Wyszukiwarka zakładów pogrzebowych
"grobeo"

System gromadzący zakłady pogrzebowe z całej Polski mający zapewnić klientom najlepsze dopasowanie usługi do ich potrzeb.

Grupa projektowa
Tymoteusz Frankiewicz
Filip Mykieta
Jan Wawruszczak

Funkcjonalność projektu

- wyszukiwarka udostępnia listę zakładów pogrzebowych świadczących usługi w wybranym obrządku, umożliwia wybór pochówku na konkretnym cmentarzu i szczegółów ceremonii,
- klient może dostosować wiele parametrów i dowie się, który zakład pogrzebowy oferuje dany pakiet usług w najkorzystniejszej cenie i ma najlepsze rekomendacje od poprzednich klientów,
- dzięki możliwości rejestrowania pogrzebów może służyć jako nekropolia on-line do wyszukiwania grobów swoich krewnych.

Możliwości

- wyszukiwanie zakładów pogrzebowych, które mają w ofercie wybrany przez użytkownika pakiet usług,
- analiza kosztów pogrzebu na podstawie wybranego pakietu,
- przeprowadzenie pogrzebu i dodanie zmarłego do bazy,
- umożliwienie lokalizacji grobu osób zmarłych.

System zarządzania: MySQL (Ver 14.14 Distrib 5.7.26, for Linux (x86_64) using EditLine wrapper)
SOFT: Terminal

Kto wchodzi w interakcje:

- rodzina chcąca pochować zmarłego,
- osoba zainteresowana zaplanowaniem swojego pogrzebu,
- osoba, która chce zadbać o groby swoich bliskich,
- osoba chcąca zlokalizować miejsce spoczynku wybranej osoby

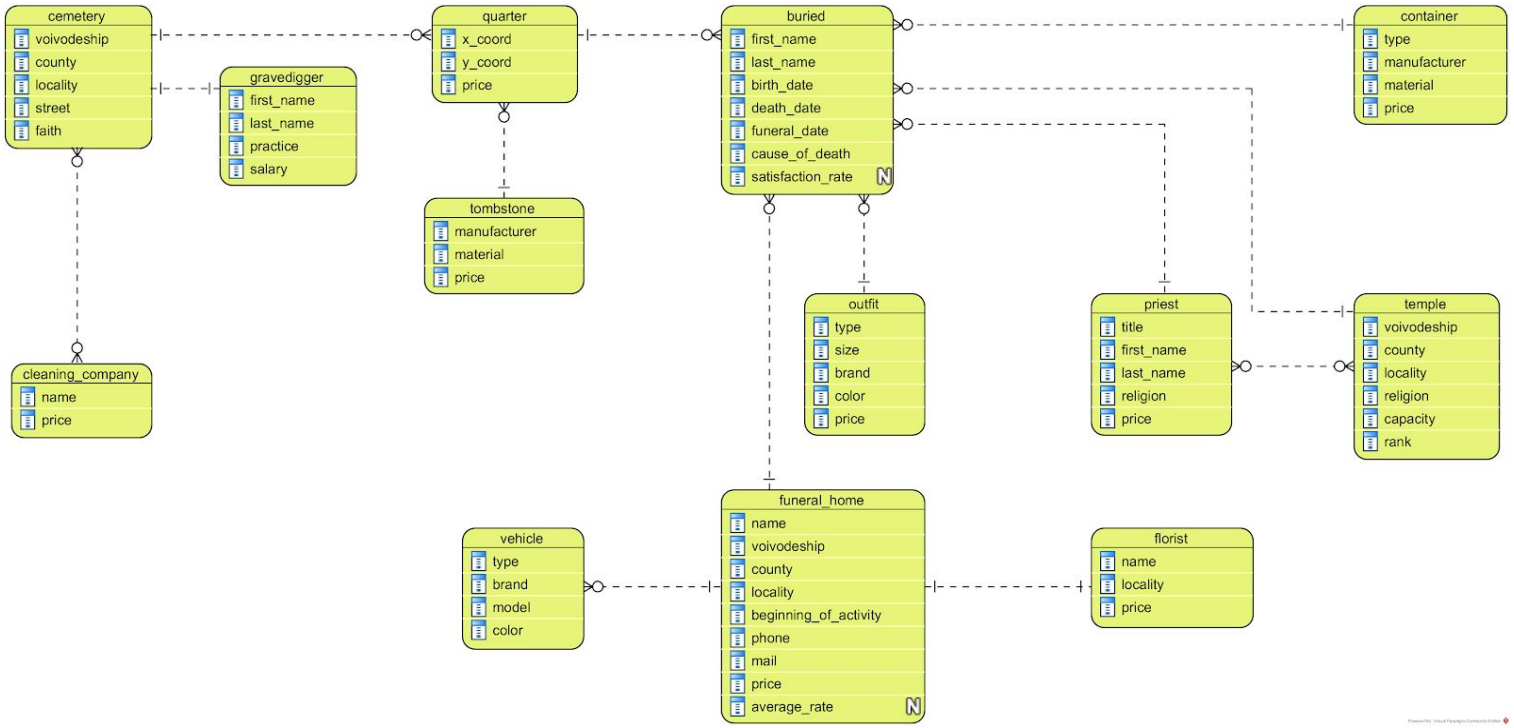
Funkcjonalności:

