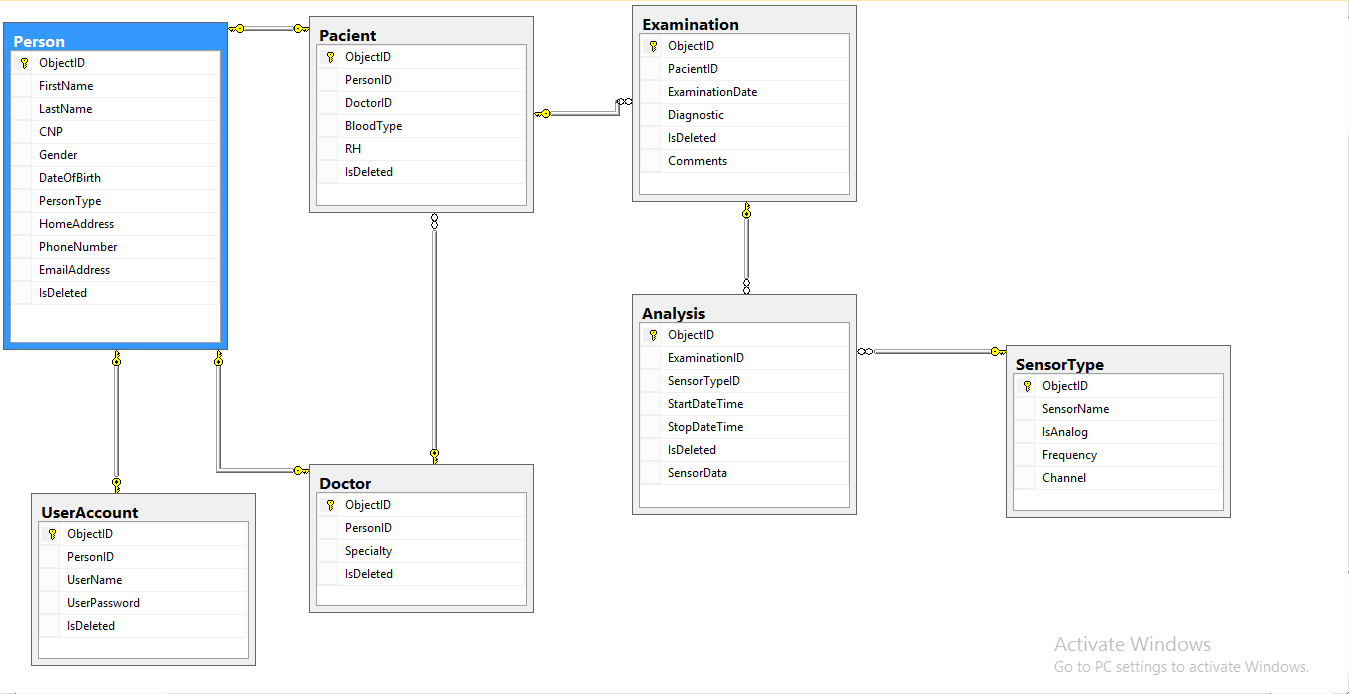
Prima etapa in proiectarea aplicatiei a fost creearea tabelelor bazei de date, deoarece in orice aplicatie medicala datele sunt elemental central. Astfel am ales o abordare “Database first” (inspirata din Entity framework ) deoarece este foarte usor sa creezi modelele (POJO) dupa ce stii care este forma tabelelor, deasemenea Microsoft SQL Server ofera o interfata grafica care vine in ajutorul programatorului fiind abordarea preferata cand vorbim de aplicatii complexe.

Principalul obiectiv al acestei etape a fost creeare unei baze de date care sa posede o logica cosistenta din punt de vedere al modelelor si incercarea de a evita redundanta datelor. În privinţa bazelor de date relaţionale, redundanţa înseamnă repetarea unor date în baza de date. În acest caz, problema redundanţei, în afară de consumul nejustificat de spaţiu de stocare, poate conduce şi la neactualizarea datelor în toate părţile unde acestea se repetă, ceea ce produce inconsistenţe în date.

