ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
Εξαμηνιαία Εργασία (Project)
Ομάδα 24

Ομάδα:

Ονοματεπώνυμο	Αριθμός Μητρώου
1.Χατζηκυριάκος Γιώργος	03111164
2.Τσιτσεκλής Κωνσταντίνος	03111409
3.Πενταράκης Εμμανουήλ	03111048

Πλατφόρμα Υλοποίησης – Περιβάλλον ανάπτυξης

Για την σχεδίαση της βάσης δεδομένων της εταιρίας "Ιδανικό Σπίτι" χρησιμοποιήθηκε το σύστημα διαχείρισης Βάσεων Δεδομένων **MySQL** . Για την διαχείριση και την παρουσίαση της παραπάνω βάσης χρησιμοποιήθηκε η γλώσσα **html** για την αναπαράσταση σε μορφή ιστοσελίδας και η γλώσσα **PHP** για επικοινωνία μεταξύ browser και server. Για την δημιουργία ενός τοπικού server χρησιμοποιήθηκε το πακέτο λογισμικού **LAMP** (Linux Apache MySQL PHP) που περιέχει το εργαλείο **phpMyAdmin** που προσφέρει γραφικό περιβάλλον για την διαχείριση της βάσης δεδομένων.

Επιλέχθηκε αυτή η πλατφόρμα διότι το γραφικό περιβάλλον που μας παρέχεται είναι εύκολα διαχειρίσιμο και βοηθάει ιδιαίτερα στο να ελεγχθεί αν η βάση μας ικανοποιεί τους περιορισμούς ακεραιότητας . Επίσης όλα τα προγράμματα που χρησιμοποιήθηκαν είναι ανοιχτού λογισμικού που σημαίνει ότι υπάρχει πολύ βοηθητικό υλικό στο διαδίκτυο. Από την άλλη, η γλώσσα σεναρίων PHP είναι αρκετά δυσνόητη σε ορισμένες εφαρμογές και υπήρχε μία μικρή δυσκολία στην πλήρη κατανόηση της.

Σχεσιακό Μοντέλο Βάσης

Αποτελείται από τις εξής σχέσεις:

ΥΠΑΛΛΗΛΟΣ (**Κωδικός_Υπαλλήλου**, Όνομα, Επώνυμο, Οδός, Αριθμός, ΤΚ, Φύλο, Μισθός, E-Mail, Τηλέφωνο, Προϊστάμενος) .

ΠΕΛΑΤΗΣ (**Κωδικός_Πελάτη**, Όνομα, Επώνυμο, Οδός, Αριθμός, ΤΚ, Μέγιστο_Ενοίκιο, Προτιμώμενο_Είδος, E-Mail, Τηλέφωνο) .

ΙΔΙΟΚΤΗΤΗΣ (**Κωδικός_Ιδιοκτήτη**, ΑΦΜ, Οδός, Αριθμός, ΤΚ, E-Mail, Τηλέφωνο).

ΙΔΙΩΤΗΣ (**Κωδικός_Ιδιοκτήτη**, Όνομα, Επώνυμο).

ΕΠΙΧΕΙΡΗΣΗ (**Κωδικός_Ιδιοκτήτη**, Όνομα_Επιχείρησης, Είδος_Επιχείρησης).

ΑΚΙΝΗΤΑ (**Κωδικός_Ακινήτου**, Είδος, Αριθμός_Τμ, Μηνιαίο_Ενοίκιο, Αριθμός_Δωματίων, Οδός, Αριθμός, ΤΚ, Κωδικός_Υπαλλήλου, Κωδικός_Ιδιοκτήτη).