- ustalenie szczegółów ceremonii pogrzebowej: kapłana, świątyni, oprawy, karawanu, sposobu pochówku, miejsca spoczynku i wyszukanie najlepiej dopasowanej oferty ze współpracującego zakładu pogrzebowego wraz z wyceną,
- możliwość oceny wykonanej usługi w skali od 1 do 5,
- wyszukiwanie grobów swoich krewnych lub znajomych.

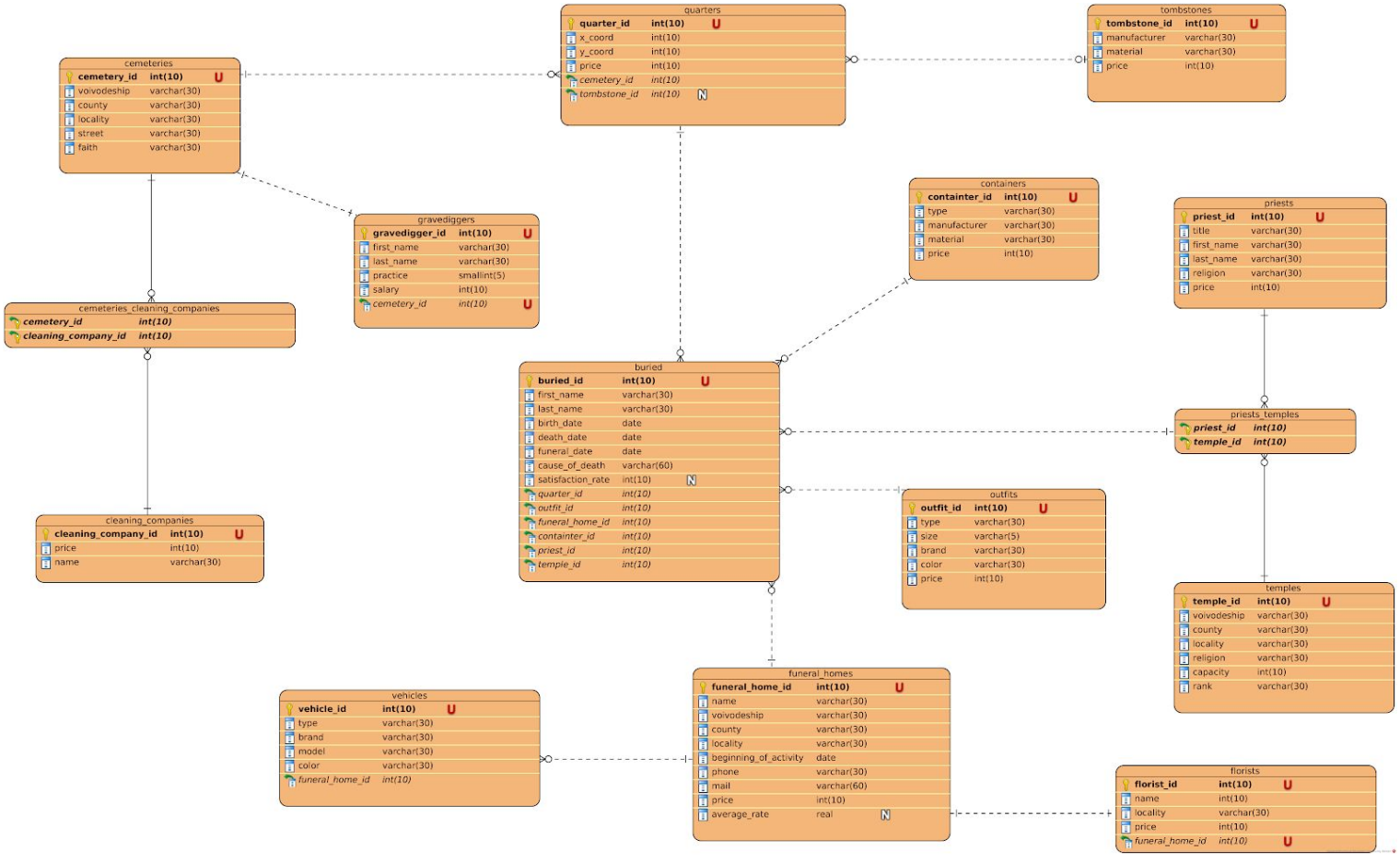
Tabele (model konceptualny)

