

УНИВЕРЗИТЕТ У БЕОГРАДУ
ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА
КАТЕДРА ЗА СОФТВЕРСКО ИНЖЕЊЕРСТВО

Релациони упитни језик

Прављење табела и постављање упита у
језику Transact-SQL

Саша Д. Лазаревић

6. фебруар 2017.

Сажетак: У првом делу овог списка објашњени су појмови модела података и релационог модела података, а у другом је изложен поступак креирања табела и постављања упита у релационом језику Transact-SQL.

САДРЖАЈ

1. МОДЕЛ ПОДАТАКА.....	3
2. БАЗА ПОДАТАКА. СИСТЕМ ЗА УПРАВЉАЊЕ БАЗАМА ПОДАТАКА	5
3. РЕЛАЦИОНИ МОДЕЛ ПОДАТАКА	7
3.1. Концепција релационог модела података.....	7
3.1.1. Независност: логичка и физичка.....	7
3.1.2. Једноставност: табела као јединица структуре података	7
3.1.3. Декларативност: упитни језик.....	7
3.2. Структурална компонента релационог модела података.....	7
3.2.1. Релација.....	8
3.2.2. Домени атрибута.....	10
3.2.3. Кључеви релације	10
3.2.4. Схема релационе базе и релационија база	12
3.2.5. NULL вредност	13
3.3. Операциона компонента релационог модела података.....	13
3.4. Интегритетна компонента релационог модела података	14
3.4.1. Ограничења	14
3.4.2. Правила интегритета	15
3.5. Кодова релационија правила.....	19
4. РЕЛАЦИОНИ УПИТНИ ЈЕЗИК: STRUCTURED QUERY LANGUAGE (SQL)	21
4.1. Врсте SQL наредби.....	22
4.2. SQL-SCHEMA STATEMENTS	22
4.2.1. SQL типови података	22
4.2.2. Домени	23
4.2.3. Табеле и колоне	23
4.2.4. Индекси.....	24
4.2.5. Схеме.....	24
4.3. SQL-DATA-MANIPULATION STATEMENTS	25
4.4. SQL-DATA-RETRIEVE STATEMENTS	27
4.4.1. Поступак извршавања SQL упита	27
4.4.2. Синтакса наредбе SELECT	28
5. РЕФЕРЕНЦЕ.....	29
ЛИТЕРАТУРА.....	29
ИНТЕРНЕТ	29
6. ПРИЛОГ	30
6.1. Концептуализација	30
6.1.1. Вербални модел	30
6.1.2. Концептуални модел	30
6.1.2.1. Структура.....	30
А) Дијаграм објекта и веза.....	30
6.1.2.2. Ограничења	31
6.2. Спецификација	32
6.2.1. Схема релационе базе података: базне релације	32
6.2.2. Схема релационе базе података: изведене релације	32
6.2.3. Правила интегритета	32
6.3. Имплементација	32
6.3.1. DDL	32
6.3.2. DML	34
6.3.3. RDB	35

6.3.4. <i>ADS</i>	36
6.3.5. <i>DRL</i>	39
1. УПИТИ НАД ЈЕДНОМ ТАБЕЛОМ КОЈИМА СЕ ПРИКАЗУЈЕ ЊЕН НЕИЗМЕЊЕН САДРЖАЈ	40
2. УПИТИ НАД ЈЕДНОМ ТАБЕЛОМ КОЈИМА СЕ ПРИКАЗУЈЕ ЊЕН ИЗМЕЊЕН САДРЖАЈ	51
3. УПИТИ НАД ДВЕ ИЛИ ВИШЕ ТАБЕЛА.....	61

1. Модел података

Апликативни софтвер је рачунарско¹ решење проблема из реалног света; он је (на рачунару извршив) модел разматраног система. За спецификацију и имплементацију тог модела потребни су одређени формални концепти и механизми. У знаменитом спису Едгара Ф. Кода (Edgar F. Codd) *Data Models in Database Management*² први пут је уведен појам **модела података** као формалног система који се користи за спецификацију и имплементацију модела разматраног система³. Циљ модела података је да омогући једнозначну и једноставну спецификацију и имплементацију:

- Ентитета из разматраног система (конструкција, ажурирање, деструкција),
- Њихових узајамних релација (повезивање, превезивање, развезивање) и
- Дозвољених трансформација (поступци и правила промене стања, начини извођења нових ентитета из постојећих).

