Основна ограничења

Ограничење коришћењем основног кључа омогућава јединственост редова и онемогућава појаву NULL као атрибута ограничења.

Сваки јединствени сет вредности у атрибутима ограничења се може појавити само једном у табели, само у једном реду.

Основни кључ се не може дефинисати у колони која допушта NULL; свака табела има само један основни кључ.

Страни кључ омогућава референциони интегритет.

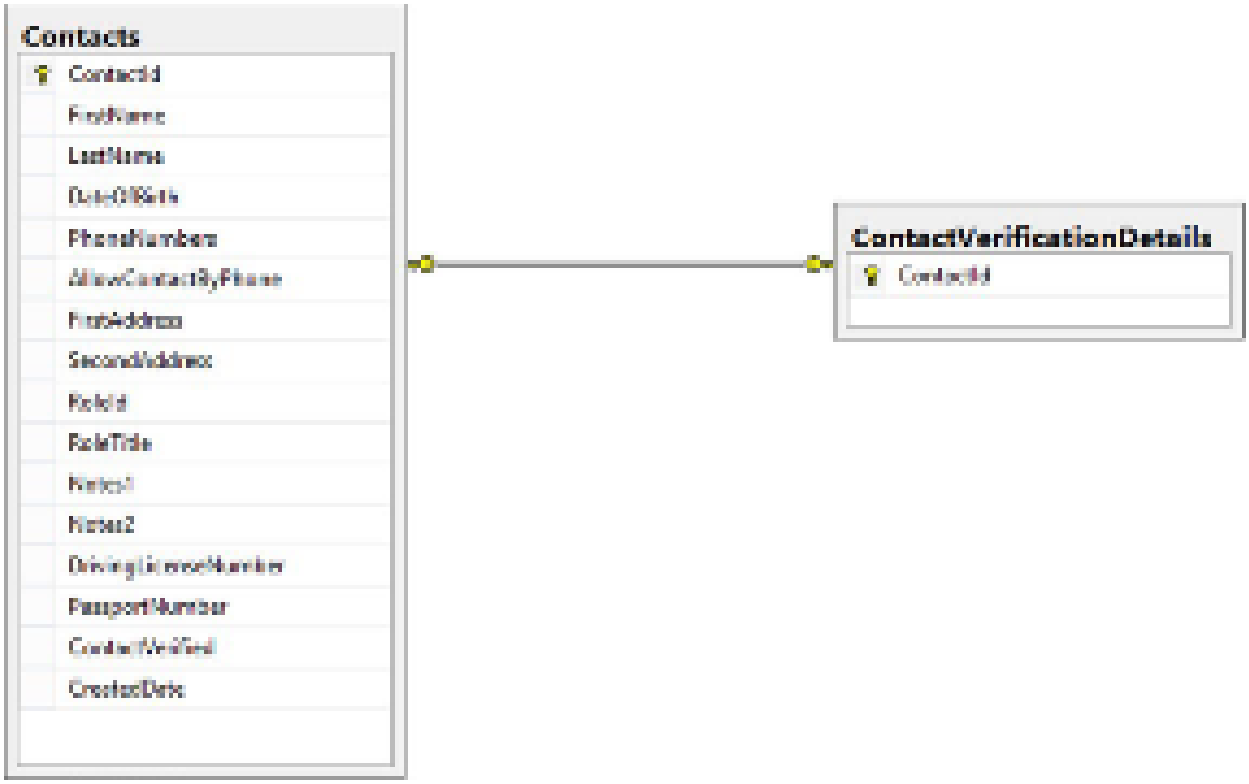
Ово ограничење је дефинисано на једном или више атрибута и упућује на кандидате кључеве.

Смисао страних кључева је да ограниче вредности које су допуштене у колонама страних кључева на оне које постоје у референтним колонама.

Страни кључеви и релације

Страни кључеви се користе за повезивање две табеле, једна табела је родитељ а друга је дете. Идеја да се направи једна велика табела са свим подацима је лоша посебно када је потребно генерисати извештаје или када треба пронаћи појединачне делове података.