1. cemetery
    a. voivodeship
    b. county
    c. locality
    d. faith
    e. street
2. quarter
    a. x_coord
    b. y_coord
    c. price
3. buried
    a. first_name
    b. last_name
    c. death_date
    d. birth_date
    e. funeral_date
    f. cause_of_death
    g. satisfaction_rate
4. funeral_home
    a. name
    b. voivodeship
    c. county
    d. locality
    e. beginning_of_activity
    f. phone
    g. mail
    h. price
    i. average_rate
5. priest
    a. title
    b. first_name
    c. last_name
    d. religion
    e. price
6. temple
    a. voivodeship
    b. county
    c. locality
    d. religion
    e. capacity
    f. rank
7. gravedigger
    a. first_name
    b. last_name
    c. practice
    d. salary
8. florist
    a. name
    b. locality
    c. price
9. container
    a. type
    b. manufacturer
    c. material
    d. price
10. tombstone
    a. manfacturer
    b. material
    c. price
11. vehicle
    a. type
    b. brand
    c. model
    d. color
12. outfit
    a. type
    b. size
    c. brand
    d. color
    e. price
13. cleaning company
    a. price
    b. name

Konceptualny diagram ERD



Fizyczny diagram ERD



Transakcje

Nasza baza danych funkcjonuje jako wyszukiwarka zakładów pogrzebowych, które podejmują z nami współpracę i chcą skorzystać z naszego pośrednictwa w dotarciu do klienta. Wobec tego częstym typem transakcji jest wstawienie przez administratorów do bazy nowego zakładu pogrzebowego, który chce się reklamować w *Grobeo* (insert do *funeral_homes*). Każdy zakład posiada flotę pojazdów, co oznacza jednoczesne uzupełnienie tabeli *vehicles*. Nie planujemy kończyć współpracy z danym zakładem pogrzebowym, czyli nie usuwamy zakładów z tabeli. Dodanie pojazdu do tabeli *vehicles* najczęściej będzie związane z zakupem przez zakład nowego karawanu, co może dziać się co jakiś czas, natomiast atrybuty danego pojazdu raczej nie będą modyfikowane. Podobnie każdy zakład współpracuje z jednym dostawcą kwiatów, czyli insert do *florists* związany jest z dodaniem nowego zakładu do bazy. Większość atrybutów w *funeral_homes* rzadko, ale może podlegać zmianom - przypuszczalnie adres, telefon czy e-mail. Zdecydowanie najczęściej będzie wymagał ponownego wyliczenia parametr średniej oceny od klientów, którzy skorzystali z usług zakładu (average_*rate*), każda nowa ocena od użytkownika (*satisfaction_rate* w *buried*) automatycznie wpłynie na tę wartość (działa odpowiedni wyzwalacz).