Сваки модел података састоји се од⁴:

(1) Структурне компоненте – колекција концепата за дефинисање структуре система (стања система). Поред основних концепата, структурна компонента садржи и конструктор(е) нових, сложених концепата. Ова компонента модела података служи за опис статичких особина разматраног система, које су споро или ретко изменљиве, те зато релативно независне од времена (и простора). Концепти структурне компоненте пресликавају се у структуру будуће базе података.

Пример. У пекарству основне концепте представљају: брашно, квасац, со, вода, млеко, јаја, шећер, цем... Од њих се граде сложенији концепти као што су: тесто, надев, прелив...

Пример. У програмирању основне концепте представљају домен, тип, променљива (варијабла), вредност, скалар, вектор, матрица, листа...

(2) Операционе компоненте – колекција оператора (тј. операција) за манипулацију концептима структуре (промена стања, приказ стања). Ова компонента модела података служи за опис динамичких особина разматраног система, којима се исказује начин рада и промене стања разматраног система. Оператори ове компоненте модела података пресликавају се у операције над базом података. Наиме, сваки систем током

¹ Рачунар = хардвер + системски софтвер

² Први пут објављеном у *Proceedings of Workshop on Data Abstraction, Databases, and Conceptual Modelling* (Michael L. Brodie and Stephen N. Yilles, eds.), Pingree Park, Colo., June 1980. Касније више пута објављиван, са мањим изменама, у различитим часописима.

³ Извесну недоумицу може да изазове несрећно изабран израз **модел података**, јер он, у зависности од контекста у којем се нађе, представља бар два појма: први, то је **апстрактни систем** који обезбеђује фундаменталне концепте за моделовање било ког система, и други, то је модел којим је описан разматрани систем (тј. модел података разматраног система), а настао је употребом концепата дефинисаних у апстрактном систему; зато се овај модел може назвати **апстрактни модел** разматраног система. Ту није крај могућим недоумицама, јер се исти израз користи у оквиру софтвера (који је једини оперативни, тј. **конкретни модел**), где се прави разлика између **презентационог модела података**, **апликационог модела података** и **модела података базе**. Све у свему, да би се избегле недоумице, потребно је уочити три међусобно повезана, али посебна нивоа:

мета-мета-ниво	модел података као апстрактан систем
мета-ниво	модел података као апстрактан модел
конкретан-ниво	софтвер као конкретан модел
	презентациони модел података
	апликациони модел података
	модел података базе

⁴ Lazarević Branislav i dr., *Baze podataka*, FON, Beograd, 2003. Стр. 14-22, 39-44 и 88-93.

свог постојања подложен је променама стања. Те промене се у моделу (тј. у бази података) исказују путем измене података о концептима структурне компоненте. Да би база података представљала верну слику разматраног система, морају се дефинисати операције које мењају садржај базе података и доводе га у сагласност са стварним стањем разматраног система. Секвенце тих операција чине:

- (i) програме за ажирирање који мењају садржај базе података сагласно променама у разматраном систему и
- (ii) програме за извештавање који приказују садржај базе података, тј. тренутно стање система.

Програми представљају имплементацију процеса који се извршавају у разматраном систему.

Пример. У пекарству операције могу бити: направи_тесто, меси_тесто, посоли_тесто, одмери_тесто_за_векну_хлеба, обликуј_векну_хлеба, испеци_хлеб... Од наведених операција могу се направити процеси: направи_бели_хлеб, направи_црни_хлеб...

Пример. У програмирању постоје аритметички оператори (сабирање, одузимање, множење, дељење), логички оператори (конјункција, дисјункција, негација), скуповни оператори (припадност скупу), стринг оператори (конкатенација)... Они се могу користити за реализацију операција сабирања, одузимања, множења...

- (3) Интегритетне компоненте – служи за утврђивање валидних података у моделу података. Састоји се од две целине: колекције ограничења и колекције правила интегритета.

