Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

PROJECT - ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Αγγέλης Γιώργος	Βαρσαμής Βασίλης	Κουλάκος Αλέξανδρος
03118030	03118033	03118144

Ιούνιος 2021

Περιεχόμενα

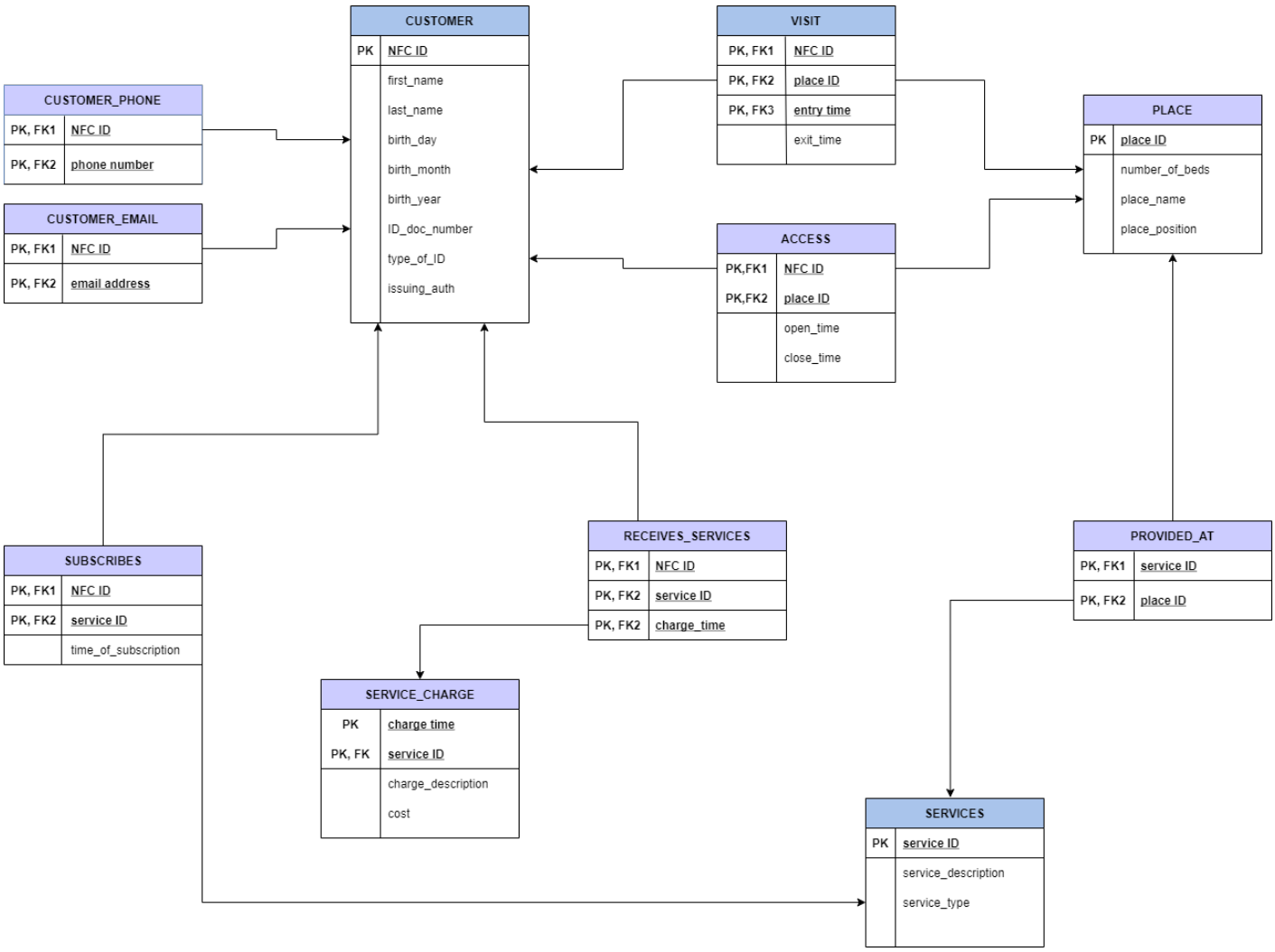
Εισαγωγή

Στο πλαίσιο του μαθήματος «Βάσεις Δεδομένων» του Ακαδημαϊκού έτους 2020-21 κληθήκαμε να δημιουργήσουμε ένα σύστημα αποθήκευσης, διαχείρισης και ανάλυσης πληροφοριών για τη ξενοδοχειακή μονάδα ASDF Palace. Το σύστημα αυτό αφορά δεδομένα που σχετίζονται με τη λειτουργία του ξενοδοχείου, κυρίως ως προς τις προσφερόμενες υπηρεσίες και τους πελάτες, υιοθετώντας συγκεκριμένα υγειονομικά πρωτόκολλα αναγκαία σε περίοδο πανδημίας.

1.

Σχεσιακό Διάγραμμα

Σχεδιάσαμε και παραθέτουμε παρακάτω το ζητούμενο Σχεσιακό Διάγραμμα με βάση το προτεινόμενο ER Διάγραμμα που αντιστοιχεί στην απλή λύση. Για τη σχεδίαση χρησιμοποιήσαμε το online εργαλείο draw.io .



1. a.

Παραδοχές - Παρατηρήσεις

- Η βάση δεδομένων που αναπτύξαμε αφορά την περίοδο Μαίου – Ιουνίου του 2021 για το ξενοδοχείο *ASDF Palace*.

- Για μια υπηρεσία το *charge_description* μπορεί να ποικίλλει (π.χ. το εστιατόριο προσφέρει πρωινό, μεσημεριανό και βραδινό, ενώ στη σάουνα το κόστος διαμορφώνεται με βάση τη διάρκεια χρήσης).