Dupa cum se poate vedea in figura de mai jos fiecare table are un nume specific pentru datele care le poseda, si deasemenea se pot vedea relatiile dintre tabele. Tipurile de relatii intre tabele:

* Relaţia unu-la-unu (1-1) este tipul cel mai simplu de relaţie, prin care unui element din tabela R1 îi corespunde un singur element din tabela R2 şi reciproc. Relaţia 1-1 este mai rar folosită, în general pentru a reduce numărul de atribute dintr-o tabelă.
* Relaţia unu-la-mai mulţi (1-N) este tipul de relaţie prin care unui element din tabela R1 îi corespund unul sau mai multe elemente din tabela R2, iar unui element din tabela R2 îi corespunde un singur element din tabela R1.
* Relaţia mai mulţi-la-mai mulţi (M-N) prin care unui element din tabela R1 îi corespunde unul sau mai multe elemente din tabela R2 şi reciproc. Acest tip de relaţie este frecvent întâlnită, dar nu se poate implementa direct în bazele de date relaţionale. Pentru modelare se foloseşte o relaţie suplimentară, de tip 1-N pentru fiecare din tabelele iniţiale.



Dupa cum se poate observa in figra fiecare table are un identificatory unic de tip cheie primara. Cheia primară este câmpul sau combinaţia de mai multe câmpuri dintr-o tabelă, care permite identificarea unică a fiecărei înregistrări din acea tabelă. O cheie primară nu poate avea valoarea NULL si o tabelă poate avea o singură cheie primară.

Se mai pot observa relatiile dintre tabele pe baza cheii externe. O cheie externă este un câmp al unui tabel prin care se completează o relaţie identificând tabela părinte. Orice relaţie dintr-o bază de date relaţională trebuie să pornească dintr-o cheie externă.

Fiecare table are un camp IsDeleted care se refera la actualitatea si disonibilitatea inregistrarilor din baza de date. Acesta este un intreg care prin valoarea logica 1 se refera la date care nu mai sunt accesibile, valoarea 0 insemannd o disponibilitate a datelor din inregistrare. Pentru o aplicatie in domeniul medical datele sunt cruciale , iar prin stergerea lor din baza de date acestea devin neaccesibile pentru o activitate precum statistica persoanelor care au vizitat un doctor intr-un anumit an.

Unul dintre obiectivele cheie când se construieste o aplicație este asigurarea că datele care sunt introduse în baza de date îndeplinesc toate regulile convenite . Validarea datelor reprezintă o parte esențială a aplicației pentru a te asigura că datele respectă cerințele. Există un număr de tehnici diferite care pot fi utilizate pentru a te asigura că datele sunt valide în raport cu regulile. Regulile de validare a datelor pot fi impuse în cadrul aplicației în stratul aplicației sau datele pot fi validate de motorul bazei de date.

Tipurile de informații care sunt salvate în baza de date se numesc "entități". Aceste entități există în patru tipuri: oameni, lucruri, evenimente și locații. Tot ce ați putea dori să puneți într-o bază de date se încadrează într-una din aceste categorii. Dacă informațiile pe care doriți să le includeți nu se încadrează în aceste categorii, probabil nu este o entitate, ci o proprietate a unei entități, un atribut. Astfel primul pas in proiectarea bazei de date a fost identificarea entitailor, si s-au putut distinge 7 entitati : Person, UserAccount, Doctor, Patient, Examination, Analysis si SensorType.

Urmatorul pas a fost sa determin relatiile si cardinalitatea dintre entitati:

* UserAccount🡨 🡪Person : Un user account este o persoana ; 1:1
* Patient🡨 🡪 Person : Un pacient este o persoana ; 1:1
* Doctor🡨🡪 Doctor: Un doctor este o persoana; 1:1
* Doctor 🡪 Patient : Un doctor are mai multi pacienti ; 1:N
* Patient🡪 Doctor: Un Pacient are un singur doctor; 1:1
* Patient🡪Examination: Un pacient are mai multe examinari; 1:N
* Examination🡪Patient: O examinare apartine unui pacient; 1:1
* Examination🡪Analysis: O examinare are mai multe analize; 1:N
* Analysis🡪Examination: O analiza apartine unei singure consultatii: 1:1
* Analysis🡪SensorType: O analiza are un singur tip de sensor: 1:1
* SensorType🡪 Analysis: Un tip de sensor se poate folosi la mai multe analize 1:N

Dupa cum se poate observa in acesta structura nu exista redundanta a datelor si de asemenea au fost evitate relatiile multi la multi prin decizia de a nu permite ca un pacient sa aibe mai multi doctori.

