with a ZIP code

```
CREATE TABLE IF NOT EXISTS people (
        pid             serial          not null unique,
        fullName        text            not null unique,
        birthDate       date            not null,
        gender          char(1)         not null,
        email           varchar(256)    not null,
        phoneNum        char (15)       not null,
        streetAddress   varchar(256)    not null,
        ZIPCode         integer         not null,
        CONSTRAINT      valid_gender    CHECK (gender = 'F' OR gender = 'M'),
        PRIMARY KEY(pid),
        FOREIGN KEY(ZIPCode) references ZIPCode
);
```

Functional Dependencies:
pid → fullName, birthDate, gender, email, phoneNum, streetAddress, ZIPCode

PEOPLE TABLE

List all employees, their basic info and their contact information

pid integer	fullname text	birthdate date	gender character(1)	email character varying(256)	phonenum character(15)	streetaddress character varying(256)	zipcode integer
0	Based Codd	1900-01-01	M	bigfish@gmail.com	411-007-1234	321 Heavenly Way	92111
1	Jess Farr	1995-11-12	F	Jessica.Farr1@marist.edu	978-914-4426	2516 Merengue Street	92253
2	Jenifer Daniels	1964-09-18	F	jenifer.daniels@cbslocal2.com	760-485-5494	45705 Coldbrook Lane	92253
3	Mitchell Xanders	1995-12-04	M	Mitchell.Xanders1@marist.edu	760-851-8161	45705 Coldbrook Lane	92253
4	Alan Labouseur	1989-01-10	M	alan@labouseur.com	914-363-8244	007 Craig Drive	90210
5	Johnny Mosho	1996-06-10	M	John.Mosho1@marist.edu	627-545-3104	575 Smorgasbord Road	12601
6	Jenny Twotone	1970-02-15	F	onehitwonder@aol.com	760-867-5309	914 Tommy Lane	90052
7	Elio Velazquez	1995-11-14	M	eliov40@gmail.com	343-589-6217	444 Gravity Drive	92276
8	Anita Campbell	1978-03-06	F	wheresmyglasses@yahoo.com	818-8507-2020	650 Black Street	94114
9	Bob Frapples	1966-10-22	M	bob4apples@aol.com	366-517-2111	213 Candy Cane Lane	92262
10	Summer Skye	1990-07-06	F	shades90@yahoo.com	760-124-6630	450 Sunset Boulevard	92260
11	Brady Thomas	1977-01-24	M	fourrings@gmail.com	714-552-3694	16 Flat Street	12601
12	PJ Pants	1985-04-01	M	pajamaparty@yahoo.com	291-505-4998	304 Night Drive	92262
13	Carl Maggio	1996-03-16	M	caaaaaaaaarl@gmail.com	760-333-1790	404 Blank Circle	92260
14	Shane King	1996-05-21	M	milkisnumba1@yahoo.com	914-286-2251	311 Gamma Lane	12601
15	Astro Beamer	1997-02-14	M	cooldawgz@gmail.com	760-522-3104	400 Runaway Lane	92276
16	Tubby Fujita	1997-01-24	F	flysrscary@yahoo.com	760-210-1159	369 Coldberry Cove	94114
17	Austin Braunschweiger	1994-11-07	M	swagg@gmail.com	760-803-4110	62 Desert Road	92253
18	Eury Fabian	1993-08-21	M	e@gmail.com	410-259-3688	101 Pennsylvania Avenue	94114
19	Flair Lens	1985-12-29	F	s0h1pster@yahoo.com	911-411-1762	1985 Polaroid Road	94114
20	Captain George	1959-03-26	M	tharsheblows@yahoo.com	909-567-0045	4500 High Sea Street	92111
21	Penny Nichols	1981-06-20	F	ca$hm0ney@gmail.com	510-292-1147	25 Dime Street	92262
22	Kay Nine	1976-08-11	F	barkbark@yahoo.com	714-202-5865	12 Plateau Drive	92260
23	Jordan Michael	1963-02-17	M	retire23@gmail.com	408-362-5174	6 Ring Drive	92111
24	Michelle Bay	1979-09-25	F	BIGexplosions@yahoo.com	253-646-9000	911 Boom Boulevard	90052
25	Cam Era	1989-04-11	M	photofinish@gmail.com	309-694-1820	133 Melody Lane	94114

```
CREATE TABLE IF NOT EXISTS studioWorkers (
        sid             integer         not null unique references people(pid), --sid=pid
        position        varchar(8)      check (position='Anchor' or position='Weather' or position='Sports' or position='Camera'),
        PRIMARY KEY(sid)
);
```

