Εξαμηνιαία Εργασία στις Βάσεις Δεδομένων

6ο Εξάμηνο, Ακαδημαϊκή Περίοδος 2020 – 2021

Ομάδα CL

Ονοματεπώνυμο	Αριθμός Μητρώου
Δούσκα Άννα	el16771
Μπούφαλης Οδυσσεύς - Δημήτριος	el18118
Στεφανάκης Γεώργιος	el18436

Στην παρούσα εργασία κληθήκαμε να κατασκευάσουμε μία εφαρμογή που διαχειρίζεται τη δραστηριότητα των πελατών ενός υποθετικού ξενοδοχείου με έμφαση σε στατιστικά που αφορούν τις χρήσεις των διαφόρων υπηρεσιών και χώρων που παρέχονται σε αυτό. Επιπλέον, υποστηρίζεται και η ιχνηλάτηση ατόμων που μπορεί να έχουν μολυνθεί από τον COVID-19, δεδομένου ότι βρέθηκε επιβεβαιωμένο κρούσμα στο ξενοδοχείο. Σε αυτό το μέρος της εργασίας παρουσιάζεται η διαδικασία μετατροπής του προτεινόμενου ER διαγράμματος σε σχεσιακό, η υλοποίησή του στο σύστημα διαχείρισης βάσεων δεδομένων MySQL, η εισαγωγή δοκιμαστικών δεδομένων σε αυτήν καθώς και η ανάπτυξη μίας web εφαρμογής η οποία χειρίζεται τη βάση και είναι φιλική προς τον χρήστη. Η εφαρμογή παρέχει τις εξής λειτουργίες:

- Βασικές λειτουργίες διαχείρισης πελατών
  - Προβολή στοιχείων ενός πελάτη
  - Εισαγωγή – διαγραφή ενός πελάτη
  - Κράτηση δωματίου
  - Εγγραφή πελάτη σε υπηρεσίες που απαιτούν κόστος εγγραφής
  - Προβολή και εξόφληση συνολικών χρεώσεων πελάτη και αποχώρηση από το ξενοδοχείο
- Στατιστικά πελατών για διάφορες ηλικιακές ομάδες
  - Πιο συχνά επισκεπτόμενοι χώροι
  - Πιο συχνά χρησιμοποιούμενες υπηρεσίες
  - Συνολικές χρεώσεις ανά υπηρεσία
- COVID-19
  - Προβολή ιστορικού επισκέψεων ενός επιβεβαιωμένου κρούσματος
  - Αναζήτηση πελατών που είναι πιθανό να έχουν μολυνθεί και εκείνοι