- Ο πίνακας *SERVICE_CHARGE* περιλαμβάνει όλες τις χρεώσεις που σημειώθηκαν στο ξενοδοχείο για τους μήνες Μάιο και Ιούνιο. Ο πίνακας *RECEIVE_SERVICES* περιλαμβάνει τις χρεώσεις που καταλογίστηκαν στους 25 πελάτες του ξενοδοχείου που έχουμε επιλέξει για την οικονομία της άσκησης. Προφανώς, λόγω του FK constraint, δεν μπορεί να υπάρχει χρέωση από υπηρεσία στη *RECEIVE_SERVICES* η οποία δεν εμφανίζεται στη *SERVICE_CHARGE*.

- Οι χρεώσεις των δωματίων είναι συνολικές και αποπληρώνονται κατά το *check*out του πελάτη. Οι υπόλοιπες χρεώσεις είναι per use και αποπληρώνονται μόλις ο πελάτης ολοκληρώσει τη χρήση κάθε υπηρεσίας (π.χ. της σάουνας).

- Προφανώς, η εγγραφή σε μια υπηρεσία προηγείται της χρήσης της, οπότε η ημερομηνία *time_of_sub* είναι παλαιότερη από τη *charge_time*.

- Η χρήση μιας υπηρεσίας προϋποθέτει την εγγραφή σε αυτή, δηλαδή οι υπηρεσίες που χρησιμοποιεί κάποιος πελάτης πρέπει να είναι υποσύνολο (όχι απαραίτητα γνήσιο) των υπηρεσιών στις οποίες έχει γραφτεί. Δηλαδή, για έναν πελάτη τα αναγνωριστικά των υπηρεσιών που χρησιμοποιεί στη *RECEIVE_SERVICES* πρέπει να εμφανίζονται και στη σχέση *SUBSCRIBES* για τον ίδιο πελάτη, εφόσον πρόκειται για υπηρεσία που απαιτεί εγγραφή.

- Για λόγους απλότητας υποθέτουμε ότι κάθε πελάτης εγγράφεται ταυτόχρονα στις υπηρεσίες που τον ενδιαφέρουν, δηλαδή για κάποιο *NFC_ID* το *time_of_sub* διατηρεί σταθερή τιμή.