Functional Dependencies:
sid → position

sid integer	position character varying(8)
2	Anchor
3	Sports
11	Sports
10	Weather
9	Anchor
16	Weather
21	Anchor
22	Anchor
20	Anchor

STUDIO WORKERS TABLE

List all employees who work in the studio by their id and what position they hold

```
CREATE TABLE IF NOT EXISTS production (
        prid            integer         not null unique references people(pid),
        position        varchar(9)      check (position='Director' or position='Producer'),
        PRIMARY KEY(prid)
);
```

prid integer	position character varying(9)
1	Director
6	Producer
13	Producer
14	Director

Functional Dependencies:
prid → position

fid integer	position character varying(8)
4	Reporter
5	Camera
7	Reporter
12	Camera
8	Camera
15	Reporter
17	Camera
19	Camera
18	Camera

Functional Dependencies:
fid → position

List all employees who work
in the field by their id and what position they hold

FIELD WORKERS TABLE AND PRODUCTION TABLE

List all employees who work in production by their id
and what position they hold

```
CREATE TABLE IF NOT EXISTS fieldWorkers (
        fid             integer         not null unique references people(pid),
        position        varchar(8)      check (position='Reporter' or position='Camera'),
        PRIMARY KEY(fid)
);
```

```
CREATE TABLE IF NOT EXISTS studioNames (
        sid             integer         not null unique references studioWorkers(sid),
        name            text            not null unique references people(fullName),
        PRIMARY KEY(sid),
        FOREIGN KEY(sid) REFERENCES studioWorkers(sid),
        FOREIGN KEY(name) REFERENCES people(fullName)
);
```

sid integer	name text
2	Jenifer Daniels
3	Mitchell Xanders
9	Bob Frapples
10	Summer Skye
11	Brady Thomas
16	Tubby Fujita
20	Captain George
21	Penny Nichols
22	Kay Nine

Functional Dependencies:
sid → name

```
CREATE TABLE IF NOT EXISTS fieldNames (
        fid             integer         not null unique references fieldWorkers(fid),
        name            text            not null unique references people(fullName),
        PRIMARY KEY(fid),
        FOREIGN KEY(fid) REFERENCES fieldWorkers(fid),
        FOREIGN KEY(name) REFERENCES people(fullName)
);
```

fid integer	name text
4	Alan Labouseur
5	Johnny Mosho
7	Elio Velazquez
8	Anita Campbell
12	PJ Pants
15	Astro Beamer
17	Austin Braunschweiger
18	Eury Fabian
19	Flair Lens