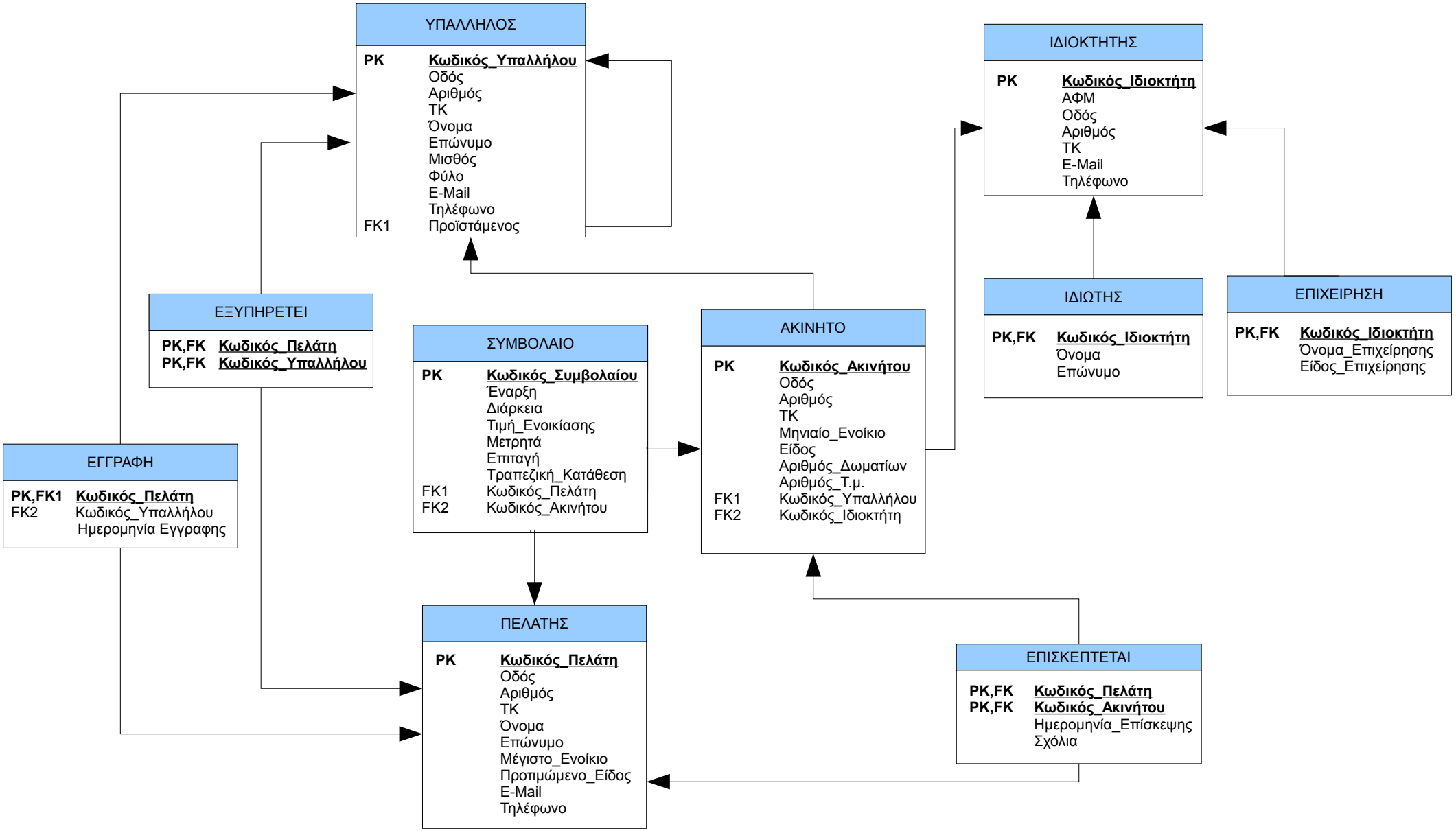
ΣΥΜΒΟΛΑΙΟ (**Κωδικός_Συμβολαίου**, Μετρητά, Επιταγή, Τραπεζική_Κατάθεση, Τιμή_Ενοικίασης, Έναρξη, Διάρκεια, Κωδικός_Ακινήτου, Κωδικός_Πελάτη).

ΕΞΥΠΗΡΕΤΕΙ (**Κωδικός_Πελάτη**, **Κωδικός_Υπαλλήλου**).

ΕΓΓΡΑΦΗ (**Κωδικός_Πελάτη**, **Κωδικός_Υπαλλήλου**, Ημερομηνία_Εγγραφής).

ΕΠΙΣΚΕΠΤΕΤΑΙ (**Κωδικός_Πελάτη**, **Κωδικός_Ακινήτου**, Ημερομηνία_Επίσκεψης, Σχόλια).

Ένα πρόβλημα που είχαμε με την υλοποίηση της βάσης χρησιμοποιώντας τους παραπάνω πίνακες ήταν η εισαγωγή ενός ιδιοκτήτη, διότι αφορά 2 πίνακες ("Ιδιοκτήτης"- "Ιδιώτης/Επειχήρηση"). Ιδιαίτερα στην αποτυχία εισαγωγής στον ένα πίνακα και επιτυχία στον άλλο η βάση δεν θα ήταν λογικά ορθή. Αυτό το λύσαμε σε επίπεδο κώδικα html και PHP.
Στην συνέχεια ακολουθεί η σχηματική αναπαράσταση του σχεσιακού μοντέλου όπου φαίνονται καλύτερα οι σχέσεις μεταξύ των πινάκων καθώς και τα πρωτεύοντα (Primary Key – PK) και ξένα κλειδιά (Foreign Key - FK).