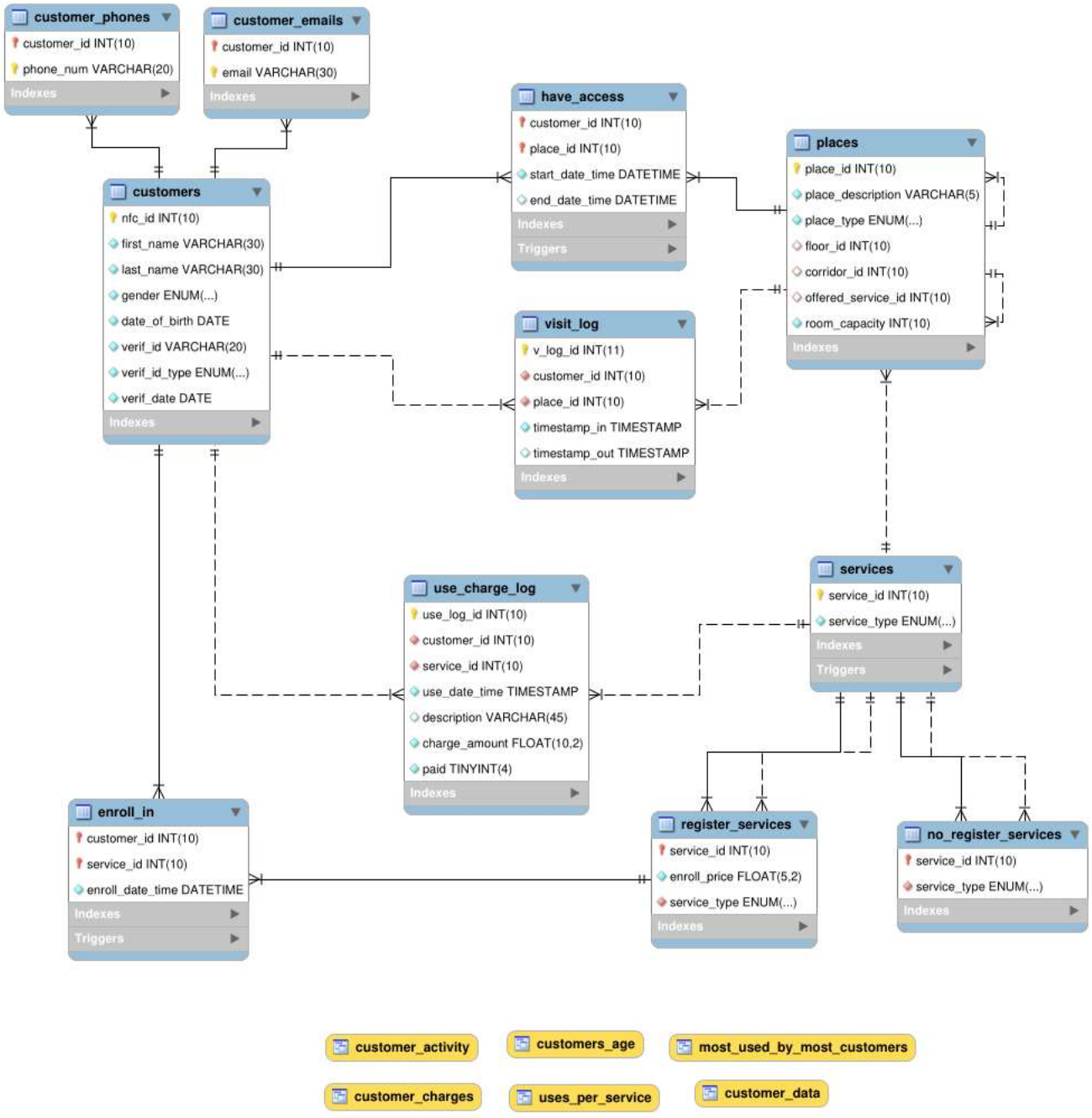
1 Σχεσιακό διάγραμμα

Για την κατασκευή του σχεσιακού διαγράμματος χρησιμοποιήσαμε την προτεινόμενη λύση που μας δόθηκε ύστερα από την παράδοση του πρώτου μέρους της εργασίας, με μερικές μικρές αλλαγές ώστε να εκτελούνται καλύτερα τα ζητούμενα ερωτήματα. Συγκεκριμένα οι αλλαγές που κάναμε είναι οι εξής:

- Η σχέση «*υπηρεσίες* **παρέχονται σε** *χώρους*» έχει cardinality 1:N καθώς θεωρήσαμε τη σύμβαση ότι σε κάθε χώρο του ξενοδοχείου μπορεί να παρέχεται μόνο μία υπηρεσία ενώ μία υπηρεσία μπορεί να παρέχεται σε πολλούς χώρους. Αυτή η αλλαγή στο σχεσιακό διάγραμμα μεταφράζεται ως προσθήκη ενός foreign key στη σχέση places (offered_service_id) το οποίο δείχνει στη σχέση services και σηματοδοτεί την υπηρεσία που παρέχεται σε ένα συγκεκριμένο χώρο.
- Η αδύναμη οντότητα «χρέωση υπηρεσίας» απέκτησε ένα επιπλέον attribute που ονομάζεσαι “paid” το οποίο, για κάθε νέα χρέωση που καταγράφεται, αρχικοποιείται με την τιμή 0 και κατά την αποχώρηση (check out) του πελάτη από το ξενοδοχείο ανανεώνεται στην τιμή 1. Με αυτό τον τρόπο το ιστορικό χρήσεων των πελατών που δεν είναι ενεργοί τη δεδομένη στιγμή στο ξενοδοχείο δεν διαγράφεται και τα στατιστικά στοιχεία που ζητούνται προκύπτουν πιο ρεαλιστικά. Επίσης, εάν ένας πελάτης κάνει πολλαπλές κρατήσεις δωματίων σε διαφορετικά χρονικά διαστήματα, το ιστορικό χρεώσεων που καλείται να εξοφλήσει κατά την αποχώρησή του πρέπει να περιέχει τις χρεώσεις που έκανε μόνο κατά την τελευταία διαμονή του. Συνεπώς με αυτό τον τρόπο στην εφαρμογή εμφανίζονται μόνο οι σωστοί λογαριασμοί, δηλαδή οι μη εξοφλημένοι.

- Το σύνολο οντοτήτων «υπηρεσίες που απαιτούν εγγραφή» έχει ένα επιπλέον πεδίο, το κόστος εγγραφής, το οποίο μας δείχνει πόσο κοστίζει να εγγραφεί κανείς σε μία μη-δωρεάν υπηρεσία. Έτσι, υπάρχει σαφής διαφορά ανάμεσα στις δωρεάν υπηρεσίες που ο πελάτης έχει πρόσβαση από τη στιγμή που θα κάνει κράτηση δωματίου και σε αυτές στις οποίες μπορεί ανά πάσα στιγμή να εγγραφεί πληρώνοντας το αντίστοιχο κόστος εγγραφής.
- Η οντότητα πελάτες απέκτησε και ένα πεδίο “gender” που θα μπορούσε να χρησιμοποιηθεί για την εξέλιξη της εφαρμογής στο μέλλον εμφανίζοντας διαφορετικά στατιστικά, ανάλογα με το φύλο των πελατών.

Παρακάτω φαίνεται το σχεσιακό διάγραμμα το οποίο σχεδιάστηκε με χρήση του MySQL Workbench και αποτέλεσε τη βάση για τη δημιουργία των προτάσεων κατασκευής της ΒΔ.



a. Περιορισμοί

customers – Πελάτες

Σε κάθε πελάτη αντιστοιχεί ένα bracelet με μοναδικό NFC ID το οποίο είναι και το αναγνωριστικό του. Ως εκ τούτου το πεδίο nfc_id είναι το πρωτεύον κλειδί της σχέσης. Επιπλέον περιέχονται και όλα τα attributes (not null) που προτείνονται από το ER διάγραμμα (όνομα, επώνυμο, ημερομηνία γέννησης, φύλο, έγγραφο ταυτοποίησης, αριθμός εγγράφου ταυτοποίησης, είδος εγγράφου ταυτοποίησης, αρχή έκδοσης) πλην τα πεδία “e-mail” και “τηλέφωνο” τα οποία ως multi-valued attributes μεταφράζονται ως δύο νέες N:1 σχέσεις στο σχεσιακό διάγραμμα.