Functional Dependencies:
fid → name

List the names and department id's of all employees

STUDIO NAMES, FIELD NAMES, AND PRODUCTION NAMES TABLES

prid integer	name text
1	Jess Farr
6	Jenny Twotone
13	Carl Maggio
14	Shane King

Functional Dependencies:
sid → name

```
CREATE TABLE IF NOT EXISTS productionNames (
        prid            integer         not null unique references production(prid),
        name            text            not null unique references people(fullName),
        PRIMARY KEY(prid),
        FOREIGN KEY(prid) REFERENCES production(prid),
        FOREIGN KEY(name) REFERENCES people(fullName)
);
```

```
CREATE TABLE IF NOT EXISTS shows(
        shid            serial          not null unique,
        showTime        varchar(20)     not null CHECK (showTime='Morning' OR showTime='Evening' OR showTime='Night'),
        anchor1         integer         not null references studioWorkers(sid),
        anchor1name     text            null references studioNames(name),
        anchor2         integer         null references studioWorkers(sid),
        anchor2name     text            null references studioNames(name),
        weather         integer         not null references studioWorkers(sid),
        weathername     text            null references studioNames(name),
        sports          integer         not null references studioWorkers(sid),
        sportsname      text            null references studioNames(name),
        camera1         integer         not null references fieldWorkers(fid),
        camera1name     text            null references fieldNames(name),
        camera2         integer         not null references fieldWorkers(fid),
        camera2name     text            null references fieldNames(name),
        reporter        integer         not null references fieldWorkers(fid),
        reportername    text            null references fieldNames(name),
        director        integer         not null references production(prid),
        directorname    text            null references productionNames(name),
        producer        integer         not null references production(prid),
        producername    text            null references productionNames(name),
        PRIMARY KEY(shid),
        FOREIGN KEY(anchor1) REFERENCES studioNames(sid),
        FOREIGN KEY(anchor1name) REFERENCES studioNames(name),
        FOREIGN KEY(anchor2) REFERENCES studioNames(sid),
        FOREIGN KEY(anchor2name) REFERENCES studioNames(name),
        FOREIGN KEY(weather) REFERENCES studioNames(sid),
        FOREIGN KEY(weathername) REFERENCES studioNames(name),
        FOREIGN KEY(sports) REFERENCES studioNames(sid),
        FOREIGN KEY(sportsname) REFERENCES studioNames(name),
        FOREIGN KEY(camera1) REFERENCES fieldNames(fid),
        FOREIGN KEY(camera1name) REFERENCES fieldNames(name),
        FOREIGN KEY(camera2) REFERENCES fieldNames(fid),
        FOREIGN KEY(camera2name) REFERENCES fieldNames(name),
        FOREIGN KEY(reporter) REFERENCES fieldNames(fid),
        FOREIGN KEY(reportername) REFERENCES fieldNames(name),
        FOREIGN KEY(director) REFERENCES productionNames(prid),
        FOREIGN KEY(directorname) REFERENCES productionNames(name),
        FOREIGN KEY(producer) REFERENCES productionNames(prid),
        FOREIGN KEY(producername) REFERENCES productionNames(name)
);
```

shid integer	showtime character varying(20)	anchor1 integer	anchor1name text	anchor2 integer	anchor2name text	weather integer	weathername text	sports integer	sportsname text	camera1 integer	camera1name text	camera2 integer	camera2name text	reporter integer	reportername text	director integer	directorname text	producer integer	producername text
1	Morning	9		2		16		11		5		8		4		14		13	
2	Evening	20		21		10		3		17		18		15		1		6	
3	Night	22				10		3		12		19		7		1		6	

Functional Dependencies:
shid → showTime, anchor1, anchor2, weather, sports, camera1, camera2, reporter, director, producer

SHOWS TABLE

List the id's of the employees working each position for each program

eid integer	name character varying(45)
1	camera 1
2	camera 2
3	camera 3
4	wireless mic 1
5	wireless mic 2
6	wireless mic 3
7	wireless mic 4
8	wireless mic 5
9	microphone 1
10	microphone 2
11	microphone 3
12	microphone 4
13	tripod 1
14	tripod 2
15	tripod 3
16	van 1
17	van 2

```
CREATE TABLE IF NOT EXISTS equipment (
        eid             serial          not null unique,
        name            varchar(45)     not null,
        PRIMARY KEY(eid)
);
```