- Τα χαρακτηριστικά *open_time* και *close_time* στη σχέση *ACCESS* αναφέρονται στην ώρα που ανοίγει και κλείνει, αντίστοιχα, ο χώρος στον οποίο παρέχεται μια υπηρεσία.

- Για να μοντελοποιήσουμε τις «Υπηρεσίες που απαιτούν εγγραφή» και τις «Υπηρεσίες που δεν απαιτούν εγγραφή», επιλέξαμε το σχεσιακό διάγραμμα να περιλαμβάνει τη σχέση «Υπηρεσίες» με ένα επιπλέον χαρακτηριστικό, το service_type, το οποίο λόγω του check constraint λαμβάνει αυστηρά μία από τις τιμές subscription ή no_subscription. Πράγματι, αυτή είναι μια καλή και κομψή σχεδιαστική πρακτική, διότι κάθε υπηρεσία είτε απαιτεί είτε δεν απαιτεί εγγραφή, οπότε δεν υπάρχουν NULL εγγραφές ή απώλειες πληροφορίας.

- Το σύνολο συσχετίσεων «Λαμβάνουν υπηρεσίες», παρ' ότι συνδέει το αδύναμο σύνολο οντοτήτων «Χρέωση υπηρεσίας» με το σύνολο οντοτήτων «Υπηρεσίες», πρέπει να μοντελοποιηθεί ξεχωριστά σαν πίνακας. Ναι μεν το *Service ID* έχει πλέον ενσωματωθεί στον πίνακα της σχέσης του αδύναμου συνόλου οντοτήτων, δεν πρέπει, όμως, να ξεχνάμε ότι η συσχέτιση «Λαμβάνουν υπηρεσίες» είναι τριαδική, διότι σε αυτή συμμετέχουν και οι «Πελάτες» και μάλιστα το cardinality ανάμεσα σε «Πελάτες» και «Υπηρεσίες» είναι *Many to Many*, οπότε το σύνολο συσχετίσεων «Λαμβάνουν υπηρεσίες» δεν πρέπει να παραλειφθεί, προκειμένου να μην υπάρξουν απώλειες πληροφορίας.

- Παρατηρούμε ότι στη σχέση «Επισκέπτονται» δεν είναι σωστό να θέσουμε ως primary key μόνο το NFC ID και το place ID, διότι μπορεί ένας πελάτης να επισκεφθεί τον ίδιο χώρο (π.χ. τη σάουνα 2) δύο διαφορετικές μέρες. Έτσι, προσθέτουμε και την ημερομηνία εισόδου στο χώρο ως μέρος του primary key.

- Τα πλειότιμα χαρακτηριστικά customer email και customer phone μοντελοποιούνται στο σχεσιακό διάγραμμα σαν ανεξάρτητες σχέσεις.

Περιορισμοί

Για τη βάση δεδομένων έχουμε επιλέξει διάφορους περιορισμούς ακεραιότητας οι οποίοι μας βοηθούν να έχουμε έλεγχο της πληροφορίας σε διάφορες αλλαγές. Οι περιορισμοί ακεραιότητας συνοδεύονται από διάφορες παραδοχές. Αναλυτικά, ως διαχειριστές και σχεδιαστές της βάσης, επιλέγουμε να διαγράφουμε έναν πελάτη από το ξενοδοχείο τουλάχιστον δύο εβδομάδες μετά την ημερομηνία check-out και με αυτό τον τρόπο διασφαλίζουμε ότι η διαγραφή αυτή μπορεί να μεταφερθεί σε όλες τις σχέσεις που εχουν το NFC ID ως foreign key χωρίς να τίθεται επιδημιολογικός κίνδυνος.