customer_phones – Τηλέφωνα

Κάθε πελάτης μπορεί να έχει 1 ή παραπάνω τηλεφωνικούς αριθμούς. Συνεπώς κατασκευάσαμε αυτή τη N:1 σχέση η οποία έχει ως πρωτεύον κλειδί το σύνθετο κλειδί {customer_id, phone_num} και το attribute customer_id είναι foreign key στο nfc_id της σχέσης customers. Κατά τη διαγραφή ή την ενημέρωση του αναγνωριστικού ενός πελάτη από τη βάση θέλουμε να ενημερώνεται αντίστοιχα και το πεδίο customer_id, άρα επιλέγουμε τους περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

customer_emails – e-mails

Αντίστοιχα με τη σχέση customer_phones. Κάθε πελάτης μπορεί να έχει 1 ή παραπάνω ηλεκτρονικές διευθύνσεις. Συνεπώς κατασκευάσαμε αυτή τη N:1 σχέση η οποία έχει ως πρωτεύον κλειδί το σύνθετο κλειδί {customer_id, email} και το customer_id είναι foreign key στο nfc_id της σχέσης customers. Κατά τη διαγραφή ή την ενημέρωση ενός πελάτη από τη βάση δεν έχει νόημα το customer_id να έχει την παλιά / διαγραμμένη τιμή του αναγνωριστικού του. Συνεπώς, θέλουμε να ενημερώνεται αντίστοιχα και το πεδίο customer_id, άρα επιλέγουμε τους περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

services – Υπηρεσίες

Η σχέση αυτή περιέχει το αναγνωριστικό ID και την περιγραφή κάθε υπηρεσίας. Το πεδίο service_id είναι το primary key της σχέσης και το πεδίο service_type (not null) μπορεί να πάρει μία από τις προδιαγεγραμμένες τιμές RECEPTION, ROOM, BAR, RESTAURANT, CONFERENCE, GYM, SAUNA, HAIR_SALON. Η σχέση συνδέεται με κληρονομικότητα ISA με τις σχέσεις register_services (υπηρεσίες που απαιτούν εγγραφή) και no_register_services (υπηρεσίες που δεν απαιτούν εγγραφή). Έτσι θεωρήσαμε τα πεδία της σχέσης services ως foreign keys των δύο σχέσεων – παιδιών υλοποιώντας με αυτό τον τρόπο τη λογική ότι μία υπηρεσία μπορεί είτε να απαιτεί εγγραφή είτε όχι.

register_services – Υπηρεσίες που απαιτούν εγγραφή

Το service_id, που είναι το αναγνωριστικό κάθε υπηρεσίας, εδώ είναι το primary key της σχέσης αλλά και foreign key στο service_id της σχέσης services. Επιπλέον υπάρχει το πεδίο enroll_price (not null) που μας δείχνει το κόστος εγγραφής για κάθε υπηρεσία και το service_type που κληρονομήθηκε λόγω της σχέσης ISA. Επειδή κατά τη διαγραφή μίας υπηρεσίας θέλουμε να διαγραφεί και η αντίστοιχη τούπλα στον πίνακα αυτό επιλέγουμε περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE και για τα δύο attributes – παιδία.

no_register_services – Υπηρεσίες που δεν απαιτούν εγγραφή

Παρόμοια λογική με εκείνη του register_services. Επιλέγουμε τους ίδιους περιορισμούς ακεραιότητας για τα πεδία service_id και service_type (not null), ON UPDATE CASCADE και ON DELETE CASCADE.

enroll_in – Εγγράφονται σε Υπηρεσίες

Σε αυτή τη σχέση επιλέξαμε το primary key να είναι το σύνθετο κλειδί {customer_id, service_id} με το customer_id να είναι foreign key στο nfc_id της customers ενώ το service_id να είναι foreign key στο service_id της σχέσης register_services. Προφανώς όπως και στα προηγούμενα, επιλέγουμε και για τα δύο foreign keys περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

use_charge_log – Λαμβάνουν Υπηρεσίες με Χρέωση Υπηρεσίας

Η αδύναμη σχέση «Λαμβάνουν Υπηρεσίες» σε συνδυασμό με το αδύναμο σύνολο οντοτήτων «Χρέωση Υπηρεσίας» μετατρέπονται στο σχεσιακό διάγραμμα σε μία N:M σχέση ονόματι use_charge_log η οποία περιέχει ως attributes όλα τα πεδία της αδύναμης οντότητας. Επιπλέον προστίθενται και το customer_id (not null) το οποίο είναι foreign key στο nfc_id της σχέσης customers, το service_id (not null) που είναι foreign key στο service_id της σχέσης services και το use_date_time (not null) που είναι η ημερομηνία χρήσης. Το use_log_id είναι το primary key της σχέσης. Όμοια με τα προηγούμενα επιλέγουμε και για τα δύο foreign keys περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

have_access – Έχουν Πρόσβαση

Κάθε τούπλα σε αυτή τη σχέση μας δείχνει ότι κάποιος πελάτης έχει πρόσβαση σε κάποιον χώρο για κάποιο χρονικό διάστημα. Συνεπώς, το primary key της σχέσης είναι ο συνδυασμός {customer_id, place_id} όπου το customer_id είναι foreign key στο nfc_id της σχέσης customers και το place_id είναι foreign key στο place_id της σχέσης places. Τα πεδία start_date_time (not null) και end_date_time (είναι null όταν η πρόσβαση δεν αφορά κράτηση σε δωμάτιο) περιγράφουν το χρονικό διάστημα που ένας πελάτης έχει πρόσβαση σε αυτό το χώρο. Επιλέγονται και για τα δύο foreign keys περιορισμούς ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

visit_log – Επισκέπτονται

Σε αυτή τη σχέση αποθηκεύεται το ιστορικό επισκέψεων όλων των πελατών. Το primary key είναι το v_log_id καθώς αναφερόμαστε σε σχέση – logger και κάποιος πελάτης μπορεί να επισκεφτεί πολλές φορές τον ίδιο χώρο. Τα πεδία customer_id (not null) και place_id (not null) είναι foreign keys στα nfc_id του customers και place_id του places αντίστοιχα και επιλέγονται και για τα δύο περιορισμοί ακεραιότητας ON UPDATE CASCADE και ON DELETE CASCADE.