Functional Dependencies:
eid → name

EQUIPMENT TABLE

List every item available to be rented out by employees

RENTALS TABLE

Keep track of what pieces of equipment are in whose hands

Functional Dependencies:
(eid, pid) → timeOut, timeIn

```
CREATE TABLE IF NOT EXISTS rentals (
        eid             integer                         not null unique references equipment(eid),
        pid             integer                         not null references people(pid) DEFAULT '0',
        timeOut         date                            not null DEFAULT '2016-01-01',
        timeIn          date                            null,
        CONSTRAINT      valid_time                      CHECK (timeOut <= timeIn),
        PRIMARY KEY(eid,pid),
        FOREIGN KEY(eid) REFERENCES equipment(eid),
        FOREIGN KEY(pid) REFERENCES people(pid)
);
```

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	16	2016-03-26	2016-03-27
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	20	2016-02-26	2016-02-29
12	3	2016-04-01	2016-04-01
13	4	2016-01-27	
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	1	2016-02-05	2016-02-07
17	2	2016-02-17	2016-02-18

```
CREATE VIEW Full_Roster AS
SELECT people.pid, people.fullName, fieldWorkers.position
FROM people RIGHT OUTER JOIN fieldWorkers ON people.pid = fieldWorkers.fid
UNION ALL
SELECT people.pid, people.fullName, studioWorkers.position
FROM people RIGHT OUTER JOIN studioWorkers ON people.pid = studioWorkers.sid
UNION ALL
SELECT people.pid, people.fullName, production.position
FROM people RIGHT OUTER JOIN production ON people.pid = production.prid
ORDER BY position ASC;
SELECT *
FROM Full_Roster;
```

FULL ROSTER VIEW

Lists names and positions of every employee

pid integer	fullname text	position character varying
20	Captain George	Anchor
21	Penny Nichols	Anchor
22	Kay Nine	Anchor
2	Jenifer Daniels	Anchor
9	Bob Frapples	Anchor
18	Eury Fabian	Camera
19	Flair Lens	Camera
12	PJ Pants	Camera
8	Anita Campbell	Camera
5	Johnny Mosho	Camera
17	Austin Braunschweiger	Camera
14	Shane King	Director
1	Jess Farr	Director
13	Carl Maggio	Producer
6	Jenny Twotone	Producer
4	Alan Labouseur	Reporter
7	Elio Velazquez	Reporter
15	Astro Beamer	Reporter
11	Brady Thomas	Sports
3	Mitchell Xanders	Sports
16	Tubby Fujita	Weather
10	Summer Skye	Weather

```
CREATE VIEW Lineups AS
SELECT showTime, anchor1name, anchor2name, weathername, sportsname, camera1name, camera2name, reportername, directorname, producername
FROM shows
ORDER BY shid ASC;
SELECT *
FROM Lineups
```

showtime character varying(20)	anchor1name text	anchor2name text	weathername text	sportsname text	camera1name text	camera2name text	reportername text	directorname text	producername text
Morning	Bob Frapples	Jenifer Daniels	Tubby Fujita	Brady Thomas	Johnny Mosho	Anita Campbell	Alan Labouseur	Shane King	Carl Maggio
Evening	Captain George	Penny Nichols	Summer Skye	Mitchell Xanders	Austin Braunschweiger	Eury Fabian	Astro Beamer	Jess Farr	Jenny Twotone
Night	Kay Nine		Summer Skye	Mitchell Xanders	PJ Pants	Flair Lens	Elio Velazquez	Jess Farr	Jenny Twotone