Ομοίως, αν κάποιος πελάτης χάσει το βραχιολάκι του και χρειαστεί ενημέρωση του NFC ID, η ενημέρωση αυτή μεταφέρεται σε όλες τις σχέσεις που έχουν ως foreign key το NFC ID. Πρακτικά, λοιπόν, σε αυτές τις σχέσεις έχει επιλεγεί το ON DELETE CASCADE ON UPDATE CASCADE. Δεν ισχύει, όμως, κάτι παρόμοιο σε περίπτωση που επιθυμούμε να διαγράψουμε ή να ενημερώσουμε μια υπηρεσία ή ένα χώρο, διότι υπάρχει πολύτιμη επιδημιολογική πληροφορία στη σχέση «Επισκέπτονται» και πολύτιμη στατιστική πληροφορία στη σχέση «Χρεώσεις υπηρεσιών». Σε αυτή την περίπτωση υπάρχει η default ρύθμιση της βάσης (ON DELETE NO ACTION ON UPDATE NO ACTION) η οποία δε μας επιτρέπει να διαγράψουμε κάποια τιμή primary key από μία σχέση πατέρα (εν προκειμένω, «Υπηρεσίες» και «Χώροι») που έχει εμφανίσεις σε σχέσεις παιδιά (εν προκειμένω, «Λαμβάνουν υπηρεσίες» και «Χρεώσεις υπηρεσίας»), έτσι ώστε να εκλείπουν οι ορφανές εγγραφές. Συνεπώς, όπως φαίνεται στην οθόνη, όταν επιχειρήσουμε DELETE ή UPDATE σε κάποια από τις παραπάνω γονικές σχέσεις, το DBMS χτυπάει error.

Όσον αφορά τα triggers, πρέπει να εξασφαλίζεται ότι σε περίπτωση εισαγωγής μιας νέας τούπλας στη σχέση «Εγγράφεται σε υπηρεσίες» το DBMS θα βρίσκει αυτόματα όλα τα place ID από τη σχέση «Παρέχονται σε» που συσχετίζονται με το εισαχθέν service ID. Στη συνέχεια, θα προσθέτει στη σχέση «Έχει πρόσβαση» τις τούπλες που προκύπτουν από την τιμή του νέου NFC ID με όλα τα place ID που συλλέχθηκαν παραπάνω. Αντίστοιχα πράγματα πρέπει να συμβούν και στην περίπτωση όπου διαγράφουμε μια τούπλα από τη σχέση «Εγγράφεται σε υπηρεσίες».

Αν προσθέσουμε μια τούπλα στη σχέση «Παρέχονται σε», δηλαδή πρακτικά αν ελευθερωθεί ένα παραπάνω δωμάτιο για να φιλοξενεί μια υπηρεσία, τότε το DBMS πρέπει αυτόματα να βρίσκει τα NFC ID όλων των πελατών που έχουν εγγραφεί στη νέα υπηρεσία και να προσθέτει στη σχέση «Έχει

πρόσβαση» τις τούπλες με τα εν λόγω NFC ID και το νέο place ID. Αντίστοιχα πράγματα πρέπει να συμβούν και στην περίπτωση όπου διαγράφουμε μια τούπλα από τη σχέση «Παρέχονται σε».

1. b.

INDICES

Προκειμένου να βελτιώσουμε στοχευμένα την ταχύτητα κάποιων ερωτημάτων για το Data Base δημιουργήσαμε τα εξής Indices:

```
********** INDICES **********
CREATE INDEX charge_time_idx ON service_charge(charge_time)
CREATE INDEX cost_idx ON service_charge(cost)
CREATE INDEX entry_time_idx ON visit(entry_time)
```

αφού παρατηρούμε ότι καλούνται πολλές φορές τα:

- **charge_time** του service_charge /7
- **cost** του service_charge /7
- **entry_time** του visit /9-10 ιχνηλατηση

΄Ετσι, με την κατάλληλη δημιουργία δεικτών διατηρούμε το σημείο της βάσης στο οποίο θα γίνει η αναζήτηση των δεδομένων με αποτέλεσμα να επιταχύνονται τα αποτελέσματα που εμφανίζονται στο ερώτημα 7. Τέλος, η βάση ορίζει αυτόματα indices για τα primary keys.