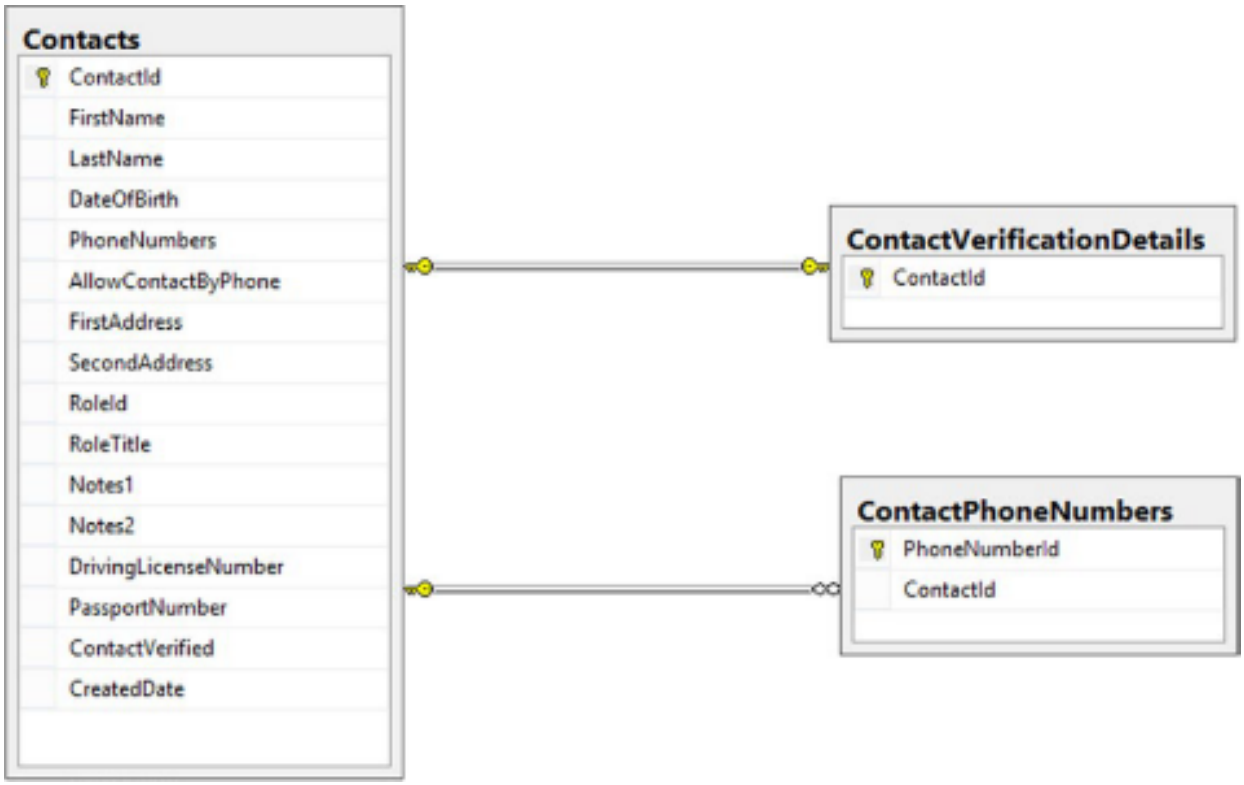
Боље је логички разделити податке у различите табеле, нпр. ако постоје две табеле Контакти и Бројеви Телефона Контаката, логично је да један контакт може имати више телефона, и тако закључујемо да је Контакти родитељ табела а Бројеви Телефона Контаката дете табела.