ΥΠΑΛΛΗΛΟΣ
PK Κωδικός_Υπαλλήλου
Οδός
Αριθμός
ΤΚ
Όνομα
Επώνυμο
Μισθός
Φύλο
E-Mail
Τηλέφωνο
FK1 Προϊστάμενος
ΙΔΙΟΚΤΗΤΗΣ
PK Κωδικός_Ιδιοκτήτη
ΑΦΜ
Οδός
Αριθμός
ΤΚ
E-Mail
Τηλέφωνο
ΕΞΥΠΗΡΕΤΕΙ
PK,FK Κωδικός_Πελάτη
PK,FK Κωδικός_Υπαλλήλου
ΣΥΜΒΟΛΑΙΟ
PK Κωδικός_Συμβολαίου
Έναρξη
Διάρκεια
Τιμή_Ενοικίασης
Μετρητά
Επιταγή
Τραπεζική_Κατάθεση
FK1 Κωδικός_Πελάτη
FK2 Κωδικός_Ακινήτου
ΑΚΙΝΗΤΟ
PK Κωδικός_Ακινήτου
Οδός
Αριθμός
ΤΚ
Μηνιαίο_Ενοίκιο
Είδος
Αριθμός_Δωματίων
Αριθμός_Τ.μ.
FK1 Κωδικός_Υπαλλήλου
FK2 Κωδικός_Ιδιοκτήτη
ΙΔΙΩΤΗΣ
PK,FK Κωδικός_Ιδιοκτήτη
Όνομα
Επώνυμο
ΕΠΙΧΕΙΡΗΣΗ
PK,FK Κωδικός_Ιδιοκτήτη
Όνομα_Επιχείρησης
Είδος_Επιχείρησης
ΕΓΓΡΑΦΗ
PK,FK1 Κωδικός_Πελάτη
FK2 Κωδικός_Υπαλλήλου
Ημερομηνία Εγγραφης
ΠΕΛΑΤΗΣ
PK Κωδικός_Πελάτη
Οδός
Αριθμός
ΤΚ
Όνομα
Επώνυμο
Μέγιστο_Ενοίκιο
Προτιμώμενο_Είδος
E-Mail
Τηλέφωνο
ΕΠΙΣΚΕΠΤΕΤΑΙ
PK,FK Κωδικός_Πελάτη
PK,FK Κωδικός_Ακινήτου
Ημερομηνία_Επίσκεψης
Σχόλια

Περιορισμοί - Ακεραιότητα

Όπως βλέπουμε από το προηγούμενο σχήμα ικανοποιείται η ακεραιότητα οντότητας (entity integrity) αφού σε κάθε πίνακα υπάρχει ένα πρωτεύον κλειδί το οποίο είναι μοναδικό (στις περισσότερες σχέσεις μάλιστα είναι "Auto-Increment" που εξασφαλίζει μοναδικότητα). Επίσης υπάρχει ο περιορισμός για τα PK να μην είναι NULL που με τον τρόπο που κατασκευάσαμε την βάση ικανοποιείται.

Ταυτόχρονα πληρείται και η αναφορική ακεραιότητα (referential integrity) καθώς δεν υπάρχει πιθανότητα τα FK να δείχνουν σε μία τούπλα που δεν υπάρχει. Συγκεκριμένα , για την ορθή και λογική λειτουργία της βάσης σχεδόν όλες οι σχέσεις μεταξύ πινάκων που δημιουργούνται λόγω των FK έχουν ρυθμιστεί "ON DELETE CASCADE". Για παράδειγμα αν διαγραφεί μία τούπλα ενός πελάτη θα σβηστεί και η αντίστοιχη τούπλα, που περιέχει FK ίδιο με το PK που διαγράφτηκε , στους πίνακες "Συμβόλαιο", "Εγγραφή", "Εξυπηρετεί", "Επισκέπτεται". Αυτό βέβαια δεν είναι λογικά ορθό στις σχέσεις μεταξύ "Υπάλληλος" - "Ακίνητο" και "Υπάλληλος"- "Προϊστάμενος". Για παράδειγμα αν διαγραφεί ένας υπάλληλος δεν είναι λογικό να διαγραφεί και το ακίνητο για το οποίο είναι υπεύθυνος. Αυτές οι δύο σχέσεις έχουν ρυθμιστεί "ON DELETE NULL" που σημαίνει ότι αν διαγραφεί ένα PK το FK που αναφέρεται σε αυτό θα γίνει NULL. Τα FK σε περίπτωση update είναι "ON UPDATE CASCADE" χωρίς να έχει ιδιαίτερη σημασία καθώς τα περισσότερα PK δεν μπορούν να αλλαχθούν από τον χρήστη και μερικά που μπορούν έχουν περιορισμένο πεδίο τιμών που δίνεται από το εκάστοτε "drop-down".

Η ακεραιότητα πεδίου τιμών (domain integrity) εξασφαλίστηκε βάζοντας τον MySQL server σε "Strict Mode", διότι διαφορετικά αν για παράδειγμα δίναμε String σε πεδίο που ήταν ορισμένο για INT η βάση δεν έδειχνε error αλλά warning και έμπαινε μια λανθασμένη τιμή (στην περίπτωση μας το 0). Με αυτό τον τρόπο, λοιπόν, τα πεδία πού έχουν σύνολο τιμών ακεραίους (INT) δεν μπορούν να πάρουν ένα String για παράδειγμα. Συγκεκριμένα στην βάση μας όλα τα PK,FK είναι INT , πεδία όπως το Postal Code είναι INT επίσης, πεδία όπως το Telephone είναι BIGINT ενώ πεδία όπως το E-Mail είναι VARCHAR , όπου το μέγεθος αλλάζει ανάλογα με το πεδίο.