LINEUPS VIEW

Shows the names of each employee and what position they have for each show

```
CREATE VIEW Inventory AS
SELECT people.fullName, equipment.name, rentals.timeOut, rentals.timeIn
FROM people, equipment, rentals
WHERE people.pid = rentals.pid
  AND equipment.eid = rentals.eid;
SELECT *
FROM Inventory
ORDER BY name ASC;
```

fullname text	name character varying(45)	timeout date	timein date
Alan Labouseur	camera 1	2016-03-09	2016-03-11
Johnny Mosho	camera 2	2016-03-07	2016-03-07
Tubby Fujita	camera 3	2016-03-26	2016-03-27
Bob Frapples	microphone 1	2016-04-20	2016-04-23
Eury Fabian	microphone 2	2016-03-06	2016-03-09
Captain George	microphone 3	2016-02-26	2016-02-29
Mitchell Xanders	microphone 4	2016-04-01	2016-04-01
Alan Labouseur	tripod 1	2016-01-27	
Alan Labouseur	tripod 2	2016-02-25	2016-04-26
Elio Velazquez	tripod 3	2016-01-19	2016-03-26
Jess Farr	van 1	2016-02-05	2016-02-07
Jenifer Daniels	van 2	2016-02-17	2016-02-18
Carl Maggio	wireless mic 1	2016-03-11	2016-03-12
Mitchell Xanders	wireless mic 2	2016-02-06	2016-02-11
Shane King	wireless mic 3	2016-03-11	2016-03-12
Anita Campbell	wireless mic 4	2016-03-12	2016-03-14
Penny Nichols	wireless mic 5	2016-04-07	2016-04-07

INVENTORY VIEW

Lists the name of the employee who most recently had each item and if that item is still out

```
SELECT people.fullName
FROM people
WHERE people.ZIPCode NOT IN (SELECT ZIPCode
                             FROM ZIPCode
                             WHERE state='CA'
                            )
```

fullname text
Johnny Mosho
Brady Thomas
Shane King

OUT-OF-STATERS REPORT

Find the employees who came to the station from out of state

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	14	2016-04-26	
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	1	2016-04-26	2016-04-26
12	4	2016-04-26	2016-04-26
13	7	2016-04-26	
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	22	2016-04-26	
17	2	2016-02-17	2016-02-18

```
SELECT people.fullName, equipment.name
FROM rentals
LEFT OUTER JOIN people ON rentals.pid = people.pid
LEFT OUTER JOIN equipment ON rentals.eid = equipment.eid
WHERE rentals.timeIn IS NULL;
```

fullname text	name character varying(45)
Elio Velazquez	tripod 1
Shane King	camera 3
Kay Nine	van 1