places – Χώροι

Σχέση που αποθηκεύει τους χώρους του ξενοδοχείου. Ένας χώρος χαρακτηρίζεται πλήρως από το place_id του το οποίο είναι και το primary key της σχέσης. Το πεδίο place_type (not null) μπορεί να πάρει μία από τις προδιαγεγραμμένες τιμές ELEVATOR, SERVICE, CORRIDOR, FLOOR. Υπάρχουν τρία foreign keys, το floor_id το οποίο μας δείχνει σε ποιόν όροφο βρίσκεται ο χώρος (κάθε όροφος έχει το δικό του place_id), το corridor_id σε ποιό διάδρομο βρίσκεται εάν υπάρχει (κάθε διάδρομος έχει το δικό του place_id) και το offered_service_id που είναι foreign key στο service_id του πίνακα services και μας δείχνει ποια υπηρεσία παρέχεται σε αυτό τον χώρο. Προφανώς και τα τρία αυτά πεδία μπορούν να πάρουν NULL τιμές καθώς δεν είναι απαραίτητο όλοι οι χώροι να έχουν κάποιον αντίστοιχο όροφο, διάδρομο και υπηρεσία. Για τα foreign keys floor_id και corridor_id έχουμε επιλέξει περιορισμούς ON UPDATE CASCADE και ON DELETE RESTRICT καθώς δεν θέλουμε να επιτρέπεται η διαγραφή κάποιου ορόφου / διαδρόμου. Για το offered_service_id επιλέξαμε ON UPDATE CASCADE και ON DELETE SET NULL καθώς η διαγραφή κάποιου service σημαίνει ότι πλέον οι χώροι που το παρείχαν δεν θα παρέχουν πλέον κάποια υπηρεσία.

b. Ευρετήρια

Εν γένει μας ενδιαφέρει η δημιουργία ευρετηρίων σε attributes τα οποία χρησιμοποιούμε περισσότερο σε αναζητήσεις που κάνουμε ώστε να επιταχύνουμε τα ερωτήματα που μας ζητούνται. Συγκεκριμένα καταλήξαμε στα εξής ευρετήρια:

- Στο αναγνωριστικό nfc_id και στην ημερομηνία γέννησης date_of_birth κάθε πελάτη στη σχέση customers καθώς τα χρησιμοποιούμε για τον διαχωρισμό των ζητούμενων στατιστικών ανά ηλικιακή ομάδα στο ερώτημα 11
- Στο αναγνωριστικό υπηρεσίας service_id της σχέσης use_charge_log καθώς χρειάζεται στο ερώτημα 11 για την καταμέτρηση γραμμών με βάση το αναγνωριστικό κάθε υπηρεσίας
- Στα attributes customer_id, place_id και timestamp_in της σχέσης visit_log για την αναζήτηση πιθανών κρουσμάτων COVID-19, όπως ζητείται στο ερώτημα 9

c. Σύστημα διαχείρισης βάσης δεδομένων και γλώσσες προγραμματισμού

Η ανάπτυξη της εφαρμογής έγινε στο λειτουργικό σύστημα Manjaro Linux 21 ενώ το DBMS που επιλέχθηκε για την ανάπτυξη της βάσης είναι το MariaDB 15.1. Χρησιμοποιήσαμε Apache 2.4 για τον web server της ιστοσελίδας και PHP 8 ως κύρια γλώσσα προγραμματισμού της εφαρμογής. Επιπλέον χρησιμοποιήσαμε HTML για να ορίσουμε τη δομή της ιστοσελίδας, CSS (Bootstrap 5) για το styling και Javascript (jQuery Datatables) για να εμφανίζονται τα δεδομένα των πινάκων με τρόπο πιο φιλικό προς τον χρήστη. Τέλος χρησιμοποιήσαμε διάφορες λειτουργίες του MySQL Workbench όπως την κατασκευή βάσης από δεδομένο σχεσιακό διάγραμμα, τη μαζική εισαγωγή τυχαίων δεδομένων από CSV αρχεία που είχαμε παράξει με Python scripts, δημιουργία backup αρχείων κ.α.

d. Βήματα εγκατάστασης

Η εγκατάσταση της εφαρμογής στο localhost αποτελείται από τρία βασικά βήματα:

1. Εγκατάσταση του Apache – MySQL – PHP Stack στον υπολογιστή που μας ενδιαφέρει (XAMPP για μηχανήματα Windows, LAMPP ή χειροκίνητη εγκατάσταση κάθε module για Linux).
2. Δημιουργία του σχήματος asdf_db το οποίο περιλαμβάνει όλα τα στοιχεία της βάσης (relations, triggers, views, δεδομένα). Μπορείτε να χρησιμοποιήσετε το dump αρχείο που περιλαμβάνεται μαζί με την αναφορά το οποίο θα εγκαταστήσει στο DBMS τόσο τη δομή της βάσης όσο και τα δεδομένα των πελατών που έχουμε παράξει ώστε να είναι λειτουργική η εφαρμογή. Επίσης σε αυτό το βήμα ίσως χρειαστεί να ορίσετε τον κωδικό του root χρήστη του DBMS ως “rootroot123” για να είναι επιτυχής η σύνδεση της ιστοσελίδας με το DBMS.
3. Αντιγραφή του κώδικα της εφαρμογής στον κατάλογο που ο Apache κρατάει την ιστοσελίδα που φιλοξενεί το localhost (για Windows συσκευές συνήθως είναι C:\xampp\htdocs ενώ για Linux είναι /srv/httpd).

