STAR TREK- THE FINAL (DATABASE)FRONTIER

A DATABASE BY AMELIA BECKHAM

TABLE OF CONTENTS

- PAGE 2- TABLE OF CONTENTS
- PAGE 3- EXECUTIVE SUMMARY OVERVIEW
- PAGE 4- EXECUTIVE SUMMARY OBJECTIVES
- PAGE 5- E/R DIAGRAM
- PAGES 6-11- CREATE STATEMENTS FOR EACH TABLES
  - *PAGE 6- PEOPLE CREATE STATEMENTS*
  - *PAGE 7-STAFF CREATE STATEMENTS*
  - *PAGE 8- EPISODES CREATE STATEMENTS*
  - *PAGE 9-SERIES CREATE STATEMENTS*
  - *PAGE 10-JOBS CREATE STATEMENTS*
  - *PAGE 11-CHARACTERS CREATE STATEMENTS*
- PAGES12-17 SAMPLE DATA FOR ALL TABLES
  - *PAGE 12-SAMPLE DATA –PEOPLE*
  - *PAGE 13-SAMPLE DATA -STAFF*
  - *PAGE 14-SAMPLE DATA -EPISODES*
  - *PAGE 15-SAMPLE DATA -SERIES*
  - *PAGE 16-SAMPLE DATA –JOBS*
  - *PAGE 17-SAMPLE DATA-CHARACTERS*
- PAGE 18- VIEW STATEMENTS
- PAGE 19-TRIGGER STATEMENT
- PAGE 20- WHOWASINWHATSHOW STORED PROCEDURE
- PAGE 21-STORED PROCEDURE
- PAGE 22-EPISODEINTERVAL PROCEDURE
- PAGES 23-24-CAREER PROCEDURE
  - *PAGE 23- CAREER PROCEDURE*
  - *PAGE 24-CAREER PROCEDURE(CONTINUED)*
- PAGES 25-26-REPORTS AND QUERIES
  - *PAGE 25-REPORTS AND QUERIES*
  - *PAGE 26-REPORST AND QUERIES*
- PAGES 27-28-CLUSTERED INDEXES
  - *PAGE 27- CLUSTERED INDEXES*
  - *PAGE 28- CLUSTERED INDEXES CODE*
- PAGE 29-SECURITY
- PAGE 30-IMPLEMENTATION NOTES
- PAGE 31-KNOWN PROBLEMS
- PAGE 32-FUTURE ENHANCEMENTS