Τέλος, όσον αναφορά τους περιορισμούς οριζόμενους από τον χρήστη (User defined integrity) δεν επιτρέπουμε από το User Interface να αλλάξουν τα PK των σχέσεων (αφήνουμε μόνο την διαγραφή τους) και έχουμε ορισμένες τιμές που θεωρήσαμε ότι είναι δυνατόν να παραμείνουν κενές (NULL) , όπως το "Οδός" στο "Υπάλληλος" και άλλες που πρέπει οπωσδήποτε να συμπληρωθούν, όπως το "Όνομα" στο "Πελάτης". Ακόμη έχουμε υποθέσει ότι έναν πελάτης μπορεί να επισκεφτεί το κάθε ακίνητο μία μόνο φορά.

Ευρετήρια

Στην βάση δεδομένων όλα τα PK έχουν ευρετήριο (Index) ώστε να γίνεται ευκολότερα η αναζήτηση στον κάθε πίνακα. Επίσης ο MySQL server για την δημιουργία και την λειτουργία των FK, απαιτούσε όποιο πεδίο πρόκειται να γίνει FK να έχει δικό του ευρετήριο. Πέρα από αυτά , κρίναμε σημαντικό να προσθέσουμε επιπλέον ευρετήρια στα "Όνομα", "Επίθετο" του "Υπάλληλος" και "Πελάτης" καθώς εκτελούμε συχνά ερωτήματα στη βάση με ισότητα ως προς το Όνομα και το Επίθετο κάποιου.
Επιπλέον, όταν δημιουργούσαμε ευρετήρια τα δηλώσαμε αρχικά "Unique" πράγμα που δεν οδηγούσε σε λογική λειτουργία της βάσης μας, καθώς για παράδειγμα ένας υπάλληλος δεν μπορούσε να είναι υπεύθυνος για περισσότερα σπίτια (το FK1 του "Ακινήτου" ήταν "Unique"). Αυτό το αντιμετωπίσαμε κάνοντας τις τιμές των ευρετηρίων όχι μοναδικές (επιλογή "Index").

Queries

Ενδεικτικά φτιάξαμε τα παρακάτω ερωτήματα που θεωρήσαμε ότι περιέχουν σημαντικές πληροφορίες για την εταιρία "Ιδανικό Σπίτι".

Εμφανίζει συνολικά πόσους μισθούς πληρώνει η εταιρία για υπαλλήλους.

- ```
  select sum(`Salary`) from `Employee`
  ```

Μας δείχνει για όλους τους πελάτες που έχουν συμβόλαιο με ποιο ακίνητο το έχουν αυτό και τα στοιχεία του συμβολαίου τους.

- ```
  select c.Name, c.Surname, e.Street, e.`Street Num`, e.`Postal
  Code`, g.Starts, g.Duration, g.Rent, g.`Payment Method`
  from `contract` as g
  inner join `customer` as c
  on c.Customer_ID=g.Customer_ID
  inner join `estate` as e
  on g.Estate_ID=e.Estate_ID
  ```

Μας δείχνει για όλους τους υπαλλήλους, για ποια ακίνητα είναι υπεύθυνος ο καθένας.

- ```
  select em.Name, em.Surname, es.Street as EstateStreet, es.`Street
  Num` as EstateStreetNum, es.`Postal Code`
  from `employee`as em
  inner join `estate` as es
  on em.Employee_ID=es.Employee_ID
  ```

Διαλέγει τους υπαλλήλους που έχουν μισθό μεγαλύτερο από όσο δίνει ο χρήστης (**X**) και τους εμφανίζει με αύξουσα σειρά.

- ```
  select e.Name , e.Surname , .e.Salary
  from `employee` as e
  where e.Salary > X
  order by Salary asc
  ```

Βρίσκει πόσα ακίνητα έχει διαθέσει κάθε ιδιώτης στην εταιρία μας.

- ```
  select p.Name,p.Surname,count(e.Owner_ID) as NumberOfEstates
  from `owner` as o
  right join `private` as p
  on p.Owner_ID=o.Owner_ID
  left join `estate`as e
  on e.Owner_ID=o.Owner_ID
  group by p.Name,p.Surname
  ```

Βρίσκει πόσα ακίνητα έχει διαθέσει κάθε επιχείρηση στην εταιρία μας.

- ```
  select c.`Company Name`,count(e.Owner_ID) as NumberOfEstates
  from `owner` as o
  right join `company` as c
  on c.Owner_ID=o.Owner_ID
  left join `estate`as e
  on e.Owner_ID=o.Owner_ID
  group by c.`Company Name`
  ```

