

» Гл. ас. д-р Георги Чолаков  
» Бази от данни

**Правила за цялостност** ➤

# Въведение

- » Във всеки момент всяка БД съдържа определена конфигурация от стойности на данните - тази конфигурация отразява (е модел на) една реална ситуация;
- » Определени конфигурации нямат реален смисъл - те не представляват някакво възможно състояние на реалния свят;
- » Дефиницията на БД трябва да бъде разширена - тя трябва да включва правила за цялостност;
- » Тяхната цел е да информират БД за някои ограничения в реалния свят - напр., теглото не може да бъде отрицателно число;
- » Така могат да се избегнат някои конфигурации, които нямат смисъл в реалния свят.



# Въведение

БД са субект на многобройни правила за цялостност – обикновено те са зависими от приложната област.

Пример за таблица с книги от библиотека:

- > Не е реално цената да е отрицателно число - т.е. стойността ѝ трябва да е по-голяма от нула;
- > Годаината на публикуване не може да е по-голяма от текущата година;
- > ISBN номерът не се формира по произволен начин – генерира се по определени международни правила и е уникален за всяка книга.



# Въведение

Освен специфични правила за цялостност релационният модел включва два типа **общовалидни** правила за цялостност, валидни за **всяка** релационна БД:

- > Концепцията за **първични ключове**
- > Концепцията за **външни ключове**



# Ключове

Ключовете в релационния модел се използват за:

- » Уникална идентификация на ред от таблица;
- » Създаване на взаимоотношения между таблици и референциален интегритет.

Ще разгледаме:

- » Суперключ;
- » Кандидат-ключ;
- » Първичен ключ;
- » Алтернативен ключ;
- » Външен ключ.





# Зависимости

- » Ролята на ключа се базира на концепцията за определяне (изчисляване);
- » Определянето е процес, при който знаейки стойността на един атрибут определяме (намираме) стойността на друг атрибут;
- » Пример:

$$\text{ПРИХОД} - \text{РАЗХОД} = \text{ПЕЧАЛБА}$$

Тази формула е прост пример за това как знаейки стойностите на два атрибута можем да определим (изчислим) еднозначно стойността на трети атрибут.



# Зависимости

» Определянето, обаче, в БД не се базира на формула, а на взаимоотношения между атрибутите;

» Пример:

(ФАКУЛТЕТЕН НОМЕР, ДИСЦИПЛИНА) → ОЦЕНКА

Т.е. знаейки кой е студентът и коя е дисциплината ние можем еднозначно да разберем каква оценка е поставена на този студент за конкретната дисциплина – в тази формула ясно се вижда, че няма изчисления, а се касае за взаимоотношение между студент и дисциплина, което завършва с поставяне на оценка.



# Зависимости

fac_no	first_name	last_name	email	specialty
1801681001	Иван	Петров	ipetrov@mail.com	STD
1801321011	Силвия	Георгиева	silvito.g@hotmail.com	SE
1801261024	Стефи	Огнянова	fire_stefi@abv.bg	INF

В таблицата със студентите можем да открием някои взаимоотношения, напр. знаейки факултетния номер еднозначно можем да определим:

- » Име;
- » Фамилия;
- » E-mail;
- » Специалност.

Защото на всеки факултетен номер съответства точно по една стойност на всеки от изброените атрибути.





# Зависимости

Терминът за подобни взаимоотношения, базирани на определяне, се нарича **функционална зависимост**.

Нека:

- > R – релация;
- > X, Y – произволни подмножества на множеството на атрибутите на R.

Тогава:

- >  $X \rightarrow Y$  : “X функционално определя Y” или “Y функционално зависи от X”, ако и само ако всяка X-стойност в R е свързана точно с една Y-стойност в R за всички възможни стойности на R.



# Зависимости

Примери:

- » (студент, дисциплина) → оценка
- » фак. номер → (име, фамилия, email)
- » рег. номер → (марка автомобил, модел, собственик)

Атрибутът от лявата страна (който определя) се нарича детерминант, а вдясно (чиято стойност е определяна) - зависим.

Ако детерминантът се състои от повече от един атрибут се нарича съставен.



# Видове зависимости

» **Пълна функционална зависимост** - атрибут е напълно функционално зависим от множество атрибути ако и само ако той е функционално зависим от цялото това множество от атрибути и не е зависим от никое негово подмножество от атрибути.

Пример: нека имаме релацията EXAMS, съдържаща данни за проведените изпити, с идентификатор (FNO, SUBJ).

FNO	SUBJ	HALL	GRADE
1701261010	DB	547	6
1701261016	SE	446	5
1701261016	DB	547	4

**(FNO, SUBJ) → (GRADE)**

Фак. номер и дисциплината определят оценката. Това означава, че студентът може изкара добра оценка по дадена дисциплина, но не е сигурно, че ще изкара добри оценки по всички дисциплини, т.е. оценката зависи и от цялата комбинация студент и дисциплината, и не зависи от никой от тях поотделно.



# Видове зависимости

- » **Частична функционална зависимост** - такава е налице, когато само подмножество от съставен детерминант е достатъчно, за да определи функционално даден атрибут.
- » Пример: при провеждане на изпити те се насрочват в конкретни зали, дори за някои от тях има изискване за конкретна зала, и може да се каже, че залата зависи от конкретната дисциплина:
- »  $(FNO, SUBJ) \rightarrow (HALL)$ , т.е. фак. номер не играе роля за определянето, защото:
- »  $(SUBJ) \rightarrow (HALL)$

FNO	SUBJ	HALL	GRADE
1701261010	DB	547	6
1701261016	SE	446	5
1701261016	DB	547	4



# Видове зависимости

» Транзитивна функционална зависимост - транзитивна ФЗ съществува, когато има „преходна“ функционална зависимост. Т.е., ако  $A \rightarrow B$ , а  $B \rightarrow C$ , то  $A \rightarrow C$ .