Колекцијом ограничења се искључују недозвољене вредности концепата структуре, односно обезбеђује појављивање само дозвољених вредности (конзистентност⁵ стања система). Деле се на:

- (i) ограничења модела података (ОМП) – општа ограничења која су својствена за сам модел података,
- (ii) ограничења разматраног система (тзв. пословна правила) – посебна ограничења која су својствена само за разматрани систем:
 - ограничења стања система (ограничења на могуће вредности података у бази) (ОСС),
 - ограничења промене стања, тј. прелаза из једног стања у друго стање система (ограничења на могуће промене вредности података у бази) (ОПС);

Следећи корак је дефинисање правилá интегритета која дефинишу основну динамику⁶ система. Свако правило интегритета чини тројка која се састоји од *операције* која се извршава (којим се мења стање система), *ограничења* које мора бити уважено (да би се очувала конзистентност стања) и *акције* која се предузима уколико је дошло до нарушања ограничења (а којом се систем преводи или враћа у конзистентно стање). Понекада се правилу интегритета додаје и четврти члан: *тренутак* испитивања ограничења, који може да буде: (1) *пре* извршења операције

⁵ Конзистентан = непротивречан, доследан, у складу са. У нашем контексту, то значи да се у бази података могу појавити само они подаци који су у складу са чињеничним стањем разматраног система.

⁶ Односно “понашање” система.

или (2) после извршења операције. Формално, правило интегритета је уређена четворка⁷:

ПИ = <Операција, Ограниччење, Акција [, Тренутак]>.

Ограниччења представљају основу за специфицирање правила интегритета, која се потом пресликају у услове интегритета⁸ будуће базе података.

Пример. У пекарству ограничења могу бити:

- О-1: Датум употребе квасца мора да буде пре датума истека његовог рока трајања. (OCC)
- О-2: За пшенични бели хлеб користи се оштро бело брашно, тип 500, добијено од тврде пшенице. (OCC)
- О-3: За пшенични црни хлеб користи се оштро црно брашно, тип 1100, добијено од тврде пшенице. (OCC)
- О-4: Векна белог хлеба типа „Сава“ тешка је 600 гр. (OCC)
- О-5: У све дуготрајне хлебове додаје се адитив AE-306 у следећем односу: на 10 кг теста додаје се 3 мл адитива. (OCC)
- О-6: Хлеб за отпис је сваки краткотрајни хлеб који је старији од три дана. (OPC)

Пример. У пекарству правила интегритета могу бити:

- ПИ-1: <направи_тесто, О-1, А-1, Пре>
- ПИ-2: <одмери_тесто_за_векну_хлеба, О-4, А-2, После>

Где су акције:

- А-1: Одби операцију и пријави надређеном разлог.
- А-2: Изврши исправку тежине теста.

Модел података је основа за развој софтвера, а посебно базе података, која је (врло често) најважнији део сваког софтвера. Формално-логички он је аксиоматски систем у којем се могу формулисати разне хипотезе и теорије о разматраном систему; информатичко-програмерски он је (семантички богат) језик за моделовање којим се могу конструисати рачунарски извршиви модели, тј. софтвери. Формално, модел података је уређена тројка:

МП = <Ст, Оп, Ин>

где су:

- Ст – структурна комонента,
- Оп – операциона компонента и
- Ин – интегритетна компонента.

2. База података. Систем за управљање базама података

База података (database, DB) је збирка узајамно повезаних података, који су трајно меморисани (похрањени) са контролисаном редундансом⁹, да би оптимално служила различитим апликацијама¹⁰. Подаци су меморисани одвојено и независно од

⁷ Средње заграде указују на опциону компоненту. Уколико је ова компонента изостављена, тренутак испитивања ограничења је после извршења операције.

⁸ Услов интегритета је механизам којим се у бази података обезбеђује да подаци о стању разматраног система буду усаглашени са: (1) ограничењима разматраног система (који су семантичког карактера) и (2) ограничењима самог модела података (који су техничког карактера).

⁹ Вишеструко понављање истих података.