W naszym zakładzie oferować swe usługi mogą się także kapłani i tutaj zależność wygląda podobnie jak z zakładami pogrzebowymi, to znaczy wstawienie do tabeli *priests* kapłana podejmującego współpracę to operacja wykonywane na porządku dziennym. Kapłan jest przypisany do świątyni na zasadzie relacji *wiele do wielu*, więc dodanie kapłana zmienia rekordy w tabeli pośredniej. Analogiczne operacje dodawania wykonujemy w *temples*. Nie przewidujemy usuwania i modyfikacji zawartości tych dwóch tabel, może z wyjątkiem atrybutu *title* i *salary* w *priests*.

Producenci pojemników i nagrobków udostępniają swoje produkty w tabelach *containers* i *tombstones*. Dodawanie rekordów do tych dwóch tabel będzie na pewno najczęstszą operacją, znacznie rzadsze będzie usuwanie. Krotki w tych encjach nie podlegają modyfikacjom.

Baza oferuje także usługi firm sprzątających. Dodawanie lub usuwanie rekordu oznacza rozpoczęcie bądź zakończenie przez firmę współpracy z nami. Firmy sprzątające powiązane są relacją *wiele do wielu* z cmentarzami i dodanie lub usuwanie firmy powoduje zmiany w tabeli pośredniczącej (podobnie ze strony cmentarzy).

W bazie są także cmentarze, na których zakłady pogrzebowe chowają nieboszczyków. Transakcja dodania nowego cmentarza jest na prośbę klienta, który nie znalazł pożądanego cmentarza w obecnej bazie. Usuwanie cmentarza raczej nie jest przewidziane. Dodanie cmentarza jest nierozerwalnie związane z uzupełnieniem o nowe rekordy tabeli *quarters*, gromadzącej cmentarne kwatery. Każda kwatera jest przypisana do cmentarza, na którym się znajduje. Natomiast cała ewidencja pochówku wykonanego przez zakład rozpoczyna się w tabeli pochowanych *buried*. Bardzo często będą dodawane tam rekordy z informacjami o zmarłym - dacie urodzenia, dacie śmierci, dacie pochówku i przyczynie śmierci. Następnie ze względu na powiązanie z tabelą *quarters* elementem transakcji będzie zmiana informacji o zajętości kwatery, która przypadła zmarłemu, poprzez uzupełnienie pustego dotychczas pola *tombsotne_id* w *quarters* - czyli dodanie na kwaterze nagrobka pieczętuje proces pochówku i wartość tego pola *NOT NULL* oznacza zajętość danej kwatery. Chowanie kolejnej osoby w danej mogile wymaga działania administratora, gdyż domyślnie zabrania tego odpowiedni trigger. Rzadko będą w buried wykonywane operacje *update* i *delete* - więcej o tych przypadkach w sekcji *Wyzwalacze* dokumentu. Należy przy tym pamiętać, że żeby w pełni zwolnić kwaterę i umożliwić ponowne jej wykorzystanie, należy w polu *tombstone_id* umieścić wartość *NULL*.

Mamy jeszcze tabelę *outfits*, gdzie przy transakcji pochówku dodajemy informację o stroju, w którym chowamy zmarłego. Nie usuwamy i nie modyfikujemy jej zawartości. Umieszczamy także powiązanie z tabelami *priests* i *temples*, przypisując do pochowanego parę ksiądz, który prowadził ceremonię, plus świątynia, gdzie ceremonia się odbyła. Wpisujemy także pojemnik, w którym zmarły został pogrzebany, przez powiązanie z *containers*. Zostawiamy także możliwość wystawienia zakładowi pogrzebowemu przez rodzinę całościowej oceny za wykonaną od początku do końca usługę (*satisfaction_rate*). Z *buried* nic nie usuwamy i nic nie modyfikujemy.