FNO	NAME	CITY	ZIP
1701261010	Никола Михалев	Смолян	4700
1701261016	Евгения Михайлова	Пловдив	4000
1701261017	Димитър Георгиев	Ямбол	8600

Тук  $FNO \rightarrow CITY$ ,  $CITY \rightarrow ZIP$ , следователно  $FNO \rightarrow ZIP$  транзитивно.





# Суперключ

» Подмножество от атрибути на релация, което уникално идентифицира всяка  $n$ -торка от тялото ѝ, т.е. всеки ред от таблицата.

Суперключът определя функционално всеки атрибут на релацията.

Суперключовете представят ограничение, което предпазва две  $n$ -торки да имат съвпадащи стойности за тези атрибути, т.е. стойностите на суперключа за всеки ред трябва да са уникални.



# Кандидат-ключ

» Кандидат-ключ за дадена релация е минималното множество от атрибути, чиито стойности уникално идентифицират  $n$ -торка в тази релация – т.е. суперключ без атрибутите, които нямат отношение към уникалната идентификация.

Таблица може да има множество кандидат-ключове, от които се избира един за идентификатор, наречен първичен ключ.

Ключ-кандидат, който се състои от повече от един атрибут, се нарича съставен, а с един атрибут – прост.



# Първичен ключ

Нека  $R$  е релация с атрибути  $\{A_1, A_2, \dots, A_n\}$ . Множеството на атрибутите  $K = (A_i, A_j, \dots, A_k)$  от  $R$  е ключ-кандидат (candidate key) на  $R$ , ако то удовлетворява следните две, независещи от времето условия:

- > уникалност (uniqueness) - в един и същ момент не съществуват два различни записа на  $R$  с еднаква стойност за  $A_i, A_j, \dots, A_k$ ;
- > достатъчност (minimality) - никой от  $A_i, A_j, \dots, A_k$  не може да бъде премахнат, без това да наруши уникалността.

Интегритет на обект (entity integrity) е състояние, в което всеки ред от таблицата (всяка инстанция на обекта) може да бъде уникално идентифицирана. Затова първичният ключ трябва да гарантира уникалност, като в същото време всички съставлящи го атрибути трябва да имат стойност (различна от NULL).



# Алтернативен ключ

- » Ако една релация има повече от един ключ-кандидат релационният модел изисква един от тях да бъде избран за първичен;
- » Останалите кандидат-ключове се наричат алтернативни – в таблицата на тях може да им бъде наложено (или не) ограничение за уникалност на стойностите им, докато това автоматично се прилага за първичния ключ.





# Periodic Table of the Elements

Periodic Table of the Elements																	
1 1IA 1A											13 IIIA 3A	14 IVA 4A	15 VA 5A	16 VIA 6A	17 VIIA 7A	18 VIIIA 8A	
1 H Hydrogen 1.008	2 IIA 2A											5 B Boron 10.811	6 C Carbon 12.011	7 N Nitrogen 14.007	8 O Oxygen 15.999	9 F Fluorine 18.998	10 Ne Neon 20.180
3 Li Lithium 6.941	4 Be Beryllium 9.012	3 IIIB 3B	4 IVB 4B	5 VB 5B	6 VIB 6B	7 VIIB 7B	8 VIII 8	9 VIII 8	10 VIII 8	11 IB 1B	12 IIB 2B	13 Al Aluminum 26.982	14 Si Silicon 28.086	15 P Phosphorus 30.974	16 S Sulfur 32.066	17 Cl Chlorine 35.453	18 Ar Argon 39.948
19 K Potassium 39.098	20 Ca Calcium 40.078	21 Sc Scandium 44.956	22 Ti Titanium 47.867	23 V Vanadium 50.942	24 Cr Chromium 51.996	25 Mn Manganese 54.938	26 Fe Iron 55.845	27 Co Cobalt 58.933	28 Ni Nickel 58.693	29 Cu Copper 63.546	30 Zn Zinc 65.38	31 Ga Gallium 69.723	32 Ge Germanium 72.631	33 As Arsenic 74.922	34 Se Selenium 78.971	35 Br Bromine 79.904	36 Kr Krypton 83.789
37 Rb Rubidium 85.468	38 Sr Strontium 87.62	39 Y Yttrium 88.906	40 Zr Zirconium 91.224	41 Nb Niobium 92.906	42 Mo Molybdenum 95.95	43 Tc Technetium 98.907	44 Ru Ruthenium 101.07	45 Rh Rhodium 102.906	46 Pd Palladium 106.42	47 Ag Silver 107.868	48 Cd Cadmium 112.414	49 In Indium 114.818	50 Sn Tin 118.711	51 Sb Antimony 121.760	52 Te Tellurium 127.6	53 I Iodine 126.904	54 Xe Xenon 131.294
55 Cs Cesium 132.905	56 Ba Barium 137.328	57-71	72 Hf Hafnium 178.49	73 Ta Tantalum 180.948	74 W Tungsten 183.84	75 Re Rhenium 186.207	76 Os Osmium 190.23	77 Ir Iridium 192.217	78 Pt Platinum 195.085	79 Au Gold 196.967	80 Hg Mercury 200.592	81 Tl Thallium 204.383	82 Pb Lead 207.2	83 Bi Bismuth 208.980	84 Po Polonium [208.982]	85 At Astatine 209.987	86 Rn Radon 222.018
87 Fr Francium 223.020	88 Ra Radium 226.025	89-103	104 Rf Rutherfordium [261]	105 Db Dubnium [262]	106 Sg Seaborgium [266]	107 Bh Bohrium [264]	108 Hs Hassium [269]	109 Mt Meitnerium [278]	110 Ds Darmstadtium [281]	111 Rg Roentgenium [280]	112 Cn Copernicium [285]	113 Nh Nihonium [286]	114 Fl Flerovium [289]	115 Mc Moscovium [286]	116 Lv Livermorium [293]	117 Ts Tennessine [294]	118 Og Oganesson [294]
Lanthanide Series		57 La Lanthanum 138.905	58 Ce Cerium 140.116	59 Pr Praseodymium 140.908	60 Nd Neodymium 144.243	61 Pm Promethium 144.913	62 Sm Samarium 150.36	63 Eu Europium 151.964	64 Gd Gadolinium 157.25	65 Tb Terbium 158.925	66 Dy Dysprosium 162.500	67 Ho Holmium 164.930	68 Er Erbium 167.259	69 Tm Thulium 168.934	70 Yb Ytterbium 173.055	71 Lu Lutetium 174.967	
Actinide Series		89 Ac Actinium 227.028	90 Th Thorium 232.038	91 Pa Protactinium 231.036	92 U Uranium 238.029	93 Np Neptunium 237.048	94 Pu Plutonium 244.064	95 Am Americium 243.061	96 Cm Curium 247.070	97 Bk Berkelium 247.070	98 Cf Californium 251.080	99 Es Einsteinium [254]	100 Fm Fermium 257.095	101 Md Mendelevium 258.1	102 No Nobelium 259.101	103 Lr Lawrencium [262]	