¹⁰ Апликациони програм (краће: апликација) је рачунарски програм који се користи за одређену врсту послса и служи за непосредно задовољење коначних потреба крајњег корисника; то је програм који је написан да реши одређени проблем, произведе одређени извештај или ажурира одређене податке. На пример: (1) програм који креира платни списак је апликација која обрађује

апликационих програма који их користе. За додавање нових података и модификовање или претраживање постојећих података користе се заједнички или контролисани приступи. Са аспекта (физичке) организације података, база података је скуп интегрисаних датотека којима апликације приступају на контролисан начин, преко система за управљање базама података (СУБП). Са архитектуралног становишта, постоје три начина представљања једне базе података:

- екстерна репрезентација – састоји се из скупа апликационих подсхема¹¹ (нпр. погледи) или апликационог интерфејса¹² (нпр. усклађене процедуре и функције); садржи подмоделе података приложење корисницима базе¹³, а опционо и скуп операција које се могу извршити над тим подацима; скуп логичких подмодела базе података; ова репрезентација базе података је делимична¹⁴ и привремена¹⁵;
- концептуална репрезентација – логичка схема целокупне базе података, општа логичка структура базе података која приказује концептуални модел разматраног система; логички модел базе података; ова репрезентација базе података је целовита¹⁶ и привремена¹⁷;
- интерна репрезентација – физичка структура базе података (битови и бајтови, физички слогови и датотеке, индекси и кластери, блокови, стазе, плоче, цилиндри и дискови), физички модел базе података који приказује начин похањивања и приступа подацима на екстерној меморији; ова репрезентација базе података је целовита и трајна¹⁸ и једина стварно физички постојећа.

Систем за управљање базама података (database management system, DBMS) је програмски производ (софтвер) који је посредник између апликационих програма и крајњих корисника, са једне стране, и записа базе података на екстерној меморији, са друге стране. Обично се заснива на неком теоријском моделу података и у складу са њим¹⁹:

- 1) омогућава ефикасно креирање, коришћење и мењање базе података:
 - дефинисање структуре, операција и правила интегритета базе података,
 - селекцију и модификацију (упис, промена и брисање) садржаја базе података;
- 2) поседује механизме за:
 - управљање трансакцијама,
 - заштиту од неовлашћеног приступа подацима,
 - заштиту од уништења података,
 - обезбеђења ефикасног коришћења базе података,
 - управљање дистрибуираним деловима базе података.

финансијске податке о платама запослених; (2) програм за обраду текста (нпр. *MS Word*) је апликација за унос, обликовање и штампање текста. Насупрот апликацијама, оперативни систем, драјвер или компјултер могу бити од суштинског значаја за ефикасно коришћење рачунарског система, али ни један од наведених програма не учествује у непосредном задовољавању коначних потреба крајњих корисника.

¹¹ Опис података који су апликацији расположиви за коришћење.

¹² Интерфејс за програмирање апликација (Application Programming Interface, API); опис процедуралних и функцијских потпрограма који су апликацији расположиви за коришћење.

¹³ Апликациони програмери, аналитичари података, крајњи корисници.

¹⁴ Приказује само део базе података.

¹⁵ Траје само док траје апликациони програм који је користи.

¹⁶ Приказује целу базу података.

¹⁷ Траје само док се извршава СУБП (и који двосмерно пресликава физички модел БП у концептуални модел БП).

¹⁸ Постоји док постоји и меморијски медијум на којем су подаци записани у бинарном облику.

¹⁹ Могин Павле, Луковић Иван, *Принципи база података*, ФТН и МП “Stylos”, Нови Сад, 1996. Стр. 62-77.

Систем база података (database system, DBS) = Систем за управљање базама података + Базе података

3. Релациони модел података

У другој половини шездесетих година прошлог века појавили су се мрежни и хијерархијски системи за управљање базама података (Database Management System, DBMS). После почетних великих очекивања, већ почетком седамдесетих година, уочени су недостаци ових система за управљање базама података²⁰:

- недовољна раздвојеност логичког од физичког аспекта базе података,
- комплексне структуре података,
- коришћење процедуралних и навигационих језика.

Настојање да се реше ови недостаци имало је за последицу дефинисање релационог модела података²¹. Данас је релациони модел (РМ) најпопуларнији модел података и основа за имплементацију најзначајнијих комерцијалних СУБП. Његове битне особине су:

- једноставност - једина структура података у РМ је табела; операције над табелама су једноставне и дају за резултат нову табелу (хомогеност);
- декларативност – једноставна структура и једноставне операције добра су основа за дефинисање декларативног релационог језика²²; такође, највећи број ограничења је могуће реализовати декларативно;
- фундираност – целокупан РМ је формално-математички заснован: одређене врсте табела се могу третирати као математичке релације и искористити добро познати формализми за развој релационог језика, релационих база и на њима заснованог софтвера; тиме се успоставља општи теоријски и методолошки оквир који развој софтвера преводи из вештине у науку.