Για να γίνει πιο κατανοητή η διαδικασία επισυνάπτουμε μαζί με την αναφορά και ένα κείμενο οδηγιών για το πως μπορούν να εγκατασταθούν χειροκίνητα τα απαραίτητα προγράμματα στο λειτουργικό σύστημα που αναπτύχθηκε η εφαρμογή.

2 Κώδικας SQL

Tables

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`customers` (
  `nfc_id` INT(10) NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(30) NOT NULL,
  `last_name` VARCHAR(30) NOT NULL,
  `gender` ENUM('M', 'F', 'NON_BINARY') NOT NULL,
  `date_of_birth` DATE NOT NULL,
  `verif_id` VARCHAR(20) NOT NULL,
  `verif_id_type` ENUM('ID', 'PASSPORT') NOT NULL,
  `verif_date` DATE NOT NULL,
  PRIMARY KEY (`nfc_id`));
```

```
CREATE TABLE IF NOT EXISTS
`asdf_db`.`customer_emails` (
  `customer_id` INT(10) NOT NULL,
  `email` VARCHAR(30) NOT NULL,
  PRIMARY KEY (`email`, `customer_id`),
  CONSTRAINT `fk_customer_emails_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS
`asdf_db`.`customer_phones` (
  `customer_id` INT(10) NOT NULL,
  `phone_num` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`phone_num`, `customer_id`),
  CONSTRAINT `fk_customer_phones_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`services` (
  `service_id` INT(10) NOT NULL AUTO_INCREMENT,
  `service_type` ENUM('RECEPTION', 'ROOM', 'BAR',
'RESTAURANT', 'CONFERENCE', 'GYM', 'SAUNA',
'HAIR_SALON') NOT NULL,
  PRIMARY KEY (`service_id`));
```

```
CREATE TABLE IF NOT EXISTS
`asdf_db`.`register_services` (
  `service_id` INT(10) NOT NULL,
  `enroll_price` FLOAT(5,2) NOT NULL,
  `service_type` ENUM('ROOM', 'CONFERENCE', 'GYM',
'SAUNA') NOT NULL,
  PRIMARY KEY (`service_id`),
  CONSTRAINT `fk_register_services_1`
    FOREIGN KEY (`service_id`)
    REFERENCES `asdf_db`.`services` (`service_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_register_services_2`
    FOREIGN KEY (`service_type`)
    REFERENCES `asdf_db`.`services`
(`service_type`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`enroll_in` (
  `customer_id` INT(10) NOT NULL,
  `service_id` INT(10) NOT NULL,
  `enroll_date_time` DATETIME NOT NULL,
  PRIMARY KEY (`customer_id`, `service_id`),
  CONSTRAINT `fk_enroll_in_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_enroll_in_2`
    FOREIGN KEY (`service_id`)
    REFERENCES `asdf_db`.`register_services`
(`service_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`places` (
  `place_id` INT(10) NOT NULL AUTO_INCREMENT,
  `place_description` VARCHAR(5) NOT NULL,
  `place_type` ENUM('ELEVATOR', 'SERVICE',
'CORRIDOR', 'FLOOR') NOT NULL,
  `floor_id` INT(10) NULL DEFAULT NULL,
  `corridor_id` INT(10) NULL DEFAULT NULL,
  `offered_service_id` INT(10) NULL DEFAULT NULL,
  `room_capacity` INT(10) NOT NULL,
  PRIMARY KEY (`place_id`),
  CONSTRAINT `fk_places_1`
    FOREIGN KEY (`floor_id`)
    REFERENCES `asdf_db`.`places` (`place_id`)
    ON UPDATE CASCADE,
  CONSTRAINT `fk_places_2`
    FOREIGN KEY (`corridor_id`)
    REFERENCES `asdf_db`.`places` (`place_id`)
    ON UPDATE CASCADE,
  CONSTRAINT `fk_places_3`
    FOREIGN KEY (`offered_service_id`)
    REFERENCES `asdf_db`.`services` (`service_id`)
    ON DELETE SET NULL
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`have_access`
(
  `customer_id` INT(10) NOT NULL,
  `place_id` INT(10) NOT NULL,
  `start_date_time` DATETIME NOT NULL,
  `end_date_time` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`customer_id`, `place_id`),
  CONSTRAINT `fk_have_access_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_have_access_2`
    FOREIGN KEY (`place_id`)
    REFERENCES `asdf_db`.`places` (`place_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS
`asdf_db`.`use_charge_log` (
  `use_log_id` INT(10) NOT NULL AUTO_INCREMENT,
  `customer_id` INT(10) NOT NULL,
  `service_id` INT(10) NOT NULL,
  `use_date_time` TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP() ON UPDATE CURRENT_TIMESTAMP(),
  `description` VARCHAR(45) NULL DEFAULT NULL,
  `charge_amount` FLOAT(10,2) NOT NULL,
  `paid` TINYINT(4) NOT NULL DEFAULT 0,
  PRIMARY KEY (`use_log_id`),
  CONSTRAINT `fk_new_table_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_new_table_2`
    FOREIGN KEY (`service_id`)
    REFERENCES `asdf_db`.`services` (`service_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`visit_log` (
  `v_log_id` INT(11) NOT NULL AUTO_INCREMENT,
  `customer_id` INT(10) NOT NULL,
  `place_id` INT(10) NOT NULL,
  `timestamp_in` TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP() ON UPDATE CURRENT_TIMESTAMP(),
  `timestamp_out` TIMESTAMP NULL DEFAULT NULL,
  PRIMARY KEY (`v_log_id`),
  CONSTRAINT `fk_visit_log_1`
    FOREIGN KEY (`customer_id`)
    REFERENCES `asdf_db`.`customers` (`nfc_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_visit_log_2`
    FOREIGN KEY (`place_id`)
    REFERENCES `asdf_db`.`places` (`place_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `asdf_db`.`no_register_services` (
  `service_id` INT(10) NOT NULL,
  `service_type` ENUM('RECEPTION', 'BAR', 'RESTAURANT', 'HAIR_SALON') NOT NULL,
  PRIMARY KEY (`service_id`),
  CONSTRAINT `fk_no_register_services_1`
    FOREIGN KEY (`service_id`)
    REFERENCES `asdf_db`.`services` (`service_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_no_register_services_2`
    FOREIGN KEY (`service_type`)
    REFERENCES `asdf_db`.`services` (`service_type`)
    ON DELETE CASCADE
    ON UPDATE CASCADE);
```

Indexes