Βρίσκει τους υπαλλήλους που εξυπηρετούν παραπάνω πελάτες από έναν αριθμό που δίνει ο χρήστης (**X**).

```
select e.Name, e.Surname, count(s.Customer_ID) as NumberOfCustomers
from `serves` as s
right join `employee` as e
on e.Employee_ID=s.Employee_ID
group by  e.Name, e.Surname
having NumberOfCustomers >= X
```

Βρίσκει τα ακίνητα που δεν έχουν συμβόλαιο.

```
select E.Street, E.`Street Num `
from   `estate` as E
where E.Estate_ID not in ( select `Estate_ID` from `contract` )
```

Πόσες επισκέψεις έχει κάθε ακίνητο.

```
select e.Street,e.`Street Num`,count(v.`Visit Date`) as
NumberOfVisits
from `estate` as e
left join `visit` as v
on v.Estate_ID=e.Estate_ID
group by  e.Street,e.`Street Num`
```

Βρίσκει τις πληροφορίες του σπιτιού με το ελαχιστο ενοίκιο.

```
select e.`Monthly Rent`, e.Street, e.`Street Num`, e.`Postal
Code`,e.`Estate Type`, e.`Room Num`, e.`Square Meters`
from `estate`as e
where e.`Monthly Rent`=(select min(`Monthly Rent`) from `estate`)
```

Βρίσκει για τον πελάτη που δίνει ο χρήστης (όνομα **X** και επίθετο **Y**) τα ακίνητα τα οποία μπορεί να νοικιάσει σύμφωνα με το μέγιστο ενοίκιο που έχει πει ότι μπορεί να διαθέσει.

```
select c.Name,c.Surname,c.`Home Type` ,c.`Maximum
Rent`,e.Street,e.`Street Num`, e.`Postal Code`,e.`Estate
Type`,e.`Monthly Rent`
from `customer` as c
inner join `estate` as e
on e.`Monthly rent`< c.`Maximum Rent`
where c.`Name` like 'X' and c.`Surname` like 'Y'
```