CHECKED OUT REPORT

List all items currently being rented out by employees
and the employee who checked each piece of equipment out

```
CREATE OR REPLACE FUNCTION identify_cast() RETURNS trigger AS
$$
BEGIN
        IF NEW.anchor1 IN (SELECT anchor1
                           FROM shows
                           WHERE anchor1name IS NULL
                           OR anchor1name NOT IN (SELECT name
                                                  FROM studioNames
                                                  WHERE studioNames.sid = NEW.anchor1)
                           )
        THEN
            UPDATE shows
            SET anchor1name = studioNames.name
            FROM studioNames
            WHERE shows.anchor1 = studioNames.sid;
        END IF;
        IF NEW.anchor2 IN (SELECT anchor2
                           FROM shows
                           WHERE anchor2name IS NULL
                           OR anchor2name NOT IN (SELECT name
                                                  FROM studioNames
                                                  WHERE studioNames.sid = NEW.anchor2)
                           )
        THEN
            UPDATE shows
            SET anchor2name = studioNames.name
            FROM studioNames
            WHERE shows.anchor2 = studioNames.sid;
        END IF;
        IF NEW.weather IN (SELECT weather
                           FROM shows
                           WHERE weathername IS NULL
                           OR weathername NOT IN (SELECT name
                                                  FROM studioNames
                                                  WHERE studioNames.sid = NEW.weather)
                           )
        THEN
            UPDATE shows
            SET weathername = studioNames.name
            FROM studioNames
            WHERE shows.weather = studioNames.sid;
        END IF;
        IF NEW.sports IN (SELECT sports
                          FROM shows
                          WHERE sportsname IS NULL
                          OR sportsname NOT IN (SELECT name
                                                FROM studioNames
                                                WHERE studioNames.sid = NEW.sports)
                          )
        THEN
            UPDATE shows
            SET sportsname = studioNames.name
            FROM studioNames
            WHERE shows.sports = studioNames.sid;
        END IF;
        IF NEW.camera1 IN (SELECT camera1
                           FROM shows
                           WHERE camera1name IS NULL
                           OR camera1name NOT IN (SELECT name
                                                  FROM fieldNames
                                                  WHERE fieldNames.fid = NEW.camera1)
                           )
        THEN
            UPDATE shows
            SET camera1name = fieldNames.name
            FROM fieldNames
            WHERE shows.camera1 = fieldNames.fid;
        END IF;
        IF NEW.camera2 IN (SELECT camera2
                           FROM shows
                           WHERE camera2name IS NULL
                           OR camera2name NOT IN (SELECT name
                                                  FROM fieldNames
                                                  WHERE fieldNames.fid = NEW.camera2)
                           )
        THEN
            UPDATE shows
            SET camera2name = fieldNames.name
            FROM fieldNames
            WHERE shows.camera2 = fieldNames.fid;
        END IF;
```

```
        IF NEW.reporter IN (SELECT reporter
                           FROM shows
                           WHERE reportername IS NULL
                           OR reportername NOT IN (SELECT name
                                                   FROM fieldNames
                                                   WHERE fieldNames.fid = NEW.reporter)
                           )
        THEN
            UPDATE shows
            SET reportername = fieldNames.name
            FROM fieldNames
            WHERE shows.reporter = fieldNames.fid;
        END IF;
        IF NEW.director IN (SELECT director
                           FROM shows
                           WHERE directorname IS NULL
                           OR directorname NOT IN (SELECT name
                                                   FROM productionNames
                                                   WHERE productionNames.prid = NEW.director)
                           )
        THEN
            UPDATE shows
            SET directorname = productionNames.name
            FROM productionNames
            WHERE shows.director = productionNames.prid;
        END IF;
         IF NEW.producer IN (SELECT producer
                            FROM shows
                            WHERE producername IS NULL
                            OR producername NOT IN (SELECT name
                                                    FROM productionNames
                                                    WHERE productionNames.prid = NEW.producer)
                            )
         THEN
             UPDATE shows
             SET producername = productionNames.name
             FROM productionNames
             WHERE shows.producer = productionNames.prid;
         END IF;
RETURN NULL;
END;
$$
LANGUAGE PLPGSQL;
```

Sample Data with Trigger on p.22

STORED PROCEDURE: IDENTIFY CAST

This automatically updates the name of the employee being assigned a position whenever their id is entered in an UPDATE query

```
CREATE OR REPLACE FUNCTION checkRent (integer) returns date AS
$$
DECLARE
        desiredItem     integer         = $1;
BEGIN
        IF desiredItem IN (SELECT eid
                           FROM rentals
                           WHERE timeIn IS NULL)
        THEN RAISE EXCEPTION 'Item not available to be checked out';
        END IF;
        RETURN timeIn FROM rentals
        WHERE eid = desiredItem;
END;
$$
LANGUAGE PLPGSQL;

SELECT checkRent(16);
```

```
CREATE OR REPLACE FUNCTION checkOut (integer, integer, date) returns date AS
$$
DECLARE
        checkoutEid     integer         = $1;
        checkoutPid     integer         = $2;
        checkoutTime    date            = $3;

BEGIN
        PERFORM checkRent(checkoutEid);
        IF checkRent(checkoutEid) IS NULL
        THEN RAISE EXCEPTION 'Item not available to be checked out';
        END IF;
        UPDATE rentals SET pid = $2 WHERE eid = $1;
        UPDATE rentals SET timeIn = NULL WHERE rentals.eid = $1
                                        AND rentals.pid = $2;
        UPDATE rentals SET timeOut = $3 WHERE eid = $1;

        RETURN checkoutTime;
END;
$$
LANGUAGE PLPGSQL;

SELECT checkOut (11,1,CURRENT_DATE);
```