3.1. Концепција релационог модела података

3.1.1. Независност: логичка и физичка

3.1.2. Једноставност: табела као једина структура података

3.1.3. Декларативност: упитни језик

3.2. Структурална компонента релационог модела података

Концепти²³: Домен, атрибут, релација (релациона схема, релациона променљива и релациона вредност), кључ (кандидат за кључ, надкључ, примарни, алтернативни, спољни), база (схема релационе базе и инстанца релационе базе), NULL ознака (вредност)

²⁰ Могин Павле, наведено дело, стр. 8-9, 55-77.

²¹ За почетак теоријске формулатије релационог модела података сматра се спис Едгара Ф. Кода: *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, који је објављен 1970. године. Током скоро две следеће деценије настављен је интензиван теоријски рад на прецизном формулисању релационог модела.

²² Који се најчешће, погрешно, назива упитним језиком, јер је постављање упита само једна од могућности овог језика.

²³ Ово поглавље у целости је засновано на: [1] стр. 53-61 и [5] стр. 52-73, 79-105. Целокупна структура овог поглавља преузета је из [1], као и најважније дефиниције.

3.2.1. Релација

Скуп обједињава мноштво предмета у једну целину. Предмети морају да имају бар једну заједничку особину, које је основ њиховог обједињавања. Унутар скупа предмети су међусобно различити, што значи да је сваки члан скупа јединствен. Чланови скупа се називају елементима скупа; могу се навести набрајањем (екстензија скупа) или навођењем особине коју морају да задовоље (интензија скупа).

Пример. Скуп свих самогласника у српском језику може се описати на два начина:

1) екстензија скупа: $S = \{E, I, A, O, U\}$

2) интензија скупа: $S = \{x \mid x \in \mathcal{A} \wedge O(x)\}$, где је: \mathcal{A} српски алфабет, а $O(x)$ је особина: „ x је самогласник“

Пример. Скуп свих непарних природних бројева који су мањи од 10.

1) екстензија скупа: $A = \{1, 3, 5, 7, 9\}$

2) интензија скупа: $A = \{x \mid x \in \mathcal{N} \wedge x < 10\}$, где је \mathcal{N} скуп природних бројева

Декартов производ скупова A и B је скуп свих могућих уређених парова $\langle a, b \rangle$ тако да је $a \in A$ и $b \in B$:

$$A \times B = \{ \langle a, b \rangle \mid a \in A, b \in B \}.$$

Уопштено, нека су дати скупови D_1, D_2, \dots, D_n . Декартов производ ових скупова је скуп свих могућих уређених n -торки $\langle d_1, d_2, \dots, d_n \rangle$ тако да је $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$:

$$D_1 \times D_2 \times \dots \times D_n = \{ \langle d_1, d_2, \dots, d_n \rangle \mid d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n \}.$$

Пример. Декартов производ скупова $A = \{1, 3, 5, 7, 9\}$ и $B = \{2, 6, 10\}$ је скуп свих могућих уређених парова, таквих да је први елемент у пару из скупа A , а други елемент у пару је из скупа B . У овом случају, укупан број уређених парова је $5 \times 3 = 15$:

$$A \times B = \{ \langle 1, 2 \rangle, \langle 1, 6 \rangle, \langle 1, 10 \rangle, \langle 3, 2 \rangle, \langle 3, 6 \rangle, \langle 3, 10 \rangle, \langle 5, 2 \rangle, \langle 5, 6 \rangle, \langle 5, 10 \rangle, \langle 7, 2 \rangle, \langle 7, 6 \rangle, \langle 7, 10 \rangle, \langle 9, 2 \rangle, \langle 9, 6 \rangle, \langle 9, 10 \rangle \}.$$

Релација R дефинисана на скуповима A и B јесте подскуп Декартовог производа тих скупова, који садржи све оне уређене парове $\langle a, b \rangle$ који задовољавају услов релације Q :

$$A \times B \supseteq R = \{ \langle a, b \rangle \mid a \in A \wedge b \in B \wedge a Q b \}.$$