Fiecare entitate are un set de attribute care se ptrivesc unor tipuri de date. Există o mulțime de tipuri diferite de date. Câteva dintre ele sunt standardizate, dar multe baze de date au propriile tipuri de date cu avantaje proprii. Unele baze de date oferă posibilitatea de a defini propriile tipuri de date, în cazul în care tipurile standard nu pot face lucrurile de care aveți nevoie.

Tipurile de date standard pe care le are fiecare bază de date și care sunt cele mai utilizate sunt: ​​CHAR, VARCHAR, TEXT, FLOAT, DOUBLE și INT.

Pentru text avem:

* CHAR (lungime) - include text (caractere, numere, semen de punctuatie). CHAR are caracteristica că salvează întotdeauna o sumă fixă ​​de poziții. Dacă definiți un CHAR (10) puteți salva maximum zece poziții, dar dacă utilizați doar două poziții, baza de date va salva încă 10 poziții. Celelalte opt poziții vor fi completate cu spații.
* VARCHAR (lungime) - include text (caractere, numere, semen de punctuatie). VARCHAR este același lucru cu CHAR, diferența fiind că VARCHAR foloseste doar spațiul necesar.
* TEXT - poate conține texte de o lungime foarte mare.

Pentru numere avem:

* INT - conține un număr întreg pozitiv sau negativ.
* FLOAT, DOUBLE - Aceeași lucru ca INT, dar poate și să stocheze numere în virgulă mobilă.

Alte tipuri:

* BLOB - pentru datele binare

Constrângerile SQL sunt utilizate pentru a specifica regulile pentru datele dintr-un tabel. Constrângerile sunt folosite pentru a limita tipul de date care pot intra într-o tabelă. Aceasta asigură precizia și fiabilitatea datelor din tabel. Dacă există vreo încălcare între constrângere și acțiunea de date, acțiunea este întreruptă.

Constrângerile pot fi nivel de coloană sau nivel de tabel. Constrângerile la nivel de coloană se aplică unei coloane, iar constrângerile la nivel de tabel se aplică întregului tabel. Următoarele constrângeri sunt utilizate în mod obișnuit în SQL:

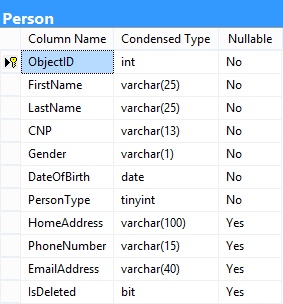
* NOT NULL - Asigură că o coloană nu poate avea o valoare NULL
* UNIQUE - Asigură că toate valorile dintr-o coloană sunt diferite
* CHEIE PRIMARĂ - O combinație de NOT NULL și UNIQUE. Identifică unic fiecare rând într-un table
* CHEIE EXTERN - identifică un rând / înregistrare într-un alt table
* CHECK - Asigură că toate valorile dintr-o coloană satisfac o anumită condiție
* DEFAULT - Setează o valoare implicită pentru o coloană când nu este specificată nicio valoare

Inainte de a prezenta structura fiecarui tabel, am scos elementele commune pentru a evita repetitia.

ObjectId este o coloana de tip IDENTITY aceasta insemna ca pote fi folosita pentru generarea valorilor. Proprietatea de identitate dintr-o coloană garantează ca fiecare valoare nouă este generată pe baza unui contor care este incrementat.

IsDeleted are o valoare implicita de 0, fapt ce marcheaza ca data este valida.

Tabela Person

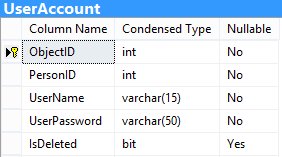


In acesta tabela sunt datele referitoare la o persoana, din figura se pot identifica campurile obligatorii, tipurile de date, numele atributelor si cheia primara. Mai trebuie adaugate informatii deoarece in imagine nu sunt afisate toate constrangerile.

Atributul CNP are o constrangere de unicitate, deoarece nu pot sa existe 2 persoane cu acelasi CNP.

Se face o verificare pe CNP si perechea formata din DateOfBirth si Gender pentru a evalua corectitudinea datelor

Tabela UserAcount.