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	16	2016-03-26	2016-03-27
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	20	2016-02-26	2016-02-29
12	3	2016-04-01	2016-04-01
13	4	2016-01-27	
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	1	2016-02-05	2016-02-07
17	2	2016-02-17	2016-02-18

← Before Procedure

After Procedure →

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	16	2016-03-26	2016-03-27
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	1	2016-04-26	
12	4	2016-04-26	2016-04-26
13	4	2016-01-27	
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	1	2016-02-05	2016-02-07
17	2	2016-02-17	2016-02-18

STORED PROCEDURES: CHECK RENT AND CHECKOUT

These procedures automatically check the availability of an item and check it out to the user if it is available

```
CREATE OR REPLACE FUNCTION checkReturn (integer) returns date AS
$$
DECLARE
        desiredItem     integer         = $1;
BEGIN
        IF desiredItem IN (SELECT eid
                           FROM rentals
                           WHERE timeIn IS NOT NULL)
        THEN RAISE EXCEPTION 'This item is still here';
        END IF;
        RETURN timeOut FROM rentals
        WHERE eid = desiredItem;
END;
$$
LANGUAGE PLPGSQL;

SELECT checkReturn(11);
```

```
CREATE OR REPLACE FUNCTION turnIn (integer, integer, date) returns date AS
$$
DECLARE
        returnEid       integer         = $1;
        returnPid       integer         = $2;
        itemTimeIn      date            = $3;
BEGIN
        PERFORM checkReturn(returnEid);
        IF checkReturn(returnEid) IS NULL
        THEN RAISE EXCEPTION 'This item is still here';
        END IF;
        UPDATE rentals SET pid = $2 WHERE eid = $1;
        UPDATE rentals SET timeIn = $3 WHERE eid = $1;
        RETURN itemTimeIn;
END;
$$
LANGUAGE PLPGSQL;

SELECT turnIn(13,4,CURRENT_DATE);
```

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	16	2016-03-26	2016-03-27
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	20	2016-02-26	2016-02-29
12	3	2016-04-01	2016-04-01
13	4	2016-01-27	
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	1	2016-02-05	2016-02-07
17	2	2016-02-17	2016-02-18

← Before Procedure

After Procedure →

eid integer	pid integer	timeout date	timein date
1	4	2016-03-09	2016-03-11
2	5	2016-03-07	2016-03-07
3	16	2016-03-26	2016-03-27
4	13	2016-03-11	2016-03-12
5	3	2016-02-06	2016-02-11
6	14	2016-03-11	2016-03-12
7	8	2016-03-12	2016-03-14
8	21	2016-04-07	2016-04-07
9	9	2016-04-20	2016-04-23
10	18	2016-03-06	2016-03-09
11	1	2016-04-26	2016-04-26
12	4	2016-04-26	2016-04-26
13	4	2016-01-27	2016-04-26
14	4	2016-02-25	2016-04-26
15	7	2016-01-19	2016-03-26
16	1	2016-02-05	2016-02-07
17	2	2016-02-17	2016-02-18

STORED PROCEDURES: CHECK RETURN AND TURN IN

These procedures check to see if the inputted item is currently being rented out and allows the user to return the item if it is

```
CREATE OR REPLACE FUNCTION identify_new_studioWorker() RETURNS trigger AS
$$
BEGIN
        IF NEW.sid = (SELECT MAX(sid) FROM studioWorkers) THEN
                INSERT INTO studioNames (sid, name)
                SELECT NEW.sid, people.fullName
                FROM people
                WHERE NEW.sid = people.pid;
        END IF;
        RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
CREATE TRIGGER identify_new_studioWorker
AFTER INSERT ON studioWorkers
FOR EACH ROW
EXECUTE PROCEDURE identify_new_studioWorker();
```

```
INSERT INTO studioWorkers
VALUES ((SELECT pid FROM people WHERE fullName='Jordan Michael'), 'Sports');
```