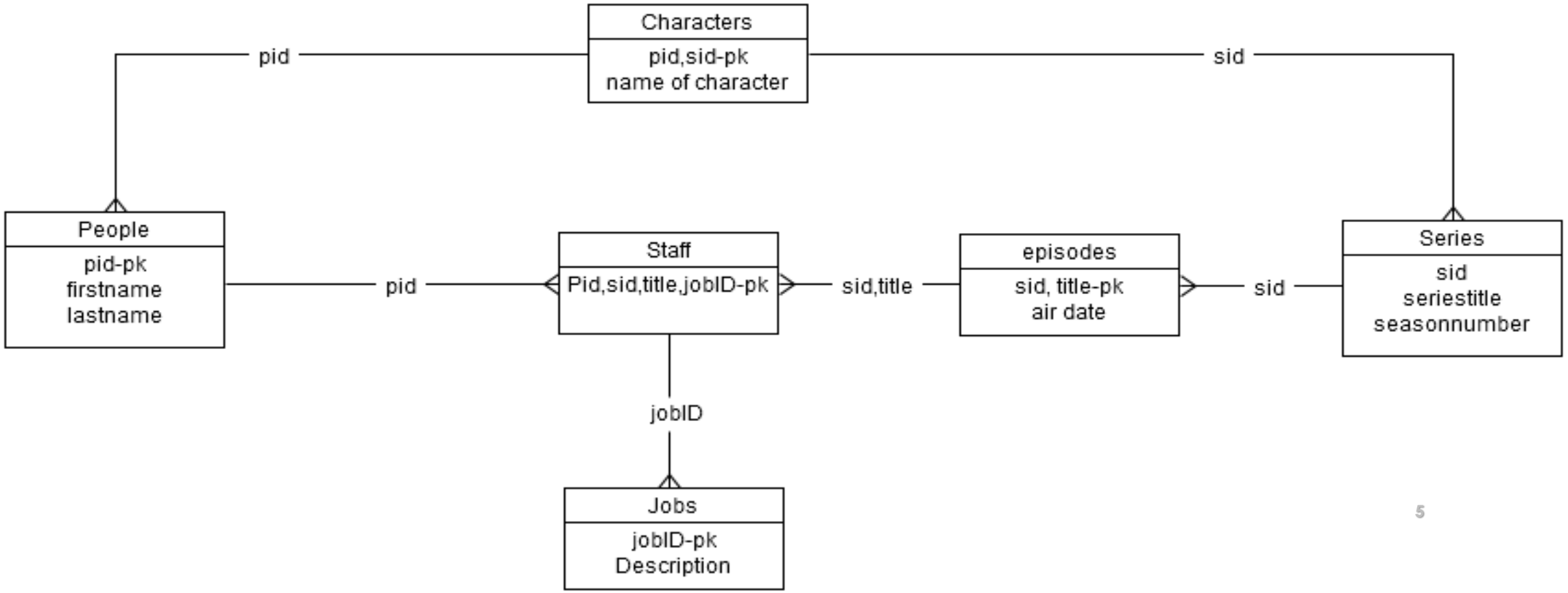
Executive Summary-Overview

- Star trek is an international sensation, having spawned 12 movies, 6 different television series and numerous video games. Until now, someone interested in details such as when a certain episode premiered or what roles Michael dorn played (the answer is worf) would have to collect this information from numerous web pages and other sources. With this database, this information is readily available and quickly accessible. This database is aimed at the general public, especially star trek fans or people who are generally curious about star trek.
- The database includes the actors, screen writers, story writers, directors, characters, series and episode title each person appeared in.
- Lastly, the database has the job title for each person.
  - Actor
  - Voice Actor
  - Director

Executive Summary-Objectives

- Using Postgres, create a database that is flexible enough to eventually expand to contain all of the star trek episodes and eventually other TV shows.
- Must be in BCNF
- Is complete and is user friendly
- Uses triggers and stored procedures to keep the data easy to use
- is accurate and is moderately fast

THE E/R DIAGRAM



PEOPLE TABLE-CREATE STATEMENT

- THIS TABLE LINKS THE PID (PEOPLE ID) WITH THE FIRST AND LAST NAMES OF ALL THE PEOPLE INVOLVED IN STAR TREK

```
Create table people (
  pid       char(4) not null,
  firstname     varchar(16),
  lastname    varchar(20),
primary key(pid)
);
```

- FUNCTIONAL DEPENDENCY: PID ⟶ FIRSTNAME,LASTNAME

STAFF CREATE STATEMENTS

- THIS TABLE CONTAINS THE PID (PEOPLE ID) SID (SERIES ID), EPISODE TITLE AND JOBID (ID OF THE PERSONS JOB) BY USING A COMPOSITE KEY MADE UP OF THE OTHER PRIMARY KEYS FROM THE DATABASE

```
CREATE TABLE STAFF (
  PID       CHAR(4) NOT NULL,
  SID       CHAR(4) NOT NULL,
  TITLE     VARCHAR(20),
  JOBID      CHAR(4) NOT NULL,
PRIMARY KEY(PID,SID,TITLE,JOBID)
);
```

- FUNCTIONAL DEPENDENCY: PID,SID,TITLE,JOBID  PID,SID,TITLE,JOBID

EPISODES CREATE STATEMENTS

- EPISODES IS A TABLE OF ALL OF THE EPISODES IN THE DIFFERENT SERIES OF STAR TREK AND WHAT SERIES THEY WERE IN

```
CREATE TABLE EPISODES (
  SID      CHAR(4) NOT NULL,
  TITLE     VARCHAR(20),
PRIMARY KEY(TITLE)
);
ALTER TABLE EPISODES ADD ORIGINALAIRDATE DATE
```

- FUNCTIONAL DEPENDENCY: SID,TITLE ORIGINALAIRDATE

SERIES CREATE STATEMENTS

THIS TABLE COVERS THE ORIGINAL SERIES UP TO DEEP SPACE NINE, AND THE NUMBER OF SEASONS IN EACH SERIES

```
CREATE TABLE SERIES (
  SID      CHAR(4) NOT NULL,
PRIMARY KEY(SID)
);
```