1. c.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε τα παρακάτω προγραμματιστικά εργαλεία και γλώσσες προγραμματισμού:

- `Python 3.9.0.`
- `Flask 2.0.1`
- `Werkzeug 2.0.1`
- `SQLITE3`
- `Bootstrap 4.0`
- `JavaScript`

1. d.

Εγκατάσταση της εφαρμογής μας σε περιβάλλον MacOS/Linux και Windows:

Καταρχάς βεβαιωθείτε ότι μπορείτε να τρέξετε μία έκδοση της Python3 στον Υπολογιστή σας (με την εντολή python --version βλέπετε την έκδοση της Python που υπάρχει στον υπολογιστή σας, αρκεί να είναι έκδοσης 3.0 και άνω)

Ακολουθήστε τις οδηγίες εγκατάστασης του Python Flask που περιγράφονται στη διεύθυνση:

https://flask.palletsprojects.com/en/2.0.x/installation/

Αφού δημιουργήσατε το virtual environment μέσα στο directory που είναι ο server δίνετε την εντολή Python3 server.py.

2.1.

SQL κώδικας ανά ερώτημα

```
**********************************************************************************
                                    Ερώτημα 7
**********************************************************************************

SELECT receive_services.NFC_id, services.service_description, receive_services.charge_time,
service_charge.cost
FROM receive_services, service_charge,services
WHERE receive_services.service_id = service_charge.service_id
AND receive_services.charge_time = service_charge.charge_time
AND services.service_id = service_charge.service_id
AND services.service_id = 1
AND strftime('%Y',service_charge.charge_time) = strftime('%Y','2021-06-15')
AND strftime('%m',service_charge.charge_time) = strftime('%m','2021-06-15')
AND strftime('%d',service_charge.charge_time) = strftime('%d','2021-06-15')
AND service_charge.cost <= 50
```

```
********************************************************************************
                                    Ερώτημα 8
********************************************************************************

CREATE VIEW sales_category AS
SELECT DISTINCT s.service_description, SUM(sc.cost)
FROM services AS s, service_charge AS sc
WHERE s.service_id=sc.service_id
GROUP BY  s.service_description


CREATE VIEW customer_data AS
SELECT DISTINCT  customer.*,customer_phone.phone,customer_email.email
FROM customer
LEFT JOIN customer_phone
ON customer.NFC_id = customer_phone.NFC_id
LEFT JOIN customer_email
ON  customer.NFC_id = customer_email.NFC_id

SELECT * FROM sales_category

SELECT * FROM customer_data

(Τα αποτελέσματα DISTINCT queries δεν είναι ενημερώσιμα (Silberschatz σελ.125))


********************************************************************************
                                    Ερώτημα 9
********************************************************************************

SELECT p.place_name, v.entry_time, v.exit_time, v.NFC_id
FROM place AS p
INNER JOIN visit AS v
ON p.place_id=v.place_id
WHERE v.NFC_id = 3


********************************************************************************
                                    Ερώτημα 10
********************************************************************************

SELECT  customer.NFC_id, customer.first_name, customer.last_name
FROM customer
NATURAL JOIN(
SELECT DISTINCT v2.NFC_id
FROM visit as v1, visit as v2
WHERE v1.place_id = v2.place_id
AND strftime('%Y',v1.entry_time) = strftime('%Y',v2.entry_time)
AND strftime('%m',v1.entry_time) = strftime('%m',v2.entry_time)
```

```
AND strftime('%H',v2.entry_time) BETWEEN strftime('%H',v1.entry_time) AND
strftime('%H',v1.exit_time,"+1 hours")
AND v1.NFC_id = 3 AND v1.NFC_id <> v2.NFC_id
)


*********************************************************************************
                                    Ερώτημα 11
*********************************************************************************

* * * * * * * * * * * * * *   Question 11(a)   * * * * * * * * * * * * * *