Пример. Дата су два скупа: $A = \{1, 3, 5, 7, 9\}$ и $B = \{2, 6, 10\}$. Дефинишими неколико релација над наведеним скуповима:

$$R_1: a = b/2, \text{ где: } a \in A, b \in B$$

$$R_2: a < b, \text{ где: } a \in A, b \in B$$

$$R_3: a = 3 * b, \text{ где: } a \in A, b \in B$$

Наведене релације важе за неке парове Декартовог производа скупова A и B , а за неке не важе. Релација је скуп свих они парови за које важи услов наведен у релацији. Претходно наведеним интензијама релације одговарају следеће екстензије релација:

$$R_1 = \{ \langle 1, 2 \rangle, \langle 3, 6 \rangle, \langle 5, 10 \rangle \}$$

$$R_2 = \{ \langle 1, 2 \rangle, \langle 1, 6 \rangle, \langle 1, 10 \rangle, \langle 3, 6 \rangle, \langle 3, 10 \rangle, \langle 5, 6 \rangle, \langle 5, 10 \rangle, \langle 7, 10 \rangle, \langle 9, 10 \rangle \}$$

$$R_3 = \{ \emptyset \}$$

Дакле, релација је скуп парова (ако се ради о бинарној релацији) за које је однос који одређује релација задовољен. Важи $R \subseteq A \times B$ и то онај подскуп који задовољава услов задате релације.

Уопшено, **релација** R дефинисана на скуповима D_1, D_2, \dots, D_n јесте подскуп Декартовог производа тих скупова, који садржи све оне н-торке $\langle d_1, d_2, \dots, d_n \rangle$ које задовољавају услов релације Q :

$$D_1 \times D_2 \times \dots \times D_n \supseteq R = \{ \langle d_1, d_2, \dots, d_n \rangle \mid d_1 \in D_1 \wedge d_2 \in D_2 \wedge \dots \wedge d_n \in D_n \wedge Q(d_1, d_2, \dots, d_n) \}.$$

Мање формално, релација R дефинисана на скуповима D_1, D_2, \dots, D_n јесте подскуп Декартовог производа наведених скупова: $R \subseteq D_1 \times D_2 \times \dots \times D_n$. Подскуп R садржи оне н-торке Декартовог производа које задовољавају задату релацију.

Домен релације. Ако је релација $R \subseteq D_1 \times D_2 \times \dots \times D_n$, онда се скупови D_1, D_2, \dots, D_n називају доменима релације R .

Степен релације. Ако је релација $R \subseteq D_1 \times D_2 \times \dots \times D_n$, онда је степен релације R једнак n . Дакле, број домена на којима је дефинисана нека релација јесте степен релације. Разликујемо унарне (на једном домену), бинарне (на два домена) и н-арне релације.

Кардиналност релације је број н-торки у релацији.

Атрибут(и) релације. Дати су скупови A, B, C и релација R :

$$A = \{1622, 1611, 1522\}$$

$$B = \{\text{Јова, Ана}\}$$

$$C = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$A \times B \times C \supseteq R = \{ \langle 1611, \text{Јова}, 3 \rangle, \langle 1622, \text{Ана}, 5 \rangle, \langle 1522, \text{Ана}, 7 \rangle \}$$

Како је релација скуп уређених н-торки, редослед елемената у једној н-торки је битан. Међутим, ако вредностима елемената у н-торкама придржимо имена домена, редослед елемената у н-торкама више неће имати значаја. Доделићемо скуповима и релацији семантичка имена:

$$A = \text{БИ} \quad \text{број индекса}$$

$$B = \text{Име} \quad \text{име студента}$$

$$C = \text{Сем} \quad \text{уписани семестар}$$

$$R = \text{СТУДЕНТ} \quad \text{студент факултета}$$

Сада је могуће претходно уведену релацију записати на следећи начин:

$$\begin{aligned} \text{СТУДЕНТ} = \{ & \langle \text{БИ:1611, Име:Јова, Сем:3} \rangle, \langle \text{БИ:1622, Сем:5, Име:Ана} \rangle, \\ & \langle \text{Сем:7, Име:Ана, БИ:1522} \rangle \} \end{aligned}$$

Атрибут релације се формално може дефинисати као пар **<назив домена, вредност домена>**. На пример, **<Име, Јова>** или **<БИ, 1611>**. Концепт атрибута омогућава представљање релације као табеле. Релација СТУДЕНТ се може представити у табеларној форми:

СТУДЕНТ			⇐ Назив релације	Схема релације (интензија релације)
БИ	Име	Сем	⇐ Атрибути релације	
1611	Јова	3	⇐ н-торка #1	Вредност релације или Р-вредност
1622	Ана	5	⇐ н-торка #2	(екstenзија релације)
1522	Ана	7	⇐ н-торка #3	

Екстензија релације је скуп свих н-торки разматране релације, односно табеларни приказ свих њених врста. Интензија релације је генерализација екстензије, она је временски непроменљива и састоји се од назива релације и атрибута релације.