- FUNCTIONAL DEPENDENCY SID ⟶ TITLE OF SERIES AND NUMBER OF SEASONS

JOBS CREATE STATEMENTS

- THE JOBS TABLE IS ALL THE DIFFERENT JOBS PEOPLE HAD ON STAR TREK

```
CREATE TABLE JOBS (
 JOBID      CHAR(4) NOT NULL,
 DESCRIPTION VARCHAR(20),
PRIMARY KEY(JOBID)
);
```

- FUNCTIONAL DEPENDENCIES: JOBID ⟶ DESCRIPTION

CHARACTERS CREATE STATEMENT

- SIMILAR TO THE JOBS TABLE, THE CHARACTERS TABLE HAS THE PID OF THE PERSON WHO PLAYED A CHARACTER IN STAR TREK, THE ID OF THE SERIES THEY WERE IN AND THAT CHARACTERS NAME.

```
CREATE TABLE CHARACTERS (
  PID       CHAR(4) NOT NULL,
  SID       CHAR(4) NOT NULL,
 CHARACTERNAME      VARCHAR(20),
PRIMARY KEY(PID,SID)
)
```

- FUNCTIONAL DEPENDENCY PID,SID ⟶ CHARACTERNAME

SAMPLE DATA- PEOPLE

This is some data for the table people. It is not the whole table, as the whole table is 372 rows.

Pid Character(4)	Firstname Character varying (16)	Lastname Character varying (20)
p029	Frank	Abatemarco
p030	Stanley	Adams
p031	Alan	Adler
p032	David	Alexander
p033	Corey	Allen
p034	Andrea	Alton
p035	Gregory	Amos

SAMPLE DATA- STAFF

This is some data for the table people. It is not the whole table, as the whole table is 5154 rows

Pid Character(4)	Sid Character(4)	Title Character(60)	Jobid Character(4)
p010	s002	1100100	j001
p011	s002	1100100	j001
p012	s002	1100100	j001
p013	s002	1100100	j001
p014	s002	1100100	j001
p015	s002	1100100	j001
p017	s002	1100100	j001

SAMPLE DATA- EPISODES

This is sample data from episodes, note that it is not the whole table, as the whole table is 430 rows

Sid Character(4)	Title Character varying(60)	Originalairdate date
s001	The Man Trap	1966-09-08
s001	Charlie X	1966-09-15
s001	Where No Man Has Gone Before	1966-09-22
s001	The Naked Time	1966-09-29
s001	The Enemy Within	1966-10-06
s001	Mudd's Women	1966-10-13
s001	What Are Little Girls Made Of?	1966-10-20

SAMPLE DATA- SERIES

This is the whole table series

Sid Character (4)	Seriestitle Varying character (50)	Numberofseasons Character (1)
s001	Star Trek: The Original Series	3
s002	Star Trek: The Next Generation	7
s003	Star Trek: Deep Space Nine	7

SAMPLE DATA- JOBS

This is the whole table jobs

Jobid character (4)	Description character varying (20)
j001	Actor
j002	Voice Actor
j003	Director
j004	Playwright
j005	Storywriter

SAMPLE DATA- CHARACTERS

This is sample data from the table characters, the whole table is 32 rows long

Pid Character (4)	Sid Character(4)	Charactername Character varying (20)
p001	s001	James T. Kirk
p002	s001	Commander Spock
p003	s001	Bones McCoy
p004	s001	Scotty
p005	s001	Uhura
p006	s001	Sulu
p007	s001	Checkov
p008	s001	Nurse Chapel
p009	s001	Yeoman Rand

VIEW STATEMENTS

- SECURITY HAS BEEN ENHANCED BY CREATING VIEWS OF THE ORIGINAL TABLES AND ALLOWING THE USER TO ONLY QUERY THE VIEWS
- THIS IS THE CREATE STATEMENT FOR THE PEOPLE TABLE FOR SECURITY (ORIGINAL TABLE IS NOT QUERIED BY USER)

```
CREATE VIEW PEOPLEINSTARTREK AS
SELECT DISTINCT FIRSTNAME,LASTNAME FROM PEOPLE
ORDER BY FIRSTNAME;
```