# Външен ключ

## SPECIALTIES

code ↵	name	tax
STD	Софтуерни технологии и дизайн	680
SE	Софтуерно инженерство	700
INF	Информатика	640
BIT	Бизнес информационни технологии	600

Имаме таблици със специалностите и студентите.  
Искаме да асоциираме всеки студент с изучаваната от него специалност

## STUDENTS

fac_no	first_name	last_name	email
1801681001	Иван	Петров	ipetrov@mail.com
1801321011	Силвия	Георгиева	silvito.g@hotmail.com
1801261024	Стефи	Огнянова	fire_stefi@abv.bg



# Външен ключ

## SPECIALTIES

code ↵	name	tax
STD	Софтуерни технологии и дизайн	680
SE	Софтуерно инженерство	700
INF	Информатика	640
BIT	Бизнес информационни технологии	600

Можем просто в таблицата със студентите за всеки да добавим изучаваната специалност, дублирайки данните за специалността.

Или?

## STUDENTS

fac_no	first_name	last_name	email	code	name	tax
1801681001	Иван	Петров	ipetrov@mail.com	STD	Софтуерни технологии и дизайн	680
1801321011	Силвия	Георгиева	silvito.g@hotmail.com	SE	Софтуерно инженерство	700
1801261024	Стефи	Огнянова	fire_stefi@abv.bg	INF	Информатика	640

# Външен ключ

## SPECIALTIES

code ↩	name	tax
STD	Софтуерни технологии и дизайн	680
SE	Софтуерно инженерство	700
INF	Информатика	640
BIT	Бизнес информационни технологии	600

Можем просто в таблицата със студентите за всеки да добавим **ключа** на изучаваната специалност, минимизирайки излишеството и премахвайки дублирането на данните за специалностите.

## STUDENTS

fac_no ↩	first_name	last_name	email	code
1801681001	Иван	Петров	ipetrov@mail.com	STD
1801321011	Силвия	Георгиева	silvito.g@hotmail.com	SE
1801261024	Стефи	Огнянова	fire_stefi@abv.bg	INF



# Външен ключ

Външен ключ: нека  $R_1$  е базова релация. Външен ключ в  $R_1$  е едно подмножество на множеството от атрибутите  $\{A_1^R, A_2^R, \dots, A_n^R\}$  така, че:

- > съществува една базова релация  $R_2$  с ключ-кандидат;
- > във всеки един момент стойностите на външния ключ в  $R_1$  са измежду стойностите на кандидат-ключа в някой запис на  $R_2$ .

Важно е да се отбележи, че не е задължително  $R_1$  и  $R_2$  да са различни релации.



# Външен ключ

## SPECIALTIES

code ↩	name	tax
STD	Софтуерни технологии и дизайн	680
SE	Софтуерно инженерство	700
INF	Информатика	640
BIT	Бизнес информационни технологии	600

Проверка дали STUDENTS.code отговаря на дефиницията:

- ✓ Подмножество от атрибутите е;
- ✓ Стойностите му са измежду стойностите на референцирания първичен ключ.

## STUDENTS

fac_no ↩	first_name	last_name	email	code
1801681001	Иван	Петров	ipetrov@mail.com	STD
1801321011	Силвия	Георгиева	silvito.g@hotmail.com	SE
1801261024	Стефи	Огнянова	fire_stefi@abv.bg	INF





# Външен ключ

В дефиницията беше подчертано, че двете релации  $R_1$  и  $R_2$  **не е необходимо да се различават**.

Като пример за това можем да споменем релацията EMPLOYEES, в която се съхраняват данните за служителите на определена компания, включително с йерархията им.

Това е типична дървовидна структура, която може да бъде съхранена в таблица.