Do cmentarza przypisani są grabarze relacją *jeden do jeden*, transakcja nowego cmentarza w bazie oznacza uzupełnienie tabeli *gravediggers* o nowe rekordy. Rzadką operacją jest usunięcie grabarza z bazy (na przykład w wyniku przejścia na emeryturę). Atrybuty grabarzy nie będą modyfikowane.

Powyżej opisane transakcje wykonywane są przez administratorów. Odmiennym typem jest wyszukiwanie w bazie, które udostępniamy użytkownikom. Mamy dwa podstawowe typy wyszukiwania - związane z planowaniem pogrzebu i wyceną lub z poszukiwaniem miejsca pochówku danej osoby.

Widoki

Widoki stosuje się aby mieć łatwy i szybki dostęp do skomplikowanych zapytań. Jest to wirtualna tabela, taka, którą otrzymuje się przy dowolnym zapytaniu, z tą różnicą, że tym razem posiada ona pewien alias, przez który można się odwoływać

Widoki w *GROBEO:*

- Widok przechowujący najpopularniejsze nagrobki (*top_tombstones*):
  CREATE VIEW top_tombstones AS SELECT t.tombstone_id, count(tombstone_id) as num_of_tombstones, t.manufacturer, t.material, t.price FROM cemeteries JOIN quarters USING(cemetery_id) JOIN tombstones t USING(tombstone_id) GROUP BY tombstone_id, manufacturer, material, price ORDER BY count(tombstone_id) DESC;
- Widok przechowujący wszystkich kapłanów (*priests*) i świątynie (*temples*), (priests_and_temples):
  CREATE VIEW priests_and_temples AS SELECT priest_id, title, first_name, last_name, priests.religion, temple_id, rank, locality, capacity, price FROM priests_temples JOIN priests USING(priest_id) JOIN temples USING(temple_id);
- Widok przechowujący najpopularniejsze stroje (*outfits*) (*most_popular_outfits*):
  CREATE VIEW most_popular_outfits AS SELECT outfit_id, count(outfit_id) as num_of_outfits , type, brand, size, color, o.price FROM cemeteries JOIN quarters USING(cemetery_id) JOIN buried USING(quarter_id) JOIN outfits o USING(outfit_id) GROUP BY outfit_id, type, brand, size, color ORDER BY count(outfit_id) DESC;
- Widok przechowujących cmentarze (*cemeteries*) i zakłady pogrzebowe (*funeral_homes*) na nich operujące (*funeral_services_on_cemeteries*):
  CREATE VIEW funeral_services_on_cemeteries AS SELECT DISTINCT f.name, f.price, c.voivodeship, c.county, c.locality, c.street FROM cemeteries c JOIN quarters USING(cemetery_id) JOIN buried USING(quarter_id) JOIN funeral_homes f USING(funeral_home_id);
- Widok przechowujący informacje o całym pogrzebie (pochowanego (*buried*), nagrobek (*tombstones*), pojemnik (*containers*), kapłana (*priests*), świątynię (*temples*) i zakład pogrzebowy (*funeral_homes*)) (*funerals*):
  CREATE VIEW funerals AS select b.first_name, b.last_name, tb.material as tombstone_material, tb.price, c.type, c.material as container_material, p.first_name as priest_name, p.last_name as priest_last_name, t.rank, t.locality, f.name funeral_home FROM buried b JOIN quarters q USING(quarter_id) JOIN tombstones tb using(tombstone_id) JOIN temples t USING(temple_id) JOIN priests p USING(priest_id) JOIN funeral_homes f USING(funeral_home_id) JOIN containers c USING(container_id);

Funkcje

Funkcjom można przekazać pewną liczbę parametrów, ale funkcja nie tylko wykonuje pewne operacje, ale także zwraca obliczony na podstawie przekazanych parametrów wynik.