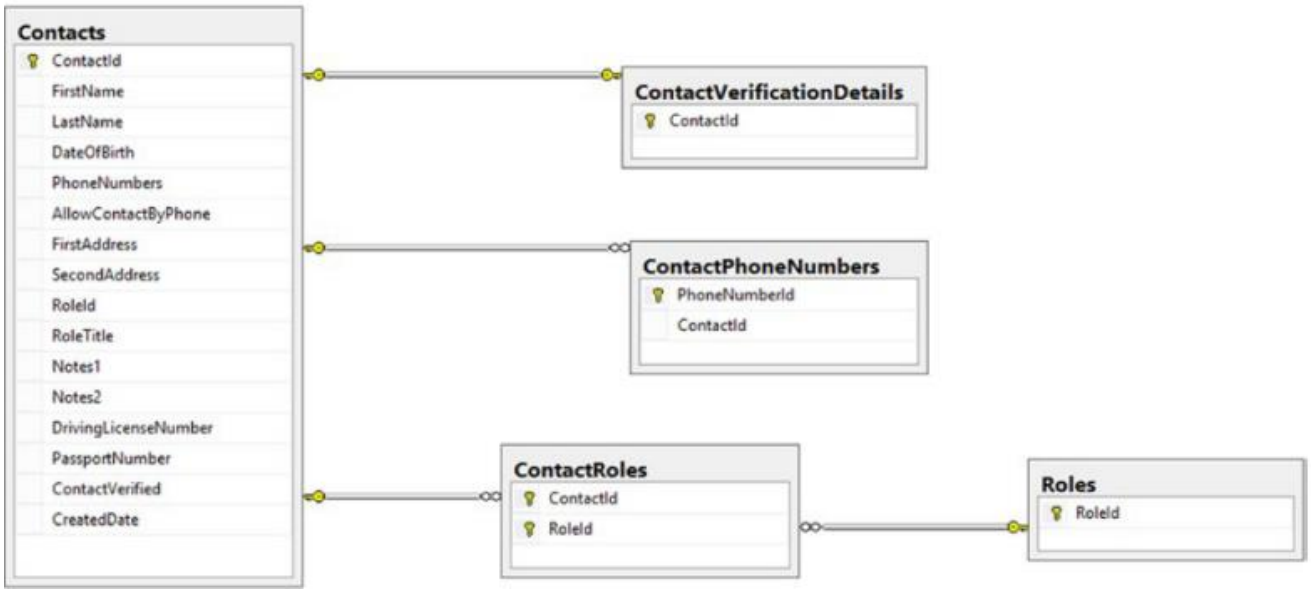
Релације између табела

Постоје релације између табела: један према један, један према више, више према један, више према више.

- један према један – користи се за поделу једне велике табеле на више мањих, ако имамо главну табелу и потребно је убацити додатне податке па уместо да се дода нова колона у главну табелу прави се нова табела и линкује се са главном табелом



- један према више или више према један – овде родитељски запис у једној табели има записе из дете табеле; записи из дете табеле не могу постојати без родитељског записа; пример је запис у Контакти као родитељске табеле, са Бројеви Телефона Контаката табелом која има дете записе; бројеви телефона нам нису корисни сем ако не знамо чији су; ово се сетује дефинисањем различитог основног кључа у свакој табели али укључујући Kontakt_ID колону (основни кључ родитељског записа) у обе табеле; онда се дефинише релација између ових колона као линк ових табела.



- више према више – најкомпликованија веза јер захтева три табеле за имплементацију да би се тачно поставила; шта ако су Контакти додељени табели Улоге? Постојећа колона Улоге би остала иста, али био били додељени другачијим контактима пошто се крећу по различитим пословима; једна улога би могла бити додељена више контаката, нпр имамо велики броја SQL Developera; то значи да један контакт може имати једну или више улога, док једна улога може бити додељена једном или више контаката; ово се зове дуалност; табеле се не могу повезати директно, пошто би то значило да Контакти или Улоге треба да буду родитељска табела, што није добро пошто су обе родитељи; решење је додавање табеле између њих, и која ће имати основне кључеве обе табеле као јединствену комбинацију; ова комбинација ће формирати компаунд кључ – основни кључ који се састоји од више од једне колоне (овде од две колоне). Овде је приказ један према више везе између Контакти и КонтактиУлоге, која је табела која уређује више према више релацију. Види се да кључ у овој табели има кључеве из обе табеле. Из Улоге иде слична релација. Зато КонтактиУлоге је табела која чува више према више релационих записа.

<u>Извршавање SELECT наредбе</u>

Редослед навођења свих клаузула у SELECT наредби је:

```
SELECT select_lista
FROM tabela_ili_pogled
WHERE uslov_za_filtriranje      --omogucava filtriranje redova prema uslovu
GROUP BY kolone_ili_izrazi      --uredjenje redova po grupama
HAVING uslov_za_filtriranje_grupa –filtrira grupe na osnovu uslova
ORDER BY lista_kolona; --sortiranje rezultata prema zeljenom redosledu
```

Редослед извршавања елемената у SELECT наредби:

1а – FROM припрема редове креирањем виртуелне табеле

1б – WHERE филтрира редове виртуелне табеле према услову и филтрирана виртуелна табела се прослеђује у следећем кораку