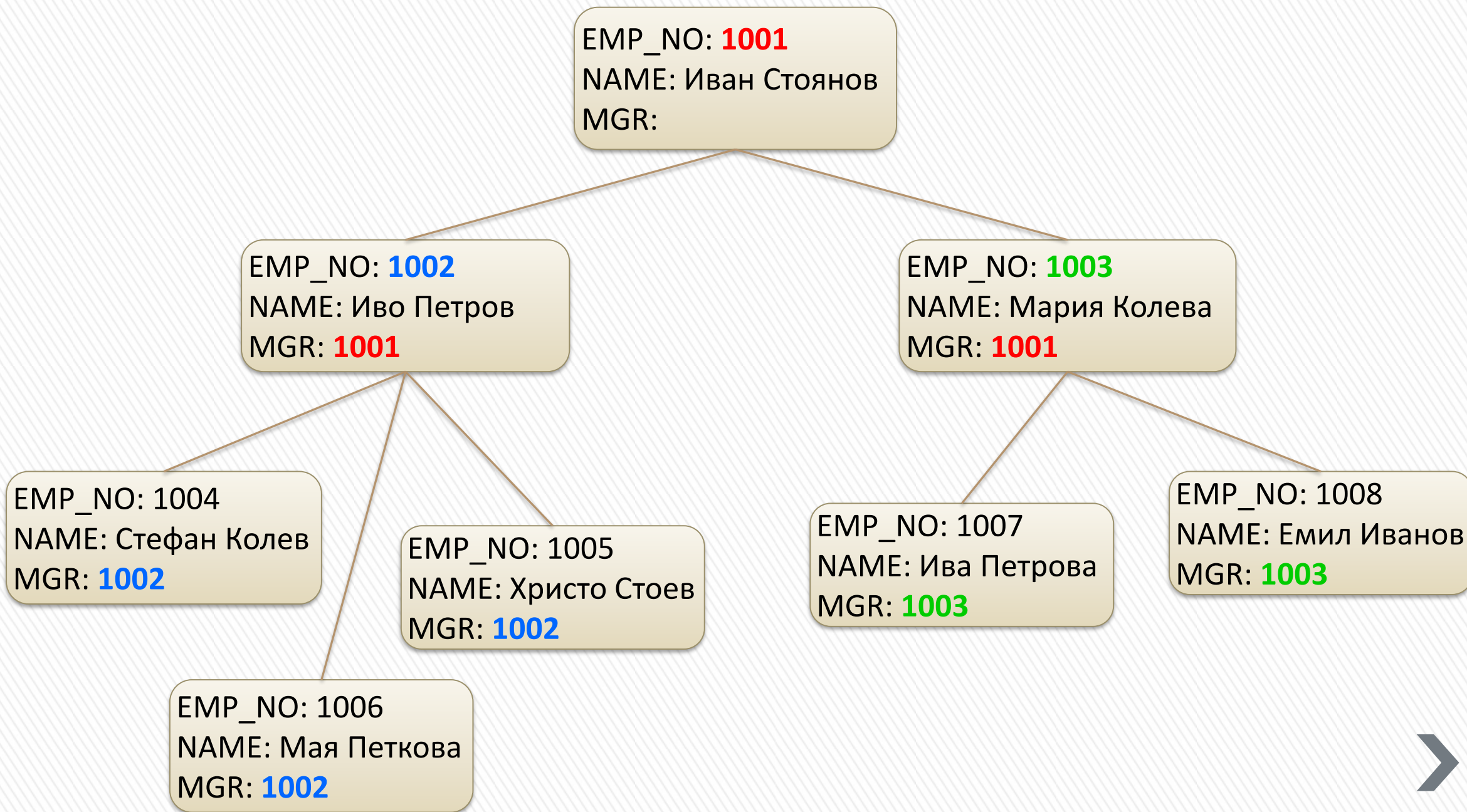
# Външен ключ

В дефиницията беше подчертано, че двете релации  $R_1$  и  $R_2$  **не е необходимо да се различават**.

Като пример за това можем да споменем релацията EMPLOYEES, в която се съхраняват данните за служителите на определена компания, включително с йерархията им.

Това е типична дървовидна структура, която може да бъде съхранена в таблица.





## EMPLOYEES

EMP_NO	NAME	SALARY	MGR
1001	Иван Стоянов	800	
1002	Иво Петров	630	1001
1003	Мария Колева	600	1001
1004	Стефан Колев	430	1002
1005	Христо Стоев	450	1002
1006	Мая Петкова	390	1002
1007	Ива Петрова	540	1003
1008	Емил Иванов	300	1003

Тук EMP\_NO е първичният ключ, а MGR е външният ключ, който референцира първичния ключ на същата релация EMPLOYEES.

Логиката е следната – служителът с номер 1001 е мениджър на служителите с номера 1002 и 1003.



# Свойства на външните ключове

- » ВК представлява подмножество от атрибути на релация;
- » По дефиниция всяка стойност на даден ВК във всеки един момент трябва да е измежду стойностите на съответния референциран първичен ключ;
- » Обратното не се изисква - т.е. КК може да има стойност, която не се появява във ВК;
- » Ако даден ред е референциран от ред(ове) от друга релация, то изтриването му се предотвратява от ВК (освен ако не е указано друго);
- » Един ВК ще бъде съставен, ако кореспондиращият КК е също съставен; ако КК е прост, ВК също ще е прост;
- » Всеки атрибут на един ВК трябва да бъде дефиниран върху същия домейн, върху който е дефиниран кореспондиращият атрибут от референцирания КК.