- THIS IS A VIEW FOR THE STAFF TABLE, FOR SECURITY. THE OTHER TABLES HAVE SIMILAR VIEWS

```
CREATE VIEW STAFFFORSTARTREK AS
SELECT * FROM STAFF;
```

TRIGGER STATEMENT

- TRIGGER STATEMENT TO MAKE SURE THEIR ARE NO DUPLICATES IN THE TABLE PEOPLE

```
CREATE FUNCTION CHECK_PEOPLE_INPUT() RETURNS TRIGGER AS $C_P_I$
BEGIN
IF EXISTS (SELECT PEOPLE.PID FROM PEOPLE WHERE PID = NEW.PID) THEN
RAISE EXCEPTION 'YOU HAVE TRIED TO INSERT A DUPLICATE PID';
END IF;
IF (EXISTS (SELECT PEOPLE.FIRSTNAME FROM PEOPLE WHERE FIRSTNAME = NEW.FIRSTNAME) AND
 EXISTS (SELECT PEOPLE.LASTNAME FROM PEOPLE WHERE LASTNAME = NEW.LASTNAME)) THEN
RAISE EXCEPTION 'THAT NAME IS ALREADY IN THE DATABASE';
END IF;
RETURN NEW;
END;
$C_P_I$ LANGUAGE PLPGSQL;
CREATE TRIGGER CHECK_PEOPLE_INPUT BEFORE INSERT OR UPDATE ON PEOPLE
FOR EACH ROW EXECUTE PROCEDURE CHECK_PEOPLE_INPUT();
```

WHOWASINWHATSHOW STORED PROCEDURE

This function, when called with a last name, returns the firstname, lastname, episode title, role in episode and original air date.

```
create function whowasinwhatshow (lastname character(20))

Returns table(firstname character(16), lastname character(20), whatepisodes2 varchar(60), description character(20), originalairdate date) as $$

select people.firstname, people.lastname, episodes.title, jobs.description, episodes.originalairdate from

jobs inner join (episodes inner join (people inner join staff on (people.pid = staff.pid)) on episodes.title = staff.title) on jobs.jobid=staff.jobid

where people.lastname = $1;

$$ language 'sql';
```

STORED PROCEDURE

- CREATES TABLE TO PUT THE INFO IN
- USED THE DEFAULT LEVEL OF ISOLATION IN POSTGRES, SINCE IT IS THE HIGHEST LEVEL OF ISOLATION POSSIBLE
- THE REASON FOR THIS IS ELIMINATE OTHER PROBLEMS THAT CAN BE CAUSED BY LOWER LEVELS OF ISOLATION
- EXAMPLE:

```
SELECT * FROM WHOWASINWHATSHOW('SHATNER');
```

- OUTPUT (SAMPLE)

Firstname	Lastname	Title	Job	originalairdate
William	Shatner	A Piece of the Action	Actor	1968-01-12
William	Shatner	A Private Little War	Actor	1968-02-02
William	Shatner	All Our Yesterdays	Actor	1969-03-14
William	Shatner	And the Children Shall Lead	Actor	1968-10-11

EPISODEINTERVAL PROCEDURE

THIS FUNCTION, WHEN CALLED WITH DATE RANGE, RETURNS THE FIRSTNAME, LASTNAME, EPISODE TITLE, ROLE IN EPISODE AND ORIGINAL AIR DATE IF THE ORIGINAL AIR DATE IS WITHINI THE DATE RANGE

```
CREATE FUNCTION episodeinterval(IN date1 date, date2 date)
 RETURNS TABLE(firstname character, lastname character, whatepisodes3 character varying, description character, originalairdate date) AS
$BODY$
select people.firstname, people.lastname, episodes.title, jobs.description, episodes.originalairdate from
 (jobs inner join (episodes inner join (people inner join staff on (people.pid = staff.pid)) on episodes.title = staff.title) on jobs.jobid=staff.jobid)
 where originalairdate between $1 and $2;
$BODY$
  LANGUAGE sql VOLATILE;
```