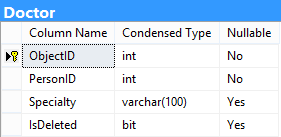


PersonId este o cheie externa catre ObjectId-ul din tabela Person, astfel fiecare cont va apartine unei personae din baza de date.

Campul UserName este un camp unic, astfel daca vrei sa adaugi un user deja existent vei primi un mesaj de eroare.

Pe campul UserPassword este aplicat un algoritm de criptare numit SHA-1. Criptarea se face cu ajutorul unui algoritm şi a unei chei de criptare. Algoritmul este o funcţie matematică folosită efectiv în procesul de criptare şi decriptare. Combinarea algoritmului cu o anume cheie de criptare dă un rezultat diferit de combinarea aceluiaşi algoritm cu o altă cheie de criptare.

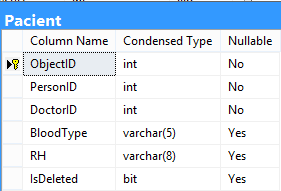
Tabela Doctor



PersonId este o cheie externa catre ObjectId-ul din tabela Person, astfel fiecare Doctor va fi o persona din baza de date.

Campul Specility este un camp extrem de important deoarece el va dicta sectile din spitalul pe care este instalata aplicatia.

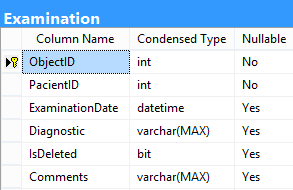
Tabela Person



PersonId este o cheie externa catre ObjectId-ul din tabela Person, astfel fiecare pacient va fi o persona din baza de date.

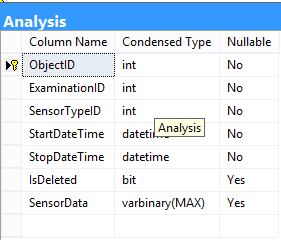
DoctorId este o cheie externa catre ObjectId-ul din tabela Doctor, astfel fiecare pacient o sa fie assignat unui doctor.

Tabela Examination



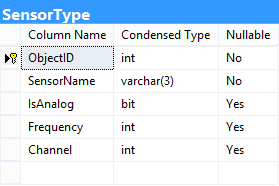
PatientId este o cheie externa catre ObjectId-ul din tabela Patient, astfel fiecare examinare va apartine unui pacient.

Tabela Analysis

 ExaminationId este o cheie externa catre ObjectId-ul din tabela Examination, astfel fiecare analiza va apartine unei examinari.

SensorId este o cheie externa catre ObjectId-ul din tabela Sensor, astfel fiecare analiza va folosi un tip de sensor.

Tabela Sensor



Procedurile stocate pot accesa sau modifica datele într-o bază de date, dar nu sunt legate de o bază de date sau un obiect specific, ceea ce oferă o serie de avantaje.

Procedura stocată oferă un nivel important de securitate între interfața de utilizator și baza de date. Aceasta susține securitatea prin intermediul controalelor de acces la date deoarece utilizatorii finali pot introduce sau modifica date, dar nu scriu proceduri. Procedura stocată păstrează integritatea datelor deoarece informațiile sunt introduse într-o manieră consecventă. Îmbunătățește productivitatea, deoarece declarațiile dintr-o procedură stocată trebuie să fie scrise o singură dată.

Procedurile stocate oferă avantaje față de integrarea interogărilor într-o interfață grafică (GUI). Deoarece procedurile stocate sunt modulare, este mai ușor să depanați când apare o problemă într-o aplicație. Procedurile stocate sunt, de asemenea, reglabile, ceea ce elimină necesitatea de a modifica codul sursă GUI pentru a-și îmbunătăți performanța. Este mai ușor să codificați procedurile stocate decât să construiți o interogare printr-o interfață grafică.

Procedurile stocate în SQL Server pot accepta parametrii de intrare și pot returna mai multe valori ale parametrilor de ieșire; În SQL Server, procedurile stocate proceduri declarații pentru a efectua operațiuni în baza de date și a returna o valoare de stare la o procedură de asteptare sau lot.

Orice operatie de se realizeaza intre baza de date si aplicatie este facuta cu ajutorul procedurilor stocate. Desemenea procedurile stocate confera un avantaj cand vine vorba de lizibilitatea codului, deoarece acestea avand nume suggestive sunt mult mai usor de inteles .