```
CREATE INDEX `customers_nfc_id_idx` ON `asdf_db`.`customers` (`nfc_id` ASC);
CREATE INDEX `customers_birthdate_idx` ON `asdf_db`.`customers` (`date_of_birth` ASC);
CREATE INDEX `fk_use_charge_log_1_idx` ON `asdf_db`.`use_charge_log` (`customer_id` ASC);
CREATE INDEX `fk_use_charge_log_2_idx` ON `asdf_db`.`use_charge_log` (`service_id` ASC);
CREATE INDEX `fk_visit_log_1_idx` ON `asdf_db`.`visit_log` (`customer_id` ASC);
CREATE INDEX `fk_visit_log_2_idx` ON `asdf_db`.`visit_log` (`place_id` ASC);
CREATE INDEX `visit_log_time_in_idx` ON `asdf_db`.`visit_log` (`timestamp_in` ASC);
```

Views

```
CREATE VIEW `customer_data` AS
    SELECT
        `c`.`nfc_id` AS `nfc_id`,
        `c`.`first_name` AS `first_name`,
        `c`.`last_name` AS `last_name`,
        `c`.`verif_id` AS `verif_id`,
        `c`.`date_of_birth` AS `birth`,
        TIMESTAMPDIFF(YEAR,
            `c`.`date_of_birth`,
            CURDATE()) AS `age`,
        `c`.`gender` AS `gender`,
        (SELECT
                `p`.`place_description`
            FROM
                ((`customers` `cu`
                JOIN `have_access` `h` ON (`cu`.`nfc_id` = `h`.`customer_id`))
                JOIN `places` `p` ON (`h`.`place_id` = `p`.`place_id`))
            WHERE
                `c`.`nfc_id` = `cu`.`nfc_id`
                    AND `p`.`offered_service_id` = 2) AS `reserved`
    FROM
        `customers` `c`

CREATE VIEW `uses_per_service` AS
    SELECT
        `s`.`service_id` AS `service_id`,
        `s`.`service_type` AS `service_type`,
        COUNT(0) AS `total_uses`
    FROM
        (`use_charge_log` `u`
        JOIN `services` `s` ON (`u`.`service_id` = `s`.`service_id`))
    WHERE
        `u`.`service_id` <> 2
    GROUP BY `u`.`service_id`


CREATE VIEW `asdf_db`.`customer_activity` AS
    SELECT
        `v`.`customer_id` AS `customer_id`,
        `v`.`timestamp_in` AS `timestamp_in`,
        `v`.`timestamp_out` AS `timestamp_out`,
        `p`.`place_id` AS `place_id`,
        `p`.`place_description` AS `place_description`
    FROM
        (`asdf_db`.`visit_log` `v`
        JOIN `asdf_db`.`places` `p` ON (`v`.`place_id` = `p`.`place_id`))
    ORDER BY `v`.`timestamp_in`
```

```
CREATE VIEW `asdf_db`.`customer_charges` AS
    (SELECT
        `c`.`nfc_id` AS `nfc_id`,
        `e`.`enroll_date_time` AS `date`,
        'SERVICE ENROLL' AS `type`,
        `r`.`enroll_price` AS `charge`,
        `r`.`service_type` AS `description`
    FROM
        ((`asdf_db`.`customers` `c`
        JOIN `asdf_db`.`enroll_in` `e` ON (`c`.`nfc_id` = `e`.`customer_id`))
        JOIN `asdf_db`.`register_services` `r` ON (`e`.`service_id` = `r`.`service_id`))
    WHERE
        `e`.`service_id` <> 2) UNION (SELECT
        `c`.`nfc_id` AS `nfc_id`,
        `u`.`use_date_time` AS `date`,
        REPLACE(CONCAT(`s`.`service_type`, ' BILL'),
            '_',
            ' ') AS `type`,
        `u`.`charge_amount` AS `charge`,
        `u`.`description` AS `description`
    FROM
        ((`asdf_db`.`customers` `c`
        JOIN `asdf_db`.`use_charge_log` `u` ON (`c`.`nfc_id` = `u`.`customer_id`))
        JOIN `asdf_db`.`services` `s` ON (`u`.`service_id` = `s`.`service_id`))
    WHERE
        `u`.`service_id` <> 2 AND `u`.`paid` = 0) UNION (SELECT
        `c`.`nfc_id` AS `nfc_id`,
        `u`.`use_date_time` AS `date`,
        'RESERVATION' AS `type`,
        `u`.`charge_amount` AS `charge`,
        `u`.`description` AS `description`
    FROM
        ((`asdf_db`.`customers` `c`
        JOIN `asdf_db`.`use_charge_log` `u` ON (`c`.`nfc_id` = `u`.`customer_id`))
        JOIN `asdf_db`.`services` `s` ON (`u`.`service_id` = `s`.`service_id`))
    WHERE
        `u`.`service_id` = 2 AND `u`.`paid` = 0) ORDER BY `date`