Funkcje w *GROBEO*:

- Funkcja przyjmująca kwotę, którą chcemy przeznaczyć na kapłana, i oceniająca, czy dany kapłan zgodzi się poprowadzić ceremonię w tej lub niższej cenie.

  ```
  CREATE FUNCTION suggest_priest(price_limit int(10), priest_price int(10)) RETURNS VARCHAR(30) DETERMINISTIC
  BEGIN DECLARE can_afforrd varchar(30);IF price_limit >= priest_price THEN SET can_afforrd = 'STAC CIE';
  ELSEIF price_limit < priest_price THEN SET can_afforrd = 'LEPIEJ ODPUSC';END IF; RETURN (can_afforrd);END;
  ```

- Funkcja oceniająca możliwość wykupienia kwatery na cmentarzu, jeśli dysponujemy zadanym budżetem.

  ```
  CREATE FUNCTION suggest_quarter(price_limit int(10), quarter_price int(10)) RETURNS VARCHAR(30) DETERMINISTIC
  BEGIN  DECLARE can_afforrd varchar(30); IF price_limit >= quarter_price THEN  SET can_afforrd = 'STAC CIE';  ELSEIF price_limit < quarter_price THEN SET can_afforrd = 'LEPIEJ ODPUSC'; END IF; RETURN (can_afforrd);END;
  ```

Wyzwalacze

Wyzwalacz to operacja wykonywana niejako w parze z *insert*, *update* lub *delete* - działa natychmiast i może sprawować kontrolę nad poprawnością danej transakcji (w razie konieczności wyświetlić odpowiedni błąd) lub też uzupełnić ją o dalsze działanie (niewidoczne dla użytkownika, lecz będące integralną częścią operacji).

Wyzwalacze w *GROBEO:*

- Wyzwalacz kontrolujący zajętość kwatery przy insercie do *buried* - zajęte kwatery mają *tombstone_id* NOT NULL. Warunek ten sprawdzany jest także, gdy robimy *update* w *buried* - jeżeli zmarły jest przenoszony do innej kwatery, nie wolno umieścić go w już zajętej.

  ```
  CREATE TRIGGER before_buried_insert_check_quarter_vacancy BEFORE INSERT ON buried FOR EACH ROW BEGIN IF NEW.quarter_id IN (SELECT quarter_id FROM quarters WHERE tombstone_id IS NOT NULL) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Only quarters with tombstone_id NULL are vacant, adding to an already occupied quarter requires administrator interference'; END IF; END;
  CREATE TRIGGER before_buried_update_check_quarter_vacancy BEFORE UPDATE ON buried FOR EACH ROW BEGIN IF NEW.quarter_id IN (SELECT quarter_id FROM quarters WHERE tombstone_id IS NOT NULL) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Only quarters with tombstone_id NULL are vacant, adding to an already occupied quarter requires administrator interference'; END IF; END;
  ```

- Wyzwalacz kontrolujący poprawność zakresu wprowadzanej oceny za pogrzeb przy operacji wstawiania lub modyfikowania - dopuszczamy wyłącznie całkowite wartości z zakresu [1, 5].

  ```
  CREATE TRIGGER before_buried_insert_check_new_rate BEFORE INSERT ON buried FOR EACH ROW BEGIN IF NEW.satisfaction_rate NOT IN (1, 2, 3, 4, 5) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot add or update row: only VALUES IN (1, 2, 3, 4, 5) allowed in satisfaction_rate column'; END IF; END;
  CREATE TRIGGER before_buried_update_check_new_rate BEFORE UPDATE ON buried FOR EACH ROW BEGIN IF NEW.satisfaction_rate NOT IN (1, 2, 3, 4, 5) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot add or update row: only VALUES IN (1, 2, 3, 4, 5) allowed in satisfaction_rate column'; END IF; END;
  ```

- Wyzwalacz kontrolujący poprawność dat w rekordzie wprowadzanym do *buried* - data śmierci nie może być wcześniejsza niż data urodzenia oraz data pochówku nie może być wcześniejsza niż data śmierci. Podobnie przy operacji *update* w *buried* (jest ona wykonywana niezwykle rzadko, na przykład gdy rodzina zorientuje się, że faktyczna data urodzenia krewnego była inna i powinna zostać zmieniona) - zachowane powinny być warunki jak przy *insercie*.

  ```
  CREATE TRIGGER before_buried_insert_check_date_order BEFORE INSERT ON buried FOR EACH ROW BEGIN IF (NEW.death_date < NEW.birth_date OR NEW.funeral_date < NEW.death_date)  THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot add or update
  ```

row: it should be NEW.death_date >= NEW.birth_date AND NEW.funeral_date >= NEW.death_date'; END IF; END;