2а – GROUP BY организује редове у виртуелној табели према јединственим вредностима у листи GROUP BY; на основу тога креира се нова виртуелна табела која садржи списак група
2б – HAVING филтрира целе групе према услову; филтрирана је виртуелна табела
3а – SELECT одређује које колоне ће бити приказане у резултатима упита
3б – ORDER BY сортира редове на основу специфициране листе колона
Пример:

```
SELECT ProdavacID,
        YEAR(DatumNarudzbine) AS Godina,
        COUNT(*) As BrojPorudzbenica
FROM dbo.Narudzbine
WHERE MusterijaID = 1001
GROUP BY ProdavacID, YEAR(DatumNarudzbine)
HAVING COUNT(*) > 1
ORDER BY ProdavacID, Godina;
```

Елиминисање дуплих редова

SQL сервер подразумева да иза кључне речи SELECT се налази кључна реч ALL, а то значи да ће SQL сервер након извршења упита, вратити све одговарајуће редове табеле укључујући и дупле резултујуће редове.
Да би се из упита искључили дупли редови, потребно је да се иза кључне речи SELECT дода кључна реч DISTINCT:

- DISTINCT налаже SQL серверу да у резултујући скуп редова укључи само јединствене редове
- уклања дупликате на основу резултујуће листе колона
- обезбеђује јединственост преко скупа наведених колона

Пример: Када се изврши следећи упит, добија се списак од 40000 градова, постоје и дупла имена градова

```
SELECT GradIme
FROM dbo.Gradovi
ORDER BY GradIme;
```

Међутим, када се у упит укључи кључна реч DISTINCT, резултујући списак има 25000 градова, искључиво јединствених назива градова.

```
SELECT DISTINCT GradIme
FROM dbo.Gradovi
ORDER BY GradIme;
```

Коришћење псеудонима (алијаса) за колоне и табеле

Приликом преузимања података из табеле или погледа, SQL сервер ће именовати сваку колону на основу назива изворне колоне.
Ако је потребно, колоне могу бити преименоване употребом палијаса у SELECT клаузули.
Алијаси се могу користити да обезбеде прилагођене називе заглавља колона у SELECT листи, различите од назива изворних табела или да преименују називе табела у FROM клаузули.
Начини креирања алијаса:

- преко кључне речи AS која раздваја колону од њеног алијаса

```
SELECT DrzavaID AS Oznaka,
       DrzavaIme AS 'Ime zemlje'
FROM dbo.Drzave;
```

- коришћење знака =; када се користи = мора се успоставити следећи редослед: алијас, = и на крају назив колоне или израз

```
SELECT Oznaka = DrzavaID,
      'Ime zemlje' = DrzavaIme
FROM dbo.Drzave;
```

- алијас за колоне ставља се одмах после назива колоне

```
SELECT DrzavaID Oznaka,
       DrzavaIme 'Ime zemlje'
FROM dbo.Drzave;
```

Када се алијаси користе у FROM клаузули, тада указују на табелу или поглед.
Алијаси табела олакшавају читљивост и њихово коришћење је практично када упит укључује више табела.
Алијаси табела могу се дефинисати користећи AS или без ње.

```
SELECT DrzavaID, DrzavaIme
FROM dbo.Drzave AS D;

SELECT DrzavaID, DrzavaIme
FROM dbo.Drzave D;

SELECT D.DrzavaID, D.DrzavaIme
FROM dbo.Drzave AS D;
```

Задаци за самосталан рад

1. Из табеле dbo.StvariIzMagacina упитом приказати све вредности за колону ProizvodjacID (није основни кључ у StvariIzMagacina).

2. Из табеле dbo.StvariIzMagacina упитом приказати само јединствене вредности за колону ProizvodjacID.

3. У упит из претходног задатка додати колону StvariIzMagacinaID (основни кључ). Да ли ће се број резултујућих редова разликовати у односу на претходни пример?

4. Написати SELECT наредбу која враћа колоне BojaID и BojaNaziv из табеле dbo.Magacin. Користити алијас с за табелу. Користити алијас као префикс назива колона у SELECT листи.

5. Написати SELECT наредбу која враћа колоне MusterijaID, MusterijaIme, MusterijaKategorijaID, KreditLimit из табеле dbo.Musterije. Са циљем да извештаји садрже имена која корисници препознају, колонама доделити следеће алијасе: Sifra kupca, Naziv kupca, Sifra kategorije, Kreditni limit.