Пошто је релација скуп, а свака табела није скуп, неопходно је одредити услове које табела мора да задовољи да би била релација:

- 1) Све врсте у табели морају да буду различите.
- 2) Редослед врста у табели је произвољан.
- 3) Редослед колона у табели је произвољан.
- 4) Све вредности поља (тј. вредности атрибута) у табели су атомске, односно није дозвољено да вредности неких поља у табели буду табеле²⁴.

Ако табела задовољава наведене услове, онда је она релација у првој нормалној форми (1НФ).

3.2.2. Домени атрибута

Сада се домени могу дефинисати као скупови из којих атрибути релације узимају своје вредности. Уобичајено је да се домени поделе на:

- Предефинисане домене – домене који су већ имплементирани и постоје у релационим језицима база података. На пример, у Transact-SQL-у постоје: цели бројеви (int, smallint, bigint, tinyint, bit), низови знакова²⁵ (char, varchar, text; nchar, nvarchar, ntext), децимални бројеви (decimal или numeric; money, smallmoney; float, real), датуми (datetime, date, time, datetime2, datetimeoffset, smalldatetime), низови бајтова²⁶ (binary, varbinary, image), просторни типови (spatial types) и посебни типови (cursor, timestamp, hierarchyid, uniqueidentifier, sql_variant, xml, table). Предефинисани домени у релационим језицима су еквивалентни предефинисаним (уграђеним) типовима података у програмским језицима.
- Семантичке домене – домене које програмер (database developer) креира употребом предефинисаних или претходно креираних семантичких домена. Дефиниција семантичког домена састоји се из навођења основног (“носећег”, underlaying) типа података и ограничења; на тај начин се новокреираном домену додељује одређено значење²⁷, због чега се и називају семантички. Семантички домени у релационим језицима су еквивалентни апстрактним типовима података²⁸ у програмским језицима.

Атрибути релације би увек требало дефинисати над семантичким доменима, јер само тако програмер/СУБП може имати пуну контролу у извршавању операција над базом података. Два атрибута у моделу су семантички еквивалентна само ако су дефинисани над истим доменом.

3.2.3. Кључеви релације

Кључ релације је непразан скуп атрибута чије вредности јединствено одређују једну н-торку. Кључ може бити прост, уколико се састоји из једног атрибута, или сложен, уколико се састоји из два или више атрибута. На пример, у релацији СТУДЕНТ атрибут БИ је прост кључ (не постоје два студента која имају исти број индекса, тј. не постоје две н-торке релације СТУДЕНТ са истом вредношћу атрибута БИ). Формално, кључ релације Р је таква колекција К њених атрибута која задовољава следећа два условия:

- 1) **особина јединствености:** не постоје било које две н-торке са истом вредношћу атрибута из К.
- 2) **особина нередундантности:** ако се изостави било који атрибут из К, губи се особина јединствености.

²⁴ Нису дозвољене табеле у табели.

²⁵ Низ знакова = ниска, стринг (string)

²⁶ Бинарне ниске

²⁷ “Уграђује” се одређено знање о разматраном систему у домен.

²⁸ Које, такође, креира програмер (application programmer).

Она колекција атрибута која задовољава само особину јединствености назива се **надкључ** релације. На пример, колекција атрибута (БИ, Име) јесте надкључ релације СТУДЕНТ, јер задовољава само услов јединствености. У једној релацији може постојати више различитих колекција атрибута које задовољавају дефиницију кључа. На пример, ако у релацију СТУДЕНТ додамо атрибут МБ (матични број), онда ће и он задовољавати дефиницију кључа. Све такве колекције се називају **кандидати за кључ**. Један од кандидата који се изабере да служи за идентификацију н-торки релације назива се **примарни кључ**. Остали неизабрани кандидати се тада називају **алтернативним кључевима**. Примарни кључ се у релационој схеми подвлачи. На пример, релациона схема за релације СТУДЕНТ и СМЕР:

СТУДЕНТ	(<u>БИ</u> , Име, Сем, Смер#)
СМЕР	(<u>С#</u> , Назив)

Атрибути релације који су кључеви или су делови сложених кључева (било примарних, било алтернативних) називају се **кључни атрибути**. Остали атрибути релације су **некључни атрибути**. На пример, у претходним релационим схемама некључни атрибути су: Име, Сем и Назив. А какав је атрибут Смер#?