# Пример за съставен външен ключ

## RECIPES

author ↗	recipe_name	date_published	type
Иван Петров	Кюфтета от тиквички	12.10.2019	Вегетарианска
Мария Иванова	Боб с наденица	29.08.2019	Месна

## INGREDIENTS

product_name	quantity	author	recipe_name
Тиквички	500 гр.	Иван Петров	Кюфтета от тиквички
Сирене	200 гр.	Иван Петров	Кюфтета от тиквички
Фасул	300 гр.	Мария Иванова	Боб с наденица
Наденица	200 гр.	Мария Иванова	Боб с наденица
Домати	250 гр.	Мария Иванова	Боб с наденица

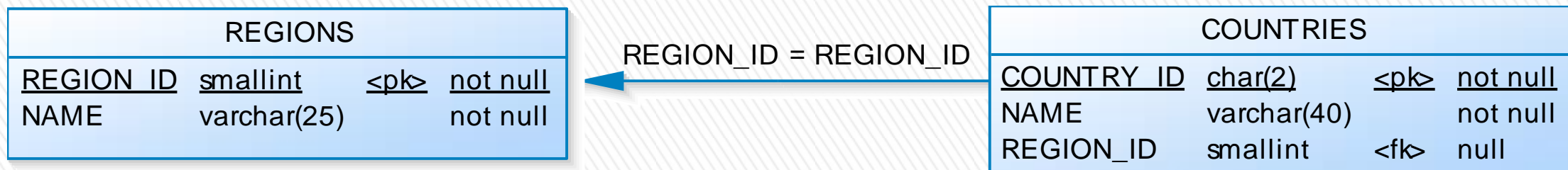
- » Атрибутите, съставлящи първичния ключ, мигрират в таблицата, където ще бъдат външен;
- » RECIPES.author и INGREDIENTS.author трябва да бъдат от един и същ тип;
- » RECIPES.recipe\_name и INGREDIENTS.recipe\_name трябва да бъдат от един и същ тип;
- » Обикновено в такава ситуация се добавя една колона в master таблицата, която да играе роля на прост ключ, за да се избегнат усложненията от съставния ключ.

- » Референциална цялостност: ситуация, в която БД не съдържа невалидни стойности на ВК.
- » Референциално ограничение: ограничението, че стойностите на един ВК трябва да съответстват на стойностите на съответния ПК.
- » Релация:
  - > референцираща - релацията, която съдържа ВК;
  - > референцирана (целева) - релацията на ПК.



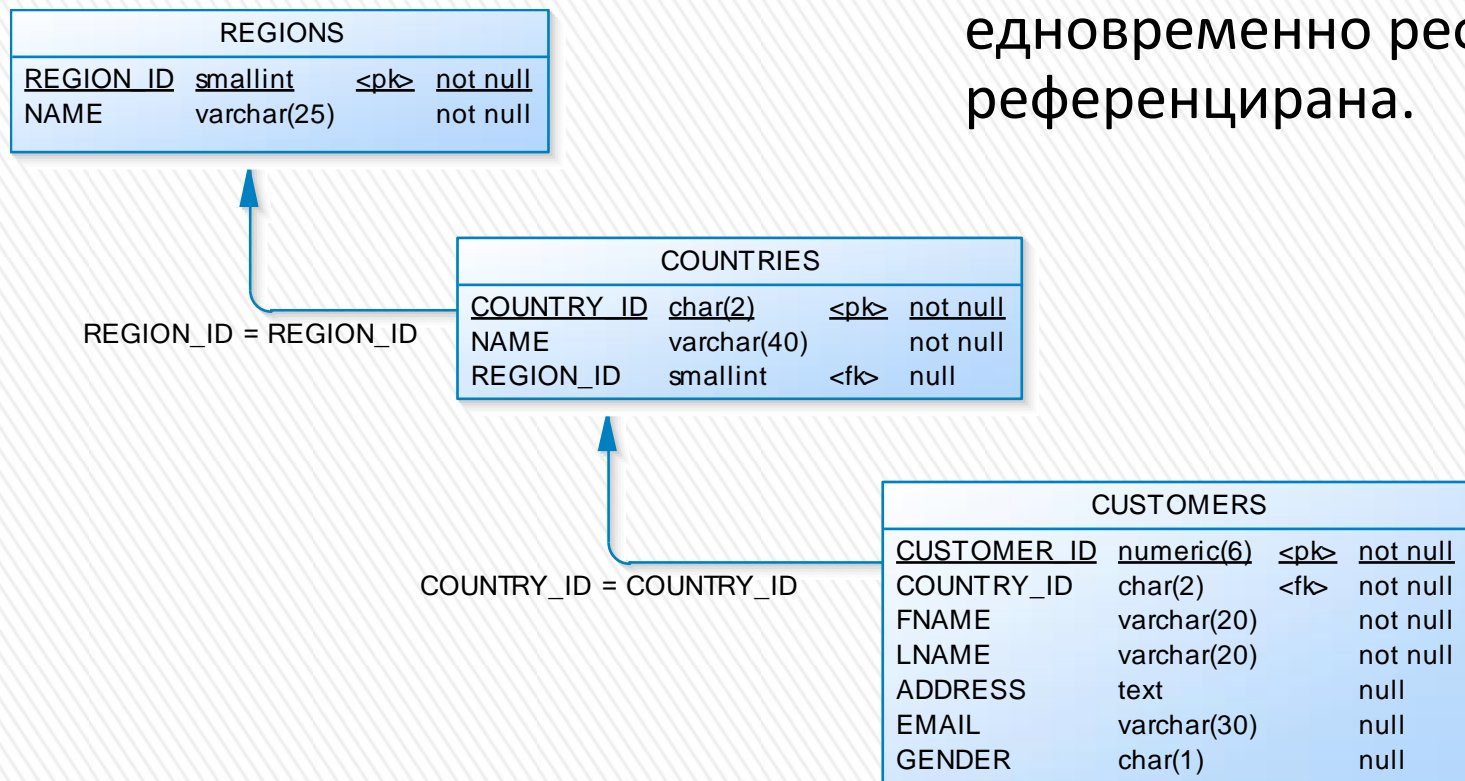
# Референциални диаграми

- » Нека разгледаме таблиците за държави и региони отново. Можем да представим референциалните ограничения чрез референциалната диаграма на фигурата;
- » Всяка стрелка показва наличие на външен ключ в релацията, от която тръгва, референциращ съответен първичен ключ в релацията, към която сочи. Добра идея е да се указват атрибутите, съставлящи външния ключ.



