ATTENDANTZ: a Pg/plSQL,C++.. Proof of concept guestlist and ticketing system

Matthew E. Duffy

1.
the non-profit organization godberd an artist run organization to promote collaborations and systems for local montreal and international artist and partner organizations, from organizes courses and a non-profit education components, to host and organizes events, in three of its locations, a gallery gamma, a even small venue "name unknown" and a larger space space named avil. the three are connected. from vernisage, to video performance, musical, and late even underground art music or dance.
system requirement of phase 2 database. the connection from the payment processing needs, to produce a "guest_list" accessible by door people via a epr.
thus, the first part of the database that needs to be operational.
dealing with "attendance" to events.
the venue needs very strict attendance policies to follow the laws around nonprofits. as well as the desire to have alcohol provided however not for sale thus, no cash must occur and tickets can not be sold at the events themselves. all money must be processed off site, and not be connected directly to events, must be contributions, and must be on a generated guest list, and check in at the door. where they can get a coat check ticket and their coat stored.

2.
//reports// See appendix 2.1-2.3 for wireframe
guest_list_master_event;
a uptable list containing names, and ticket numbers of guest for a particular event, as well as display total guest count. name of event and the date start. and end time. in the corner.

final_event_report;
event name,
event id,
date end,
total number of guest attended,
total related contributions,

active_gust_list;
one is a active "guest_list" that displays, and inputting a ticket number brings up the gust and marks them "attending".

3. events(id_event, begin_date, end_date, name, budget, revenue)
guests(id_guest, name, email, total_contribution)
locations(id_location, name, capacity)
contributions(id_contri, c_date, c_amount,c_type,frkey_id_guest,frkey_id_event)
data_schedual (id_event, id_location)

4. SEE appendix 4.1 for ER, 4.2 for overview context map, and 4.3 for a draft DFD1 dealing with the input output process for guestlist.

5.

```
SET search_path TO attendants;
```

schéma attendants
[authorization]?

[***schema_element***]

create table locations
(id_loc bigint Primary Key,
l_name varchar(120),
capacity int);

constraint pk_loc Primary key (id_loc),

Create table Contributions
id_contri PRIMARY KEY
is_event bool,
c_date timestamp not NULL,
c_amount decimal(18,4) not null Default '00.00',
c_type char(10) ,
frkey_id_guest bigint,
frkey_id_event bigint
;
alter table contributions add constraint check chk_ife_then_ekeyid (NOT (is_event = true AND frkey_id_event IS NULL));
alter table contributions add constraint check hk_ifnote_then_noev CHECK (NOT (NOT is_event = true AND frkey_id_event IS NOT NULL))

Indexes:
"pk_contri" PRIMARY KEY, btree (id_contri)
Check constraints:
"chk_ife_then_ekeyid" CHECK (NOT (is_event = true AND frkey_id_event IS NULL))
"chk_ifnote_then_noev" CHECK (NOT (NOT is_event = true AND frkey_id_event IS NOT NULL))

"chk_type" CHECK (c_type = ANY (ARRAY['event'::bpchar, 'donation'::bpchar, 'other'::bpchar]))

"frkey_guestnl" CHECK (frkey_id_guest IS NOT NULL)

Foreign-key constraints:
"fk_contri_guest" FOREIGN KEY (frkey_id_guest) REFERENCES guests(id_guest)
"frkey_contri_event" FOREIGN KEY (frkey_id_event) REFERENCES events(id_event)

create table events
(id_event bigint constraint id_ekey Primary key,
e_name varchar(250),
start_event timestamp,
end_event timestamp,
e_budget decimal(18,4),
e_revune decimal (18,4)
)
;

constraint pk_event primary key (id_event)
CONSTRAINT chk_begin_end_valid CHECK (end_event > begin_event);

create table guests
(
g_name varchar(100),
g_email varchar(100),
id_guest bigint constraint pk_guest primary key
)
;

create table guest_list
(

ticket_number bigint primary key,
frkey_id_guest bigint,
frkey_id_event bigint,
attend bool default false ,
coatcheck bool default false,
coatcheck_num bigint ;

constraint fkey_gl_guest foreign key (frkey_id_guest) references guests(id_guest)
CONSTRAINT fkey_gl_event FOREIGN KEY (frkey_gl_event_id) REFERENCES events(id_event)

create table data_schedule
(frkey_id_event bigint,
frkey_id_loc bigint)

add CONSTRAINT fkey_dats_evnt FOREIGN KEY (frkey_id_event) REFERENCES events(id_event),

add CONSTRAINT fkey_dats_loc FOREIGN KEY (frkey_id_loc) REFERENCES locations(id_location)

Indexes:
    "prk_dats" PRIMARY KEY, btree (frkey_loc_id, frkey_event_id)

Foreign-key constraints:
    "fkey_dats_evnt" FOREIGN KEY (frkey_id_event) REFERENCES events(id_event)

create view guest_contrbutional_value
 as SELECT count(contributions.frkey_id_guest) AS number_of_contributions,

    guests.name,   sum(contributions.c_amount) AS total_contributions

   FROM guests
     JOIN contributions ON guests.id_guest::text = contributions.frkey_id_guest::text

  GROUP BY guests.name;

create view schedule

```
SELECT events.name AS event,   events.begin_event AS event_start,
    events.end_event,    locations.name AS location   FROM data_schedule
     JOIN locations ON data_schedule.frkey_id_loc::text = locations.id_loc::text
     JOIN events ON data_schedule.frkey_id_event::text = events.id_event::text
  ORDER BY events.begin_event;
```