CREATE TRIGGER before_buried_update_check_date_order BEFORE UPDATE ON buried FOR EACH ROW BEGIN IF (NEW.death_date < NEW.birth_date OR OLD.death_date < NEW.birth_date OR NEW.death_date < OLD.birth_date OR NEW.funeral_date < NEW.death_date OR OLD.funeral_date < NEW.death_date OR NEW.funeral_date < OLD.death_date) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot add or update row: it should be death_date >= birth_date AND funeral_date >= death_date'; END IF; END;

- Wyzwalacz aktualizujący po dodaniu nowej pochowanej osoby (w tym oceny za dany pochówek) średnią ocenę zakładu, który wykonał usługę. Ma też wersję aktualizacji po modyfikacji rekordu w *buried* (ocena może być początkowo *NULL* i dodana potem lub też zmieniona po namyśle klienta); modyfikacja *average_rate* działa także przy usuwaniu rekordu z *buried* (na przykład kiedy naprawiana jest pomyłka administratora).

  CREATE TRIGGER after_buried_insert_calculate_avg_rate AFTER INSERT ON buried FOR EACH ROW BEGIN UPDATE funeral_homes SET average_rate = (SELECT ROUND(SUM(satisfaction_rate)/COUNT(satisfaction_rate), 1) FROM buried WHERE funeral_home_id = NEW.funeral_home_id) WHERE funeral_home_id = NEW.funeral_home_id; END;

  CREATE TRIGGER after_buried_update_calculate_avg_rate AFTER UPDATE ON buried FOR EACH ROW BEGIN UPDATE funeral_homes SET average_rate = (SELECT ROUND(SUM(satisfaction_rate)/COUNT(satisfaction_rate), 1) FROM buried WHERE funeral_home_id = NEW.funeral_home_id) WHERE funeral_home_id = NEW.funeral_home_id; END;

  CREATE TRIGGER after_buried_delete_calculate_avg_rate AFTER DELETE ON buried FOR EACH ROW BEGIN UPDATE funeral_homes SET average_rate = (SELECT ROUND(SUM(satisfaction_rate)/COUNT(satisfaction_rate), 1) FROM buried WHERE funeral_home_id = OLD.funeral_home_id) WHERE funeral_home_id = OLD.funeral_home_id; END;

Przykładowe zapytania

- Liczba pozostałych wolnych kwater na poszczególnych cmentarzach.

  SELECT cemetery_id, count(*) as num_of_available_quarters, voivodeship, county, locality, street, faith FROM quarters JOIN cemeteries USING (cemetery_id) WHERE tombstone_id IS NULL GROUP BY cemetery_id;
- Liczba zajętych kwater na poszczególnych cmentarzach.

  SELECT cemetery_id, count(*) as num_of_occupied_quarters, voivodeship, county, locality, street, faith FROM quarters JOIN cemeteries USING (cemetery_id) WHERE tombstone_id IS NOT NULL GROUP BY cemetery_id;
- Po wybraniu cmentarza następuje wybór kwatery - prezentacja wolnych kwater na cmentarzu o danym *id* (zmieniamy na wybrane *cemetery_id* w *WHERE*).

  SELECT quarter_id, x_coord as x_coordinate, y_coord as y_coordinate, price FROM quarters WHERE cemetery_id = 1 AND tombstone_id IS NULL;
- Wybór zakładu pogrzebowego - liczba wykonanych przez dany zakład usług lub średnia ocena mogą pomóc klientowi w decyzji.

  SELECT funeral_home_id, name, voivodeship, county, locality, beginning_of_activity as founded, phone, mail, price, count(*) as burials_in_GROBEO, average_rate as rating FROM buried JOIN funeral_homes USING (funeral_home_id) GROUP BY funeral_home_id;
- Pokazanie pojazdów, którymi dysponują zakłady pogrzebowe.