# Референциални диаграми

- » Референциалните диаграми се използват за представяне на референциални ограничения;
- » Една и съща релация може да бъде едновременно референцираща и референцирана.



# Референциален път

- » Нека релациите от  $R_n$  до  $R_1$  са такива, че има референциално ограничение от  $R_n$  към  $R_{n-1}$ , от  $R_{n-1}$  към  $R_{n-2}$  и т.н. до  $R_1$ .
- » Последователността от референциални ограничения от  $R_n$  до  $R_1$  представя **референциален път** от  $R_n$  до  $R_1$ .



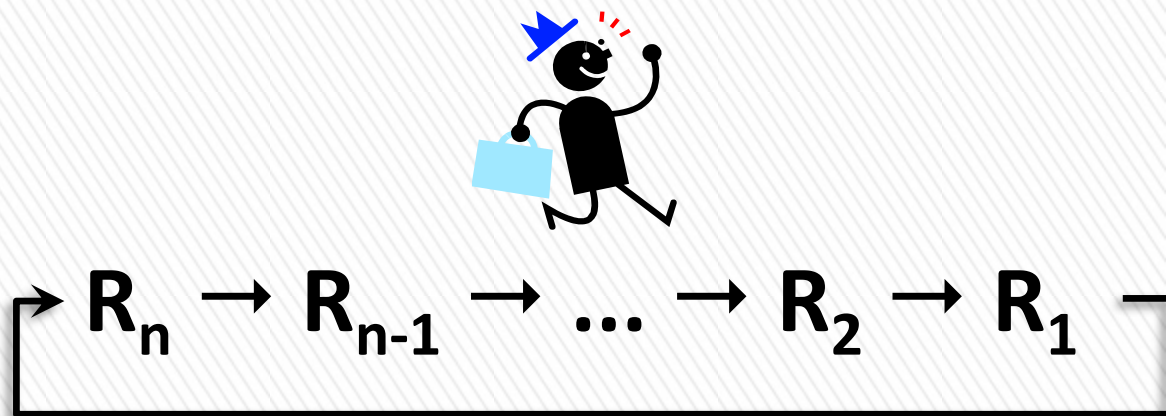
$$R_n \rightarrow R_{n-1} \rightarrow \dots \rightarrow R_2 \rightarrow R_1$$





# Референциален цикъл

- » Нека релациите от  $R_n$  до  $R_1$  са такива, че има референциално ограничение от  $R_n$  към  $R_{n-1}$ , от  $R_{n-1}$  към  $R_{n-2}$  и т.н. до  $R_1$ , като  $R_1$  също има външен ключ към  $R_n$ .  
Така представени, релациите формират **референциален цикъл**.
- » Самореференциращите се релации са частен случай на референциален цикъл.
- » По-точно, референциален цикъл съществува, ако има референциален път от  $R_n$  до  $R_n$ .



# Изтриване на редове при референциален цикъл

## EMPLOYEES

EMP_NO	NAME	SALARY	MGR
1001	Иван Стоянов	800	
1002	Иво Петров	630	1001
1003	Мария Колева	600	1001
1004	Стефан Колев	430	1002
1005	Христо Стоев	450	1002
1006	Мая Петкова	390	1002
1007	Ива Петрова	540	1003
1008	Емил Иванов	300	1003

- » Ако е налице референциален цикъл изтриването на редове от таблиците в него може да се окаже проблемно;
- » Например, как да изтрием реда с EMP\_NO = 1003, след като той е референциран от други редове?



# Изтриване на редове при референциален цикъл

## EMPLOYEES

EMP_NO	NAME	SALARY	MGR
1001	Иван Стоянов	800	
1002	Иво Петров	630	1001
1003	Мария Колева	600	1001
1004	Стефан Колев	430	1002
1005	Христо Стоев	450	1002
1006	Мая Петкова	390	1002
1007	Ива Петрова	540	1003->1002
1008	Емил Иванов	300	1003->1002

» Можем да прехвърлим служителите с MGR = 1003 под ръководството на 1002, след което да изтрием ред 1003.



# Правила за външните ключове

- » Правило за референциалната цялостност: БД не трябва да съдържа стойности на ВК, които нямат съответна стойност на референцирания ПК в целевата релация;
- » Всяко състояние на БД, което не удовлетворява правилото за референциалната цялостност, по дефиниция е некоректно;
- » Но правилото не казва как да се предпазим от такива некоректни състояния.

## Възможности:

- > Системата да отхвърли всяка операция, която би довела до некоректно състояние, ако бъде изпълнена;
- > Системата извършва операцията, а след това извършва допълнителни компенсиращи операции.



» За всеки ВК съществуват два основни въпроса, на които трябва да се отговори:

1. Какво ще се случи при опит да се изтрие целевата на ВК референция?

Например при опит да се изтрие регион, за който съществува поне една държава.

Съществуват два подхода:

- » **RESTRICTED** - операцията изтриване е ограничена само за случая, когато няма свързани данни; ако такива са налице, изтриване не се извършва;
- » **CASCADES** - операцията се разширява каскадно и изтрива също така и свързаните записи.





# RESTRICTED

Операцията се  
отхвърля




REGION_ID	REGION_NAME
1	Eastern Europe
2	Americas
3	Asia
4	Middle East and Africa
5	Western Europe



COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
BE	Belgium	5
BG	Bulgaria	1
US	United States of America	2
GR	Greece	1





# CASCADES



REGION_ID	REGION_NAME
1	Eastern Europe
2	Americas
3	Asia
4	Middle East and Africa
5	Western Europe

COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
BE	Belgium	5
BG	Bulgaria	1
US	United States of America	2
GR	Greece	1



Операцията се  
разширява  
каскадно



2. Какво ще стане при опит да се промени стойността на един ПК, който е целева референция на ВК? Например - опит да се промени идентификатор на регион, за който съществува поне една държава.