CAREER PROCEDURE

THIS FUNCTION, WHEN CALLED WITH LAST NAME, RETURNS THE FIRSTNAME, LASTNAME, ROLE IN EPISODE, EARLIEST DATE IN THAT ROLE, LAST DATE IN THAT ROLE AND A COUNT OF HOW MANY APPEARANCES IN THAT JOB.

```
CREATE OR REPLACE FUNCTION career(IN lastname character)
  RETURNS TABLE(firstname character, lastname character, description character,
   minoriginalairdate date, maxoriginalairdate date, episodecount bigint) AS
$BODY$
SELECT people.firstname, people.lastname, jobs.description, min(episodes.originalairdate),
max(episodes.originalairdate),count(episodes.title) from
 jobs inner join (episodes inner join (people inner join staff on (people.pid = staff.pid)) on episodes.title =
staff.title) on jobs.jobid=staff.jobid
 where people.lastname = $1
 group by description,firstname,lastname;
$BODY$
  LANGUAGE sql VOLATILE;
```

CAREER PROCEDURE (CONTINUED)

EXAMPLE:

SELECT * FROM CAREER('BARRETT');

OUTPUT

Firstname	Lastname	Job	Earliest date	Last date	Count of appearances
Majel	Barrett	Voice Actor	1987-11-30	1999-05-26	130
Majel	Barrett	Actor	1966-09-29	1988-10-15	33

REPORTS AND QUERIES

- This report allows a person to view the complete series of star trek, along with all of the episodes in the series

```
create view compeleteseries as

select series.seriestitle, episodes.title, episodes.originalairdate from series inner join episodes on episodes.sid = series.sid;
```

- This query allows a person to see all of the episodes in the original series

```
create view theoriginalseries as

select title

from episodes

Where sid='s001'
```

REPORTS AND QUERIES

- This query allows a person to see all of the episodes in the original series

```
create view theoriginalseries as
select title
from episodes
Where sid='s001`
```

- This query allows a person to see all of the episodes data was in

```
create view episodesdatawasin as
select * from episodes
where sid in (select sid
from characters
where nameofcharacter='Data')
```

CLUSTER INDEXES

- EACH OF MY LARGE TABLES (STAFF, EPISODES AND PEOPLE) HAS AN CLUSTERED INDEX
- THESE 3 TABLES ARE THE LARGEST, SO THEY EACH GET A CLUSTERED INDEX TO SPEED UP THE DATABASE
- TOO MANY INDEXES WILL SLOW DOWN THE DATABASE, SO SMALLER TABLES DO NOT GET CLUSTERED INDEXES

CLUSTERED INDEXES CODE

- Staff clustered index

```
create index starsterkstaff on staff (pid,sid,title,jobid)
cluster staff using starsterkstaff
```

- Episodes clustered index

```
create index starsterkepisodes on episodes (sid,title,originalairdate)
cluster episodes using starsterkepisodes
```

- People clustered index

```
create index starsterkpeople on people (pid,firstname,lastname)
cluster people using starsterkpeople
```

SECURITY

- Used default security in my database
- views of the base tables and the query reports are used to keep the general public out of the actual tables
- Normal Database access is by username locutus with password borg
  - Locutus has authorization only for select and execute functions
  - Can not alter, update, insert or drop tables
- Database administration is by username postgres

Implementation Notes

- People's names are limited to first name and last name
  - Much of the input data has names of the form firstname, middlename,lastname or firstname, middleinitial,lastname
  - Data must be put into proper form (firstname, lastname only)
  - Jobids can be any 4 characters, but form jXXX where X is 0-9 is preferred
  - Similarly, sid has the form sXXX and pid the form pXXX

KNOWN PROBLEMS

- DATA LIMITED TO FIRST THREE STAR TREK SERIES
- JOBS LIMITED TO ACTOR,VOICE ACTOR,DIRECTOR, STORYWRITER AND PLAYWRIGHT
- INFLEXIBLE FORMAT FOR PEOPLE'S NAMES

FUTURE ENHANCEMENTS

- INPUT VERIFICATION FOR ALL TABLE INSERTS/UPDATES, NOT JUST ON TABLE PEOPLE
- ADD THE REST OF THE STAR TREK SERIES
- ADD ALIENS, SHIPS, MORE CHARACTERS
- ADD MORE TV SHOWS, SUCH AS DR.WHO, TRANSFORMERS, ETC
- ADD PROMPT TO UPDATE ALL TABLES WHEN ONE TABLE IS UPDATED