Without Trigger

sid integer	name text
2	Jenifer Daniels
3	Mitchell Xanders
9	Bob Frapples
10	Summer Skye
11	Brady Thomas
16	Tubby Fujita
20	Captain George
21	Penny Nichols
22	Kay Nine

With Trigger

sid integer	name text
2	Jenifer Daniels
3	Mitchell Xanders
9	Bob Frapples
10	Summer Skye
11	Brady Thomas
16	Tubby Fujita
20	Captain George
21	Penny Nichols
22	Kay Nine
23	Jordan Michael

TRIGGER: IDENTIFY NEW STUDIO WORKERS

This trigger automatically enters the name of a new employee entering a department

```
CREATE TRIGGER identify_cast
AFTER UPDATE ON shows
FOR EACH ROW
EXECUTE PROCEDURE identify_cast();
```

Without Trigger

shid integer	showtime character varying(20)	anchor1 integer	anchor1name text	anchor2 integer	anchor2name text	weather integer	weathername text	sports integer	sportsname text	camera1 integer	camera1name text	camera2 integer	camera2name text	reporter integer	reportername text	director integer	directorname text	producer integer	producername text
1	Morning	9		2		16		11		5		8		4		14		13	
2	Evening	20		21		10		3		17		18		15		1		6	
3	Night	22				10		3		12		19		7		1		6	

```
UPDATE shows
SET camera1 = 5
WHERE shid = 1;
```

With Trigger

shid integer	showtime character varying(20)	anchor1 integer	anchor1name text	anchor2 integer	anchor2name text	weather integer	weathername text	sports integer	sportsname text	camera1 integer	camera1name text	camera2 integer	camera2name text	reporter integer	reportername text	director integer	directorname text	producer integer	producername text
1	Morning	9	Bob Frapples	2	Jenifer Daniels	16	Tubby Fujita	11	Brady Thomas	5	Johnny Mosho	8	Anita Campbell	4	Alan Labouseur	14	Shane King	13	Carl Maggio
2	Evening	20	Captain George	21	Penny Nichols	10	Summer Skye	3	Mitchell Xanders	17	Austin Braunschweiger	18	Eury Fabian	15	Astro Beamer	1	Jess Farr	6	Jenny Twotone
3	Night	22	Kay Nine			10	Summer Skye	3	Mitchell Xanders	12	PJ Pants	19	Flair Lens	7	Elio Velazquez	1	Jess Farr	6	Jenny Twotone

TRIGGER: IDENTIFY CAST

ADMIN

```
CREATE ROLE admin;
GRANT ALL ON ALL TABLES
SCHEMA PUBLIC
TO admin;
```

PERSONALITIES

```
CREATE ROLE personalities;
GRANT SELECT ON people, studioNames, fieldNames,
                        productionNames, shows,
                        equipment, rentals
TO personalities;
GRANT UPDATE ON people, shows, rentals
TO personalities;
```

SECURITY AND PRIVILEGES

MANAGEMENT

```
CREATE ROLE management;
GRANT SELECT ON people, studioWorkers, fieldWorkers, production,
                shows, equipment, rentals
TO management;
GRANT INSERT, UPDATE ON people, studioWorkers, fieldWorkers,
                production, shows, equipment, rentals
TO management;
```

SECURITY AND PRIVILEGES (CONT.)

There were some difficulties in trying to find a way to display the name of every person with a spot on any program, and I'm not sure the most efficient way was chosen, but thanks to some triggers it got the job done. There definitely is room to expand as evidenced by the CHECK in the studioWorkers table allowing for 'Camera' as an acceptable value in the positions column. With more sample data it'd be possible to expand the positions available and possibly create a more complex table for shows.

Overall, it is a solid start to a database, especially in the rentals and returns tables, but it has a lot of room for expansion provided the right resources.

NOTES, ISSUES, & THE FUTURE