  SELECT name, type, brand, model, color, price FROM funeral_homes JOIN vehicles USING(funeral_home_id) ORDER BY price DESC;
- Pokazanie najczęściej wybieranych kapłanów.

  SELECT count(priest_id) as funeral_number, p.first_name, p.last_name, p.price FROM buried JOIN priests p USING(priest_id) GROUP BY p.first_name, p.last_name, p.price ORDER BY count(priest_id) DESC;
- Pokazanie kapłana, który najwięcej zarobił na pogrzebach.

  SELECT p.first_name, p.last_name, p.price * count(priest_id) as earnings FROM buried JOIN priests p USING(priest_id) GROUP BY p.first_name, p.last_name, p.price ORDER BY earnings DESC;

Planowanie pochówku

1. Wybór cmentarza (proszę zapamiętać *id* cmentarza).
   SELECT cemetery_id, count(*) as num_of_available_quarters, voivodeship, county, locality, street, faith FROM quarters JOIN cemeteries USING (cemetery_id) WHERE tombstone_id IS NULL GROUP BY cemetery_id;
2. Wybór kwatery przy zadanym w funkcji *suggest_quarter* budżecie (proszę zapamiętać *id* kwatery).
   SELECT quarter_id, x_coord as x_coordinate, y_coord as y_coordinate, price, suggest_quarter(400,price) FROM quarters WHERE cemetery_id = 1 AND tombstone_id IS NULL;
3. Wybór kapłana i świątyni z wykorzystaniem widoku *priests_and_temples* (proszę zapamiętać parę *id* kapłana i *id* świątyni).
   SELECT * FROM priests_and_temples;
   Możemy też wybierać w wersji, która dla zadanego budżetu zasugeruje, czy klienta stać na daną opcję (wykorzystujemy funkcję *suggest_priest*).
   SELECT *, suggest_priest(500, price) FROM priests_and_temples;
4. Wybranie stroju z wykorzystaniem widoku *most_popular_outfits*, w którym przedstawione są panujące trendy (proszę zapamiętać *id* stroju).
   SELECT * FROM most_popular_outfits;
5. Wybór zakładu pogrzebowego - liczba wykonanych przez dany zakład usług lub średnia ocena mogą pomóc klientowi w decyzji (proszę zapamiętać *id* zakładu).
   SELECT funeral_home_id, name, voivodeship, county, locality, beginning_of_activity as founded, phone, mail, price, count(*) as burials_in_GROBEO, average_rate as rating FROM buried JOIN funeral_homes USING (funeral_home_id) GROUP BY funeral_home_id;
6. Wybranie pojemnika (proszę zapamiętać *id* pojemnika).
   SELECT * FROM containers;
7. Wybór nagrobka z wykorzystaniem widok *top_tombstones*, w którym przedstawione są najczęściej wybierane opcje (proszę zapamiętać *id* nagrobka).
   SELECT * FROM top_tombstones;
8. *Insert* do buried z wykorzystaniem *id* wybranych w całym procesie elementów.
9. Ustawienie na kwaterze nagrobka (uzupełnienie pola *tombstone_id* dla wybranej kwatery w *quarters*).

Wnioski

1. Program Visual Paradigm pozwolił nie tylko wytworzyć diagramy ERD, ale również automatycznie wygenerował na ich podstawie DDL niezbędne do utworzenia bazy danych.
2. Skrypty w Pythonie pozwoliły na pewną automatyzację generowania rekordów wypełniających bazę danych.
3. Obsługa mySQL w terminalu Linuxa okazała się nad wyraz wygodna i praktyczna. Szczególnie użyteczne było dump'owanie skryptu bazy danych za pomocą mysqldump (dump zawiera wszystkie DDL i dane).
4. Korzystanie z kontroli wersji i zdalnego repozytorium gwarantowało bezpieczeństwo na wypadek awarii..
5. Stworzona baza danych jest solidną podstawą pod budowę aplikacji webowej/mobilnej, która ma potencjał, by zapełnić niszę rynkową.