WITH age_group1(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-40 AND 2021-20)

SELECT place_name, place_position, COUNT(place_name) AS total_visits
FROM visit, place, age_group1
WHERE visit.place_id = place.place_id AND age_group1.val1 = visit.NFC_id
GROUP BY place_name, place_position
ORDER BY COUNT(place_name) DESC
LIMIT 10

WITH age_group2(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-60 AND 2021-41)

SELECT place_name, place_position, COUNT(place_name) AS total_visits
FROM visit, place, age_group2
WHERE visit.place_id = place.place_id AND age_group2.val1 = visit.NFC_id
GROUP BY place_name
ORDER BY COUNT(place_name) DESC
LIMIT 10

WITH age_group3(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year >= 2021-61)

SELECT place_name, place_position, COUNT(place_name) AS total_visits
FROM visit, place, age_group3
WHERE visit.place_id = place.place_id AND age_group3.val1 = visit.NFC_id
GROUP BY place_name
ORDER BY COUNT(place_name) DESC
LIMIT 10


* * * * * * * * * * * * * *   Question 11(b)   * * * * * * * * * * * * * *

WITH age_group1(val1) AS
```

```
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-40 AND 2021-20)

SELECT service_description, COUNT(service_description) AS total_uses
FROM receive_services, services, age_group1
WHERE receive_services.service_id = services.service_id AND age_group1.val1 =
receive_services.NFC_id
GROUP BY service_description
ORDER BY COUNT(service_description) DESC

WITH age_group2(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-60 AND 2021-41)

SELECT service_description, COUNT(service_description) AS total_uses
FROM receive_services, services, age_group2
WHERE receive_services.service_id = services.service_id AND age_group2.val1 =
receive_services.NFC_id
GROUP BY service_description
ORDER BY COUNT(service_description) DESC

WITH age_group3(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year <= 2021-61)

SELECT service_description, COUNT(service_description) AS total_uses
FROM receive_services, services, age_group3
WHERE receive_services.service_id = services.service_id AND age_group3.val1 =
receive_services.NFC_id
GROUP BY service_description
ORDER BY COUNT(service_description) DESC




* * * * * * * * * * * * * * *   Question 11(c)   * * * * * * * * * * * * * * *

 WITH age_group1(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-40 AND 2021-20),
helpy(val2, val3) AS
        (SELECT DISTINCT NFC_id, service_id
        FROM receive_services)

SELECT service_description, COUNT(service_description) as total_choices
FROM services, helpy, age_group1
WHERE age_group1.val1 = helpy.val2 AND helpy.val3 = services.service_id
GROUP BY service_description
```

```
ORDER BY COUNT(service_description) DESC

WITH age_group2(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year BETWEEN 2021-60 AND 2021-41),
helpy(val2, val3) AS
        (SELECT DISTINCT NFC_id, service_id
        FROM receive_services)

SELECT service_description, COUNT(service_description) as total_choices
FROM services, helpy, age_group2
WHERE age_group2.val1 = helpy.val2 AND helpy.val3 = services.service_id
GROUP BY service_description
ORDER BY COUNT(service_description) DESC

WITH age_group3(val1) AS
        (SELECT NFC_id
        FROM customer
        WHERE birth_year <= 2021-61),
helpy(val2, val3) AS
        (SELECT DISTINCT NFC_id, service_id
        FROM receive_services)

SELECT service_description, COUNT(service_description) as total_choices
FROM services, helpy, age_group3
WHERE age_group3.val1 = helpy.val2 AND helpy.val3 = services.service_id
GROUP BY service_description
ORDER BY COUNT(service_description) DESC
```

2.2.