Отново се използват същите два подхода:

- » **RESTRICTED** - операцията промяна е ограничена само за случая, когато няма държава за региона; ако такава е налице, промяна не се извършва;
- » **CASCADES** - операцията се разширява каскадно и променя също така и записа за държавата.



# RESTRICTED

Операцията се  
отхвърля



REGION_ID	REGION_NAME	POPULATION
100	Eastern Europe	500 000 000
2	Americas	1 000 000 000
3	Asia	2 500 000 000
4	Middle East and Africa	1 000 000 000
5	Western Europe	500 000 000



COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
BE	Belgium	5
BG	Bulgaria	1
US	United States of America	2
GR	Greece	1



# CASCADES



REGION_ID	REGION_NAME	POPULATION
100	Eastern Europe	500 000 000
2	Americas	1 000 000 000
3	Asia	2 500 000 000
4	Middle East and Africa	1 000 000 000
5	Western Europe	500 000 000



COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
BE	Belgium	5
BG	Bulgaria	100
US	United States of America	2
GR	Greece	100



Операцията се  
разширява каскадно





# Забележка 1

- » Опциите за правилата DELETE и UPDATE за ВК не изчерпват възможностите - те по-точно представят най-общите, които се изискват от практиката.
- » По принцип съществуват повече възможности. Например, опит за изтриване на един регион възможностите могат да бъдат:
  - > инициализира се диалог с краен потребител за потвърждение;
  - > информацията може да се архивира (какво ще се прави);
  - > държавата може да се причисли към друг регион.

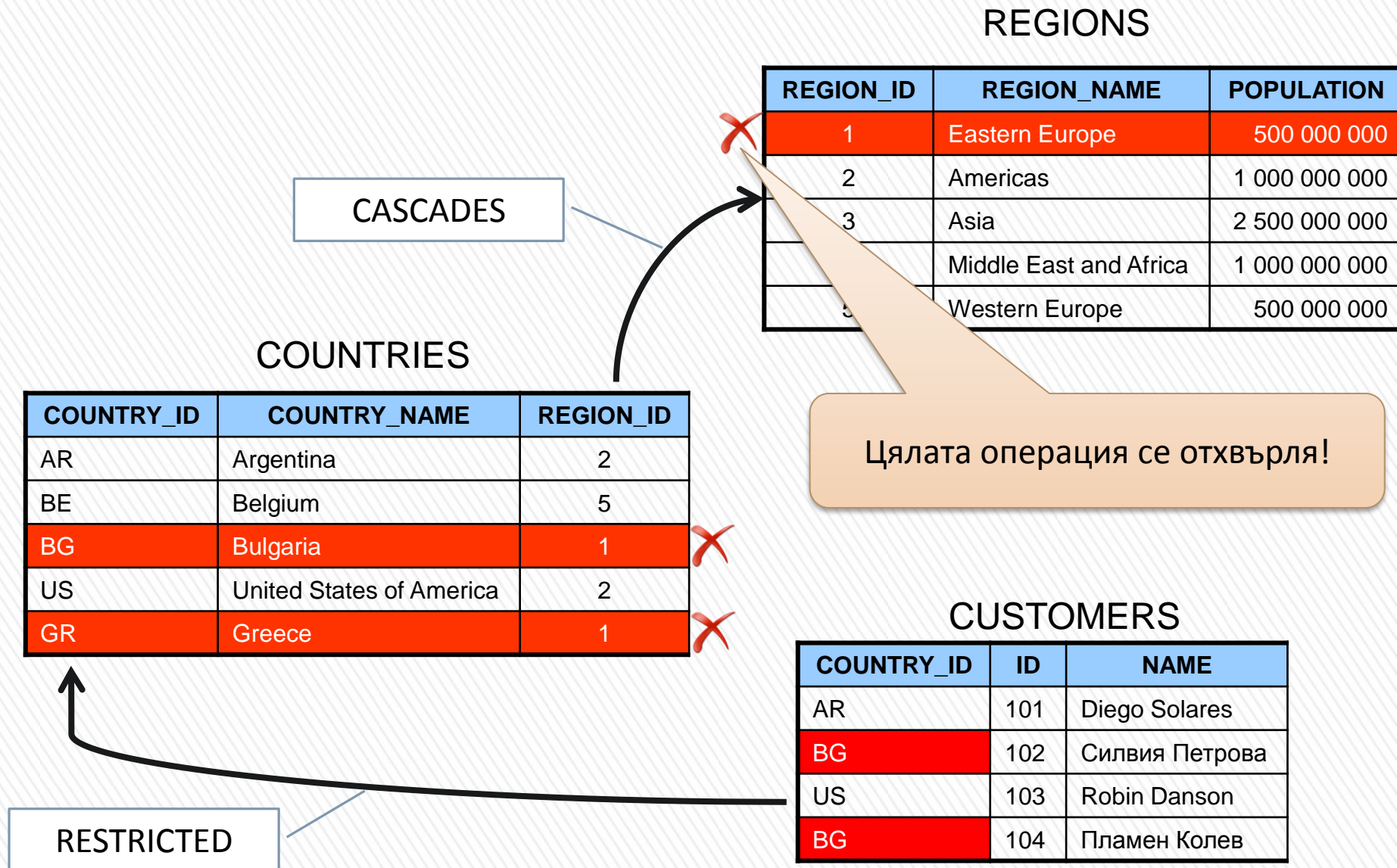


## Забележка 2

- » Нека  $R_2$  и  $R_1$  са референцираща и целева  $R_2 \rightarrow R_1$  и нека delete-правилото за това референциално ограничение е **CASCADES**;
- » Следователно изтриването на  $n$ -торка в  $R_1$  ще предизвика изтриване на  $n$ -торка в  $R_2$ ;
- » Да допуснем, че  $R_2$  е целева за  $R_3$ :  $R_3 \rightarrow R_2 \rightarrow R_1$ . Тогава изтриването на  $n$ -торка зависи и от delete-правилото между  $R_3$  и  $R_2$ . Ако то забранява изтриване (**RESTRICTED**), тогава не се изтрива нищо и БД остава непроменена. И т.н. до произволно ниво.