CREATE VIEW `asdf_db`.`most_used_by_most_customers` AS
    SELECT
        `s`.`service_id` AS `service_id`,
        `s`.`service_type` AS `service_type`,
        COUNT(DISTINCT `u`.`customer_id`, `u`.`service_id`) AS `count(distinct u.customer_id,
u.service_id)`
    FROM
        (`asdf_db`.`use_charge_log` `u`
        JOIN `asdf_db`.`services` `s` ON (`u`.`service_id` = `s`.`service_id`))
    WHERE
        `s`.`service_id` <> 2
    GROUP BY `s`.`service_id`
```

Triggers

Για να διατηρείται η ακεραιότητα των δεδομένων και να ενημερώνονται κατάλληλα όλες οι σχέσεις της ΒΔ κατά την εισαγωγή – διαγραφή σε κάποια από αυτές κατασκευάσαμε τα εξής triggers:

- Κατά την εισαγωγή μίας εγγραφής στη σχέση enroll_in
  - Εάν αυτή η εγγραφή αφορά κράτηση σε δωμάτιο τότε ενημερώνεται ο πίνακας have_access ώστε ο πελάτης να έχει πρόσβαση σε όλους τους χώρους που παρέχουν δωρεάν υπηρεσίες
  - Εάν αφορά εγγραφή σε κάποια υπηρεσία τότε ενημερώνεται ο πίνακας have_access ώστε ο πελάτης να έχει πρόσβαση σε όλους τους χώρους που παρέχουν αυτή την υπηρεσία
- Κατά τη διαγραφή μίας εγγραφής στη σχέση enroll_in (αποχώρηση από το ξενοδοχείο)
  - Εξόφληση όλων των χρεώσεων στη σχέση use_charge_log
  - Διαγραφή όλων των εγγραφών που αφορούν αυτόν τον πελάτη στη σχέση have_access
- Κατά την εισαγωγή μίας εγγραφής στη σχέση have_access (κράτηση δωματίου για ένα χρονικό διάστημα)
  - Εισαγωγή λογαριασμού κράτησης δωματίου στον πίνακα use_charge_log με χρέωση που υπολογίζεται ως # μέρες κράτησης $\times$ κόστος διαμονής για ένα βράδυ.

```
CREATE TRIGGER `asdf_db`.`enroll_in_AFTER_INSERT`
AFTER INSERT ON `asdf_db`.`enroll_in`
FOR EACH ROW
BEGIN
	-- new service enrolled
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, p.place_id, NEW.enroll_date_time, NULL
		FROM places p
		WHERE p.offered_service_id = NEW.service_id AND NEW.service_id <> 2;

	-- reception
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 26, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	-- elevators
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 11, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 12, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 13, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 14, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, 15, NEW.enroll_date_time, NULL
		WHERE NEW.service_id = 2;

	-- bars
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, p.place_id, NEW.enroll_date_time, NULL
		FROM places p
		WHERE p.offered_service_id = 3 AND NEW.service_id = 2;

	-- restaurants
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, p.place_id, NEW.enroll_date_time, NULL
		FROM places p
		WHERE p.offered_service_id = 4 AND NEW.service_id = 2;

	-- hair salon
	INSERT INTO have_access (customer_id, place_id, start_date_time, end_date_time)
		SELECT NEW.customer_id, p.place_id, NEW.enroll_date_time, NULL
		FROM places p
		WHERE p.offered_service_id = 8 AND NEW.service_id = 2;

END

CREATE TRIGGER `asdf_db`.`have_access_AFTER_INSERT`
AFTER INSERT ON `asdf_db`.`have_access`
FOR EACH ROW
BEGIN
	INSERT INTO use_charge_log (customer_id, service_id, use_date_time, description, charge_amount)
		SELECT NEW.customer_id, 2, NEW.start_date_time, 'ROOM RESERVATION',
			DATEDIFF(NEW.end_date_time, NEW.start_date_time) * s.enroll_price
		FROM register_services s, places p
		WHERE s.service_id = 2 AND NEW.place_id = p.place_id AND p.offered_service_id = 2;

END
```

```
CREATE TRIGGER `asdf_db`.`services_BEFORE_DELETE`
BEFORE DELETE ON `asdf_db`.`services`
FOR EACH ROW
BEGIN
        DELETE FROM have_access
        WHERE have_access.place_id IN
        (SELECT p.place_id
          FROM places p
          WHERE p.offered_service_id = OLD.service_id);

        UPDATE places
        SET place_description = 'EMPTY'
        WHERE offered_service_id = OLD.service_id;
END

CREATE TRIGGER `asdf_db`.`enroll_in_AFTER_DELETE`
AFTER DELETE ON `asdf_db`.`enroll_in`
FOR EACH ROW
BEGIN
    UPDATE use_charge_log
    SET paid = 1
    WHERE OLD.customer_id = customer_id;

    DELETE FROM have_access
    WHERE customer_id = OLD.customer_id;
END
```

Ιστορικό επισκέψεων και COVID-19 (ερωτήματα 9, 10)

