Кластеровани и некластеровани кључеви

Без обзира да ли се основни кључ односи на једну или више колона (основни кључеви на више колона се називају компаунд кључеви) могу бити кластеровани или некластеровани.

Кластеровани кључ одређује како се подаци смештају на хард диск, а некластеровани кључ подржава брз рад упита над подацима али не одређује како ће подаци бити смештени на хард диску.

Основни кључ је само специјални тип индекса.

По дифолту, кластеровани основни кључ ће бити направљен ако се експлицитно не захтева тип основног кључа. Може бити само један кластеровани индекс по табели и он је најчешће и основни кључ.

Некластерованих кључева може бити неограничено.

Експлицитно креирање некластерованог кључа:

CONSTRAINT PK_Kontakti PRIMARY KEY NONCLUSTERED (Kontakti_ID).

Ако се стави CLUSTERED или се изостави било шта, сматра се да је то кластероване основни кључ.

Страни кључеви и релације

Страни кључеви се користе за повезивање две табеле.

Једна табела је родитељ а друга је дете.

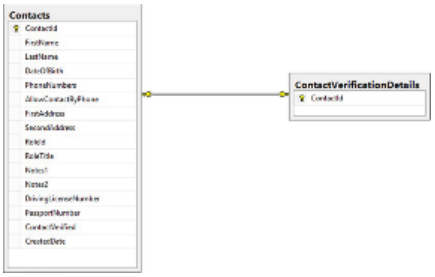
Идеја да се направи једна велика табела са свим подацима је лоша посебно када је потребно генерисати извештаје или када треба пронаћи појединачне делове података.

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.

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

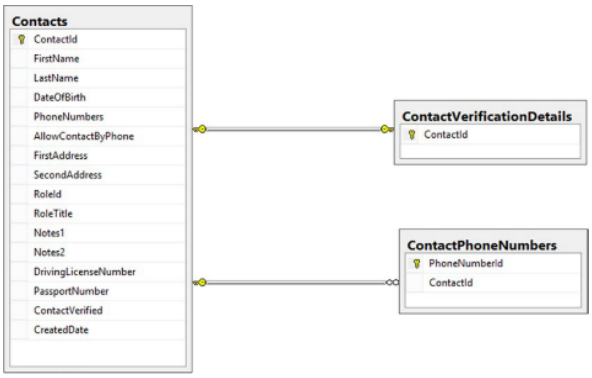
- један према један –



Користи се за поделу једне велике табеле на више мањих, ако имамо табелу Контакти и потребно је убацити податке о националностима па уместо да се дода нова колона у табелу прави се нова табела Детаљи Верификације Контаката и линкује се са табелом Контакти.

Ово се ради специфицирањем Контакт_ID као основног кључа, што ће се користити за повезивање табела; основни кључ се може наћи у табели тражењем имена колоне са нацртаним кључем до тог имена.

- један према много или много према један –



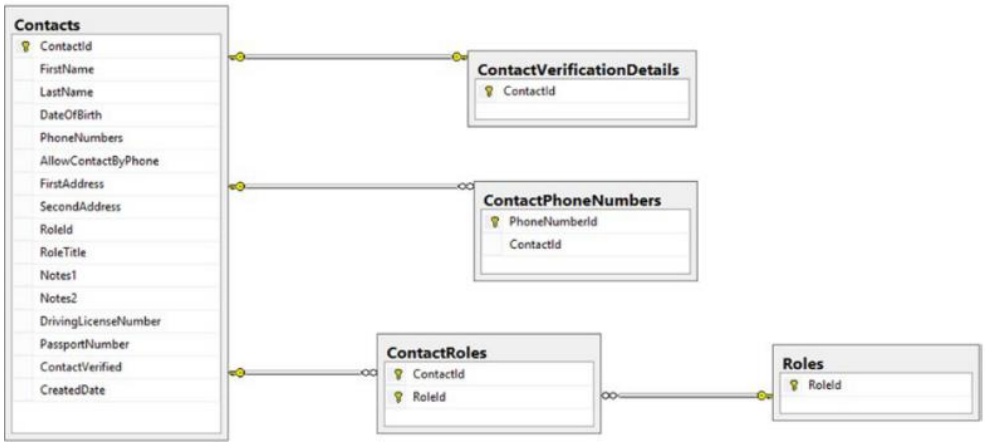
Овде родитељски запис у једној табели има записе из дете табеле; записи из дете табеле не могу постојати без родитељског записа;

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.

- много према много –



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 вби 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 многовезе између Контакти и КонтактиУлоге, коа је табела која уређује много према много релацију. Види се да кључ у овој табели има кључеве из обе табеле.
Из Улоге иде слична релација.
Зато КонтактиУлоге је табела која чува много према много релационих записа.

<u>Колона IDENTITY</u>
Ова колона аутоматски попуњава себе са бројчаним вредностима.
Програмер је конфигурише да би одредио који је почетни број и како се тај број повећава.
Може се направити једна колона IDENTITY по табели.
Скоро увек се постављају као основни кључеви.

Пример12: Претварање колоне Kontakt_ID у IDENTITY колону.

```
USE Adresar;

IF EXISTS (SELECT 1 FROM sys.tables WHERE [Name] = 'Kontakti')
BEGIN
    DROP TABLE dbo.Kontakti;
END;

CREATE TABLE dbo.Kontakti
 (
    Kontakt_ID INT IDENTITY(1,1),
    Ime VARCHAR(40),
    Prezime VARCHAR(40),
    Datum_Rodjenja DATE,
    Brojevi_Telefona VARCHAR(200),
    Dozvoljen_Kontakt_Telefonom BIT,
    Adresa_Prva VARCHAR(200),
    Adresa_Druga VARCHAR(200),
    Uloga_ID INT,
    Uloga_Titula VARCHAR(200),
    Beleska_1 VARCHAR(200),
    Beleska_2 VARCHAR(200),
    Broj_Vozacke_Dozvole VARCHAR(40),
    Broj_Pasosa VARCHAR(40),
    Potvrdjen_Kontakt BIT,
    Datum_Kreiranja DATETIME,
    CONSTRAINT PK_Kontakti PRIMARY KEY CLUSTERED (Kontakt_ID)
);

 GO
```

IDENTITY (seed, increment) где је seed почетни број, а increment диктира који ће бити следећи број; у скрипти се види да ће се кренути од броја 1 а следећи ће бити 2, па 3...(за овај случај могло је и само IDENTITY ()). Сачувати скрипту и отворити за преглед 200 редова.

	Kontakt_ID	Ime	Prezime	Datum_Ro.
	1	Mica	Micko	*NULL*
	2	Kica	Ki	*NULL*
	3	Pera	Petko	*NULL*
▸*	*NULL*	*NULL*	*NULL*	*NULL*

Почнимо да попуњавамо табелу. Није нам допуштено да било шта убацујемо у колону Контакт_ID која се аутоматски попуњава бројевима по редоследу од 1.

Пример13: Брисање редова у табели са IDENTITY колоном.
Десни клик на сиву ћелију пре броја 2 у Контакт_ИД и избор Delete.
После потврђивања избора:

	Kontakt_ID	Ime	Prezime	Datum_Ro...
	1	Mica	Micko	*NULL*
	3	Pera	Petko	*NULL*
▸*	*NULL*	*NULL*	*NULL*	*NULL*

IDENTITY колона не дозвољава поновно коришћење вредности. Зато не треба претпостављати да вредности у IDENTITY колони морају бити секвенцијалне (по утврђеном редоследу).
Брисање или случајно прескакање ће довести до изостајања неких вредности. За ову колону најбитније је јединственост вредности у њој.