# CUSTOMERS → COUNTRIES → REGIONS



# Задача от интервю

- » Да се провери налице ли е референциална цялостност в представената база от данни с SQL команда.

REGIONS			
<u>REGION_ID</u>	<u>smallint</u>	<u>&lt;pk&gt;</u>	<u>not null</u>
NAME	varchar(25)		not null

COUNTRIES			
<u>COUNTRY_ID</u>	<u>char(2)</u>	<u>&lt;pk&gt;</u>	<u>not null</u>
NAME	varchar(40)		not null
REGION_ID	smallint	<fk>	null



# Задача от интервю

- » Да се провери налице ли е референциален интегритет в представената база от данни с SQL команда.

REGIONS			
<u>REGION_ID</u>	<u>smallint</u>	<u>&lt;pk&gt;</u>	<u>not null</u>
NAME	varchar(25)		not null

COUNTRIES			
<u>COUNTRY_ID</u>	<u>char(2)</u>	<u>&lt;pk&gt;</u>	<u>not null</u>
NAME	varchar(40)		not null
REGION_ID	smallint	<fk>	null





# Задача от интервю - решение

- » **Какво е референциален интегритет?** Състояние на БД, в която няма стойности на външни ключове, които не се срещат сред стойностите на референцираните от тях първични.
- » Значи трябва да проверим има ли държави със стойност в атрибута **COUNTRIES.region\_id**, която да не се среща сред стойностите на **REGIONS.region\_id**.
- » Ако има такива – няма референциален интегритет! Ако няма – налице е референциален интегритет.

REGIONS			
<u>REGION_ID</u>	smallint	<pk>	not null
NAME	varchar(25)		not null

COUNTRIES			
<u>COUNTRY_ID</u>	char(2)	<pk>	not null
NAME	varchar(40)		not null
REGION_ID	smallint	<fk>	null



# Задача от интервью - решение

```
SELECT *  
FROM COUNTRIES  
WHERE region_id NOT IN (SELECT region_id  
                        FROM REGIONS)
```

Results Messages			
	COUNTRY_ID	NAME	REGION_ID

Results Messages			
	COUNTRY_ID	NAME	REGION_ID
1	AR	Аржентина	20
2	CN	Китай	30
3	DK	Дания	50



# Нулеви стойности

- » Нулева стойност: липса на стойност (данни) за даден атрибут.
- » В реалния свят има много такива ситуации - напр. неизвестен адрес;
- » В релационния модел е приет един специален маркер за означаване на липсваща информация – NULL;
- » Един атрибут може да съдържа или може да му бъде забранено да съдържа нулеви стойности.



# Външни ключове и нулеви стойности

- » Да се върнем на примера “Държава-Регион”;
- » Възможно е някоя държава да не е причислена към конкретен географски регион  $\Rightarrow$  допуска се ВК да има нулеви стойности.

Забележка: за всеки ВК конструкторът на БД трябва да реши дали да се допускат нулеви стойности или не в зависимост от конкретната логика.



- » Да се върнем на въпроса какво ще стане, ако искаме да изтрием n-торка от целевата релация, например, да изтрием регион, за който съществува държава:
  - > Ако за BK са разрешени NULL - съществува нова трета възможност – NULLIFIES - т.е. BK получава стойност NULL и тогава се изтрива основната n-торка;
  - > Същото важи и за промяната.





# NULLIFIES

Записът се  
изтрива.



REGION_ID	REGION_NAME	POPULATION
1	Eastern Europe	500 000 000
2	Americas	1 000 000 000
3	Asia	2 500 000 000
4	Middle East and Africa	1 000 000 000
5	Western Europe	500 000 000



COUNTRY_ID	COUNTRY_NAME	REGION_ID
AR	Argentina	2
BE	Belgium	5
BG	Bulgaria	NULL
US	United States of America	2
GR	Greece	NULL



Стойността на ВК се  
променя на NULL.



# NULL и логическите оператори

- » SQL има три логически стойности: TRUE, FALSE и UNKNOWN;
- » Стойността UNKNOWN произлиза в резултат от използването на NULL в сравнения с други предикати, но UNKNOWN е логическа стойност и не е еквивалентна на NULL, което е маркер за липса на стойност;
- » Ето защо в SQL пишем

**x IS [NOT] NULL**

вместо

**x = NULL**



# NULL и операторите

<b>AND</b>	TRUE	UNKNOWN	FALSE
TRUE	TRUE	UNKNOWN	FALSE
UNKNOWN	UNKNOWN	UNKNOWN	FALSE
FALSE	FALSE	FALSE	FALSE

<b>OR</b>	TRUE	UNKNOWN	FALSE
TRUE	TRUE	TRUE	TRUE
UNKNOWN	TRUE	UNKNOWN	UNKNOWN
FALSE	TRUE	UNKNOWN	FALSE

<b>x</b>	<b>NOT</b>
TRUE	FALSE
UNKNOWN	UNKNOWN
FALSE	TRUE

**X \* UNKNOWN = UNKNOWN**  
**X / UNKNOWN = UNKNOWN**  
**X + UNKNOWN = UNKNOWN**  
**X - UNKNOWN = UNKNOWN**  
**X % UNKNOWN = UNKNOWN**