SQL DDL

```
CREATE TABLE "access" (
        "NFC_id"        INTEGER,
        "place_id"      INTEGER,
        "open_time"     INTEGER COLLATE UTF16CI,
        "close_time"    INTEGER,
        FOREIGN KEY("place_id") REFERENCES «place"("place_id") ON DELETE NO ACTION ON
UPDATE NO ACTION,
        FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON
UPDATE CASCADE,
```

```
	PRIMARY KEY("NFC_id","place_id")
)

CREATE TABLE "customer" (
	"NFC_id"	INTEGER NOT NULL UNIQUE,
	"first_name"	TEXT NOT NULL,
	"last_name"	TEXT NOT NULL,
	"birth_day"	INTEGER NOT NULL,
	" birth_month"	INTEGER NOT NULL,
	"birth_year"	INTEGER NOT NULL,
	"id_doc_number"		TEXT NOT NULL UNIQUE,
	"type_of_id"	TEXT NOT NULL,
	"issuing_auth"	TEXT NOT NULL,
	PRIMARY KEY("NFC_id" AUTOINCREMENT)
)

CREATE TABLE "customer_email" (
	"NFC_id"	INTEGER,
	"email" NUMERIC CHECK("email" LIKE '%_@__%.__%') UNIQUE,
	FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON
UPDATE CASCADE,
	PRIMARY KEY("NFC_id","email")
)

CREATE TABLE "customer_phone" (
	"NFC_id"	INTEGER,
	"phone"NUMERIC(10, 0) UNIQUE,
	PRIMARY KEY("NFC_id","phone"),
	FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON
UPDATE CASCADE
)

CREATE TABLE "place" (
	"place_id"	INTEGER UNIQUE,
	"bed_num"	INTEGER NOT NULL,
	"place_name"	INTEGER NOT NULL,
	"place_position"		INTEGER NOT NULL,
	PRIMARY KEY("place_id" AUTOINCREMENT)
)

CREATE TABLE "provided_at" (
	"service_id"	INTEGER ON DELETE NO ACTION ON UPDATE NO ACTION,
	"place_id"	INTEGER ON DELETE NO ACTION ON UPDATE NO ACTION,
	PRIMARY KEY("place_id","service_id"),
	FOREIGN KEY("service_id") REFERENCES "services"("service_id"),
	FOREIGN KEY("place_id") REFERENCES "place"("place_id")
)

CREATE TABLE "receive_services" (
	"NFC_id"	INTEGER,
	"service_id"	INTEGER,
	"charge_time"	TEXT,
	PRIMARY KEY("NFC_id","service_id","charge_time"),
	FOREIGN KEY("service_id") REFERENCES «services"("service_id") ON DELETE NO ACTION
ON UPDATE NO ACTION,
```

```
	FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY("charge_time") REFERENCES "service_charge"("charge_time")
)

CREATE TABLE "service_charge" (
	"service_id"	INTEGER,
	"charge_time"	INTEGER,
	"charge_description"	TEXT,
	"cost"	INTEGER,
	PRIMARY KEY("service_id","charge_time"),
	FOREIGN KEY("service_id") REFERENCES "services"("service_id") ON DELETE NO ACTION ON UPDATE NO ACTION
)

CREATE TABLE "services" (
	"service_id"	INTEGER,
	"service_description"	TEXT NOT NULL UNIQUE,
	"service_type"	TEXT NOT NULL CHECK("service_type" IN ('no subscription', 'subscription')),
	PRIMARY KEY("service_id")
)

CREATE TABLE "subscribes" (
	"NFC_id"	INTEGER,
	"service_id"	INTEGER,
	"time_of_sub"	TEXT,
	PRIMARY KEY("NFC_id","service_id"),
	FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON UPDATE CASCADE
	 FOREIGN KEY("service_id") REFERENCES) "services"("service_id") ON DELETE NO ACTION ON UPDATE NO ACTION

CREATE TABLE "visit" (
	"NFC_id"	INTEGER,
	"place_id"	INTEGER,
	"entry_time"	TEXT,
	"exit_time"	TEXT NOT NULL,
	PRIMARY KEY("NFC_id","place_id","entry_time"),
	FOREIGN KEY("place_id") REFERENCES «place"("place_id") ON DELETE NO ACTION ON UPDATE NO ACTION,
	FOREIGN KEY("NFC_id") REFERENCES "customer"("NFC_id") ON DELETE CASCADE ON UPDATE CASCADE
)
```