```
$activity_query = "select concat(timestamp_in, ' / ', timestamp_out), place_description
                   from customer_activity
                   where customer_id = '$customer'
                   and timestamp_in between date_add('$datetime', INTERVAL -5 DAY) and '$date'";
$infections_query = "select concat(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'), p.place_description,
                     concat(v.timestamp_in, ' / ', v.timestamp_out)
                     from visit_log v join customers c on v.customer_id = c.nfc_id
                     join places p on p.place_id = v.place_id
                     where v.place_id in
                       (select v1.place_id from visit_log v1
                       where v.timestamp_in between v1.timestamp_in and date_add(v1.timestamp_out, INTERVAL 1 HOUR)
                       and v1.customer_id = '$customer'
                       and v1.timestamp_in between date_add('$datetime', INTERVAL -5 DAY) AND '$datetime')
                       and v.customer_id <> '$customer' and v.timestamp_in between date_add('$datetime', INTERVAL -5 DAY) AND '$datetime'";
```

Πιο πολυσύχναστοι χώροι ανά ηλικιακή ομάδα (ερώτημα 11)

```
select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age between 20 and 40
and v.timestamp_in between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;

select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age between 41 and 60
and v.timestamp_in between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;

select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age >= 61
and v.timestamp_in between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;

select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age between 20 and 40
and v.timestamp_in between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;

select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age between 41 and 60
and v.timestamp_in between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;

select p.place_description,p.place_id, count(*) as number_of_visits
from visit_log v join places p on v.place_id = p.place_id join customers_age c on c.nfc_id = v.customer_id
where p.offered_service_id <> 2 and c.Age >= 61
and v.timestamp_in between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by p.place_id
order by number_of_visits desc
limit 10;
```

Συχνότερα χρησιμοποιούμενες υπηρεσίες ανά ηλικιακή ομάδα (ερώτημα 11)

```
select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age between 20 and 40
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by number_of_uses desc;

select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age between 41 and 60
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by number_of_uses desc;

select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age >= 61
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by number_of_uses desc;

select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age between 20 and 40
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by number_of_uses desc;

select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age between 41 and 60
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by number_of_uses desc;

select s.service_type,s.service_id, count(*) as number_of_uses
from use_charge_log u join services s on u.service_id = s.service_id
join customers_age c on c.nfc_id = u.customer_id
where u.service_id <> 2 and c.Age >= 61
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by number_of_uses desc;
```

Υπηρεσίες που χρησιμοποιούνται από τους περισσότερους πελάτες ανά ηλικιακή ομάδα (ερώτημα 11)

```
select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age between 20 and 40
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by most_used desc;

select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age between 41 and 60
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by most_used desc;

select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age >= 61
and u.use_date_time between date_add(curdate(), INTERVAL -1 MONTH) and curdate()
group by s.service_id
order by most_used desc;

select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age between 20 and 40
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by most_used desc;

select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age between 41 and 60
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by most_used desc;

select s.service_type, count(distinct u.customer_id, u.service_id) as most_used
from use_charge_log u join customers_age c on u.customer_id = c.nfc_id
join services s on u.service_id = s.service_id
where u.service_id <> 2 and c.age >= 61
and u.use_date_time between date_add(curdate(), INTERVAL -1 YEAR) and curdate()
group by s.service_id
order by most_used desc;
```

Ιστορικό επισκέψεων με βάση πολλαπλά κριτήρια (ερώτημα 7)

```
SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE p.offered_service_id = '$service' AND u.charge_amount <= '$charge'
AND v.timestamp_in BETWEEN '$start' AND '$end';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE p.offered_service_id = '$service' AND v.timestamp_in BETWEEN '$start' AND '$end';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE p.offered_service_id = '$service' AND u.charge_amount <= '$charge';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE p.offered_service_id = '$service';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE u.charge_amount <= '$charge' AND v.timestamp_in BETWEEN '$start' AND '$end';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE v.timestamp_in BETWEEN '$start' AND '$end';

SELECT CONCAT(v.timestamp_in, ' / ', v.timestamp_out),
CONCAT(c.first_name, ' ', c.last_name, ' (', c.verif_id, ')'),
p.place_description, u.charge_amount
FROM customers c JOIN visit_log v ON c.nfc_id = v.customer_id
JOIN places p ON v.place_id = p.place_id
JOIN use_charge_log u ON u.use_date_time = v.timestamp_out
WHERE u.charge_amount <= '$charge';
```

3 Αρχεία τεχνοδιαμόρφωσης

Στον φάκελο asdf_app-master που έχουμε επισυνάψει στην αναφορά συμπεριλαμβάνονται όλα τα αρχεία πηγαίου κώδικα της εφαρμογής καθώς επίσης και ο φάκελος database που περιέχει το πιο πρόσφατο αντίγραφο της ΒΔ Dump20210615.sql, το MySQL Workbench Model αρχείο relational.mwb και τον φάκελο init που περιέχει όλα τα δεδομένα αρχικοποίησης της ΒΔ και τα δεδομένα πελατών που παράξαμε με τα αντίστοιχα Python scripts. Για την ανακατασκευή της ΒΔ σε κάποιον άλλον υπολογιστή αρκεί η εκτέλεση του αρχείου Dump20210615.sql. Επιπλέον, στο αρχείο db_connect.php βρίσκονται τα στοιχεία με τα οποία η εφαρμογή συνδέεται στο DBMS. Για τη σωστή λειτουργία της πρέπει το όνομα χρήστη και ο κωδικός να είναι σε συμφωνία με εκείνα που έχει ορίσει ο χρήστης κατά την εγκατάσταση του DBMS.

Το GitHub Repository της εφαρμογής βρίσκεται στον σύνδεσμο: https://github.com/stefanaki/asdf_app