DDLs Των Tables Της Βάσης

Για τις εντολές που γεμίζουν τον πίνακα θα αναφέρεται ενδεικτικά για μία τουπλα.

```
--
-- Table structure for table `company`
--

CREATE TABLE IF NOT EXISTS `company` (
  `Owner_ID` int(10) NOT NULL,
  `Company Name` varchar(20) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
  `Company Type` varchar(25) CHARACTER SET utf8 COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`Owner_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `company`
--

INSERT INTO `company` (`Owner_ID`, `Company Name`, `Company Type`) VALUES
(11, 'Ideal Home', 'LTE');

--
-- Table structure for table `contract`
--

CREATE TABLE IF NOT EXISTS `contract` (
  `Contract_ID` int(10) NOT NULL AUTO_INCREMENT,
  `Starts` date NOT NULL,
  `Duration` int(10) NOT NULL,
  `Rent` int(10) NOT NULL,
  `Payment Method` varchar(15) NOT NULL,
  `Customer_ID` int(10) NOT NULL,
  `Estate_ID` int(10) NOT NULL,
  PRIMARY KEY (`Contract_ID`),
  UNIQUE KEY `Estate_ID` (`Estate_ID`),
  KEY `Customer_ID` (`Customer_ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;

--
-- Dumping data for table `contract`
--

INSERT INTO `contract` (`Contract_ID`, `Starts`, `Duration`, `Rent`, `Payment Method`,
`Customer_ID`, `Estate_ID`) VALUES
(1, '2015-01-05', 12, 1000, 'Bank Deposit', 8, 29);

--
-- Table structure for table `customer`
--

CREATE TABLE IF NOT EXISTS `customer` (
  `Customer_ID` int(10) NOT NULL AUTO_INCREMENT,
  `Street` varchar(15) DEFAULT NULL,
  `Street Num` int(10) DEFAULT NULL,
  `Postal Code` int(10) DEFAULT NULL,
  `Name` varchar(15) NOT NULL,
  `Surname` varchar(20) NOT NULL,
  `Maximum Rent` int(10) NOT NULL,
  `Home Type` varchar(25) DEFAULT NULL,
  `E-Mail` varchar(40) NOT NULL,
```

```
  `Telephone` bigint(20) NOT NULL,
  PRIMARY KEY (`Customer_ID`),
  KEY `FullName` (`Name`,`Surname`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=19 ;

--
-- Dumping data for table `customer`
--

INSERT INTO `customer` (`Customer_ID`, `Street`, `Street Num`, `Postal Code`, `Name`,
`Surname`, `Maximum Rent`, `Home Type`, `E-Mail`, `Telephone`) VALUES
(5, 'Moscow St.', 5, 123123, 'Diana', 'Cyka', 420, 'Small Appartment',
'giffmemana@dota2.ru', 4567886);

--
-- Table structure for table `employee`
--

CREATE TABLE IF NOT EXISTS `employee` (
  `Employee_ID` int(10) NOT NULL AUTO_INCREMENT,
  `Street` varchar(20) CHARACTER SET ascii DEFAULT NULL,
  `Street Num` int(10) DEFAULT NULL,
  `Postal Code` int(10) DEFAULT NULL,
  `Name` varchar(15) CHARACTER SET ascii NOT NULL,
  `Surname` varchar(20) CHARACTER SET ascii NOT NULL,
  `Salary` int(15) NOT NULL,
  `Sex` char(1) CHARACTER SET ascii NOT NULL,
  `E-Mail` varchar(40) NOT NULL,
  `Telephone` bigint(20) NOT NULL,
  `Supervisor` int(11) DEFAULT NULL,
  PRIMARY KEY (`Employee_ID`),
  KEY `Supervisor` (`Supervisor`),
  KEY `fullName` (`Name`,`Surname`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=60 ;

--
-- Dumping data for table `employee`
--

INSERT INTO `employee` (`Employee_ID`, `Street`, `Street Num`, `Postal Code`, `Name`,
`Surname`, `Salary`, `Sex`, `E-Mail`, `Telephone`, `Supervisor`) VALUES
(54, 'Obere Str.', 37, 12209, 'Johnny', 'Depp', 8000, 'M', 'johnny@gmail.com',
2106789550, NULL);

--
-- Table structure for table `estate`
--
CREATE TABLE IF NOT EXISTS `estate` (
  `Estate_ID` int(10) NOT NULL AUTO_INCREMENT,
  `Street` varchar(15) NOT NULL,
  `Street Num` int(10) NOT NULL,
  `Postal Code` int(10) NOT NULL,
  `Monthly Rent` int(10) NOT NULL,
  `Estate Type` varchar(25) NOT NULL,
  `Room Num` int(10) NOT NULL,
  `Square Meters` int(10) NOT NULL,
  `Employee_ID` int(10) DEFAULT NULL,
  `Owner_ID` int(10) NOT NULL,
  PRIMARY KEY (`Estate_ID`),
  KEY `Owner_ID` (`Owner_ID`),
  KEY `Employee_ID` (`Employee_ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=36 ;
```

```sql
--
-- Dumping data for table `estate`
--

INSERT INTO `estate` (`Estate_ID`, `Street`, `Street Num`, `Postal Code`, `Monthly
Rent`, `Estate Type`, `Room Num`, `Square Meters`, `Employee_ID`, `Owner_ID`) VALUES
(29, 'Laurent', 54, 15423, 1000, 'Flat', 3, 83, 57, 8);

--
-- Table structure for table `owner`
--

CREATE TABLE IF NOT EXISTS `owner` (
  `Owner_ID` int(10) NOT NULL AUTO_INCREMENT,
  `Street` varchar(15) CHARACTER SET ascii DEFAULT NULL,
  `Street Num` int(10) DEFAULT NULL,
  `Postal Code` int(10) DEFAULT NULL,
  `E-Mail` varchar(40) NOT NULL,
  `Telephone` bigint(20) NOT NULL,
  `Tax Registration Number` bigint(20) NOT NULL,
  PRIMARY KEY (`Owner_ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=14 ;

--
-- Dumping data for table `owner`
--

INSERT INTO `owner` (`Owner_ID`, `Street`, `Street Num`, `Postal Code`, `E-Mail`,
`Telephone`, `Tax Registration Number`) VALUES
(8, 'Karaoli', 34, 16232, 'manolios93@hotmail.com', 697895432, 1608789909);

--
-- Table structure for table `private`
--

CREATE TABLE IF NOT EXISTS `private` (
  `Owner_ID` int(10) NOT NULL,
  `Name` varchar(15) NOT NULL,
  `Surname` varchar(20) NOT NULL,
  PRIMARY KEY (`Owner_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `private`
--

INSERT INTO `private` (`Owner_ID`, `Name`, `Surname`) VALUES
(8, 'Manolis', 'Pentarakis');

--
-- Table structure for table `registration`
--

CREATE TABLE IF NOT EXISTS `registration` (
  `Registration Date` date NOT NULL,
  `Customer_ID` int(11) NOT NULL,
  `Employee_ID` int(11) DEFAULT NULL,
  PRIMARY KEY (`Customer_ID`),
  KEY `Employee_ID` (`Employee_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```sql
--
-- Dumping data for table `registration`
--

INSERT INTO `registration` (`Registration Date`, `Customer_ID`, `Employee_ID`) VALUES
('2014-12-15', 5, 54);

--
-- Table structure for table `serves`
--

CREATE TABLE IF NOT EXISTS `serves` (
  `Customer_ID` int(10) NOT NULL,
  `Employee_ID` int(10) NOT NULL,
  PRIMARY KEY (`Customer_ID`,`Employee_ID`),
  KEY `Employee_ID` (`Employee_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `serves`
--

INSERT INTO `serves` (`Customer_ID`, `Employee_ID`) VALUES
(5, 54);

--
-- Table structure for table `visit`
--

CREATE TABLE IF NOT EXISTS `visit` (
  `Customer_ID` int(10) NOT NULL,
  `Estate_ID` int(10) NOT NULL,
  `Visit Date` date NOT NULL,
  `Comments` varchar(60) DEFAULT NULL,
  PRIMARY KEY (`Customer_ID`,`Estate_ID`),
  KEY `Estate_ID` (`Estate_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Dumping data for table `visit`
--

INSERT INTO `visit` (`Customer_ID`, `Estate_ID`, `Visit Date`, `Comments`) VALUES
(5, 29, '2016-03-25', '(Future visit, will add comments later)');

--
-- Constraints for table `company`
--
ALTER TABLE `company`
  ADD CONSTRAINT `company_ibfk_1` FOREIGN KEY (`Owner_ID`) REFERENCES `owner`
(`Owner_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `contract`
--
ALTER TABLE `contract`
  ADD CONSTRAINT `contract_ibfk_3` FOREIGN KEY (`Customer_ID`) REFERENCES `customer`
(`Customer_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `contract_ibfk_4` FOREIGN KEY (`Estate_ID`) REFERENCES `estate`
(`Estate_ID`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
--
-- Constraints for table `employee`
--
ALTER TABLE `employee`
  ADD CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`Supervisor`) REFERENCES `employee`
(`Employee_ID`) ON DELETE SET NULL ON UPDATE CASCADE;

--
-- Constraints for table `estate`
--
ALTER TABLE `estate`
  ADD CONSTRAINT `estate_ibfk_3` FOREIGN KEY (`Employee_ID`) REFERENCES `employee`
(`Employee_ID`) ON DELETE SET NULL ON UPDATE CASCADE,
  ADD CONSTRAINT `estate_ibfk_4` FOREIGN KEY (`Owner_ID`) REFERENCES `owner`
(`Owner_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `private`
--
ALTER TABLE `private`
  ADD CONSTRAINT `private_ibfk_1` FOREIGN KEY (`Owner_ID`) REFERENCES `owner`
(`Owner_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `registration`
--
ALTER TABLE `registration`
  ADD CONSTRAINT `registration_ibfk_1` FOREIGN KEY (`Customer_ID`) REFERENCES
`customer` (`Customer_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `registration_ibfk_2` FOREIGN KEY (`Employee_ID`) REFERENCES
`employee` (`Employee_ID`) ON DELETE SET NULL ON UPDATE CASCADE;

--
-- Constraints for table `serves`
--
ALTER TABLE `serves`
  ADD CONSTRAINT `serves_ibfk_1` FOREIGN KEY (`Employee_ID`) REFERENCES `employee`
(`Employee_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `serves_ibfk_2` FOREIGN KEY (`Customer_ID`) REFERENCES `customer`
(`Customer_ID`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Constraints for table `visit`
--
ALTER TABLE `visit`
  ADD CONSTRAINT `visit_ibfk_3` FOREIGN KEY (`Customer_ID`) REFERENCES `customer`
(`Customer_ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `visit_ibfk_4` FOREIGN KEY (`Estate_ID`) REFERENCES `estate`
(`Estate_ID`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Views

Στην βάση δεδομένων μας πέρα από τα ήδη υπάρχοντα table δημιουργήσαμε και 2 όψεις.

Η πρώτη όψη που δημιουργήθηκε αφορά την διαχείριση των πελατών τις εταιρίας “Ιδανικό σπίτι” και ονομάζεται “employee-customer” και δείχνει ποιος υπάλληλος εξυπηρετεί ποιον πελάτη. Η δημιουργία της όψης αυτής γίνεται με το παρακάτω κομμάτι κώδικα:

```
--
-- Stand-in structure for view `employee-customer`
--
CREATE TABLE IF NOT EXISTS `employee-customer` (
`EmployeeName` varchar(15)
,`EmployeeSurname` varchar(20)
,`CustomerName` varchar(15)
,`CustomerSurname` varchar(20)
);
--
-- Structure for view `employee-customer`
--
DROP TABLE IF EXISTS `employee-customer`;

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `employee-customer` AS
select `e`.`Name` AS `EmployeeName`,`e`.`Surname` AS `EmployeeSurname`,`c`.`Name` AS
`CustomerName`,`c`.`Surname` AS `CustomerSurname`
from ((`serves` `s` join `employee` `e` on((`e`.`Employee_ID` = `s`.`Employee_ID`)))
join `customer` `c` on((`c`.`Customer_ID` = `s`.`Customer_ID`))) WITH LOCAL CHECK
OPTION;
```

Όπως βλέπουμε αυτή η όψη δεν είναι ενημερώσιμη καθώς προκύπτει από συνένωση πινάκων.

Η δεύτερη όψη που κάναμε αφορά τα πολυτελή ακίνητα ,συγκεκριμένα αυτά που έχουν μηνιαίο ενοίκιο πάνω από 1500 χρηματικές μονάδες , και αυτό που κάνει να δείχνει να στοιχεία του ακινήτου αυτού που αφορούν το ακίνητο αυτό καθ' εαυτό. Η δημιουργία της όψης αυτής γίνεται με το παρακάτω κομμάτι κώδικα:

```
--
-- Stand-in structure for view `luxuryestates`
--
CREATE TABLE IF NOT EXISTS `luxuryestates` (
`Estate_ID` int(10)
,`Street` varchar(15)
,`Street Num` int(10)
,`Postal Code` int(10)
,`Monthly Rent` int(10)
,`Estate Type` varchar(25)
,`Room Num` int(10)
,`Square Meters` int(10)
);
--
-- Structure for view `luxuryestates`
--
DROP TABLE IF EXISTS `luxuryestates`;

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`
SQL SECURITY DEFINER VIEW `luxuryestates` AS select `e`.`Estate_ID` AS
`Estate_ID`,`e`.`Street` AS `Street`,`e`.`Street Num` AS `Street Num`,`e`.`Postal Code`
AS `Postal Code`,`e`.`Monthly Rent` AS `Monthly Rent`,`e`.`Estate Type` AS `Estate
Type`,`e`.`Room Num` AS `Room Num`,`e`.`Square Meters` AS `Square Meters`
from `estate` `e` where (`e`.`Monthly Rent` > 1500) order by `e`.`Monthly Rent`;
```

Αυτή η όψη όπως φαίνεται είναι ενημερώσιμη για update και delete καθώς προκύπτει από έναν πίνακα μόνο και περιέχει το PK του πίνακα από τον οποίο προέκυψε. Όμως σε αυτή την όψη δεν είναι σωστό να γίνει insert καθώς δεν περιέχει όλα τα στοιχειά του πίνακα προέλευσης που απαγορεύεται να είναι NULL.

Triggers

Η βάση δεδομένων που φτιάξαμε διαθέτει τρία triggers. Τα πρώτα δύο αφορούν τον ελάχιστο μισθό που μπορούν να έχουν οι υπάλληλοι. Υποθέσαμε ότι ο κατώτατος μισθός στην εταιρία "Ιδανικό Σπίτι" είναι 600 χρηματικές μονάδες και έτσι πριν από την εισαγωγή ή την αλλαγή στον πίνακα "Υπάλληλος" εάν ο μισθός είναι κάτω από 600 τον θέτουμε αυτόματα να γίνεται ίσο με το κατώτατο ποσό. Ο κώδικας που πραγματοποιεί τα trigger αυτά είναι ο παρακάτω:

```
DROP TRIGGER IF EXISTS `insertMinSalary`;
DELIMITER //
CREATE TRIGGER `insertMinSalary` BEFORE INSERT ON `employee`
 FOR EACH ROW IF new.Salary < 600
THEN
      SET new.Salary = 600;
END IF
//
DELIMITER ;

DROP TRIGGER IF EXISTS `updateMinSalary`;
DELIMITER //
CREATE TRIGGER `updateMinSalary` BEFORE UPDATE ON `employee`
 FOR EACH ROW IF new.Salary < 600
THEN
      SET new.Salary = 600;
END IF
//
DELIMITER ;
```

Το τελευταίο trigger γίνεται πάλι στον πίνακα "Υπάλληλος" και αφορά το γεγονός, ότι το FK "Κωδικός_Υπαλλήλου" έχει τεθεί "ON DELETE NULL", δηλαδή άμα φύγει ένας υπάλληλος το αντίστοιχο FK στον πίνακα "Ακίνητο" θα γίνει NULL. Οπότε πριν από τη διαγραφή υπαλλήλου βρίσκουμε εκείνο που είναι υπεύθυνος για τα λιγότερα σε αριθμό ακίνητα και δεν πρόκειται να διαγραφεί (διότι αυτός που διαγράφεται μπορεί να έχει τα λιγότερα) και του αναθέτουμε τα ακίνητα του υπαλλήλου που επρόκειτο να φύγει. Ο κώδικας γι' αυτό το trigger είναι ο παρακάτω:

```
DROP TRIGGER IF EXISTS `deleteEmployee`;
DELIMITER //
CREATE TRIGGER `deleteEmployee` BEFORE DELETE ON `employee`
FOR EACH ROW UPDATE `estate` SET Employee_ID=
      (select m.Employee_ID from (  select e.Employee_ID,count(w.Estate_ID) as Number
                                    from `employee`as e
                                    left join `estate`as w
                                    on w.Employee_ID=e.Employee_ID
                                    where e.Employee_ID <>old.Employee_ID
                                    group by e.Employee_ID
                                    order by Number ASC limit 0,1) as m)
WHERE Employee_ID=old.Employee_ID
//
DELIMITER ;
```