Свака релација може садржати атрибут(е) који је(су) у некој другој релацији примарни кључ. Такав атрибут(и) се назива **спољни кључ**²⁹. Вредност спољног кључа користи се за повезивање са вредношћу примарног кључа неке друге релације. У релационом моделу, везе између релација успостављају се преко вредности спољних кључева³⁰. У горе наведеном примеру, атрибут Смер# је спољни кључ; преко њега се релација СТУДЕНТ повезује са релацијом СМЕР. Спољни кључ СТУДЕНТ.Смер# референцира примарни кључ СМЕР.С#. Формално, спољни кључ се може одредити на следећи начин: Нека је релација R1 базна релација (*референцирајућа релација*). Спољни кључ SK у релацији R1 јесте подскуп њених атрибута такав да задовољава следећа два услова:

- 1) *егзистенција референциране релације*: постоји базна релација R2 која садржи кандидат за кључ KK и
- 2) *егзистенција референцирајуће вредности*: свака вредност SK у релацији R1 једнака је некој вредности KK у релацији R2 или је NULL³¹ вредност.

На пример, дате су следеће релације:

СТУДЕНТ				СМЕР	
БИ	Име	Сем	Смер#	C#	Назив
1611	Јова	3	10	10	Софтверско инжењерство
1622	Ана	5	NULL	20	Информациони системи
1522	Ана	7	20	30	Информационе технологије
1533	Ема	7	10		

Егзистенција референцирајуће релације:

Референцирајућа релација (R1): СТУДЕНТ

Спољни кључ (R1.SK): СТУДЕНТ.Смер#

Егзистенција референциране релације:

Референцирана релација (R2): СМЕР

Кандидат за кључ (R2.KK): СМЕР.С#

Егзистенција референцирајуће вредности:

{ СТУДЕНТ.Смер# } \subseteq { СМЕР.С# } ³²

²⁹ Или: страни кључ.

³⁰ Зато се релациони модел назива вредносно-оријентисани модел података.

³¹ Видети поглавље 3.2.5.

³² Прецизније: { СТУДЕНТ.Смер# } \subseteq { СМЕР.С# } \cup { NULL }

3.2.4. Схема релационе базе и релациона база

Схема релационе базе (relational database scheme, RDS) дефинише логичку структуру базе података као скупа интензија релација. На пример, за концептуални модел наведен у прилогу, схема релационе базе података је:

ОДСЕК	(<u>О#</u> , Назив)
СМЕР	(<u>С#</u> , Назив, Одсек#)
СТУДЕНТ	(<u>БИ</u> , Име, Презиме, СредњеИме, ДатумРођења, Слика, ГодинаСтудија, ВозачкаКатегорија, Кредит, Стипендија, Смер#)
ПРЕДМЕТ	(<u>П#</u> , Назив, Врста, ЕСПБ, Одсек#)
ИСПИТ	(<u>БИ</u> , <u>П#</u> , ДатумПолагања, Оцена)

Релациона база података (relational database, RDB) је:

- физички: скуп интегрисаних датотека са минималном редундансом³³ које су међусобно повезане³⁴ и променљиве у времену;
- концептуално: скуп временски променљивих релација³⁵;
- апликативно: скуп табела у којима се трајно чувају ажурабилни подаци о ентитетима из разматраног система.

На пример, за концептуални модел наведен у прилогу, део релационе базе података је:

ПРЕДМЕТ				СМЕР	
П#	Назив	Врста	ЕСПБ	C#	Назив
1	Програмирање I	ОБВ	6	10	Софтверско инжењерство
2	Математика I	ОБВ	6	20	Информациони системи
3	Базе података	ОБВ	6	30	Информационе технологије
4	Економија	ОБВ	8	40	Управљање производњом
5	Увод у ИС	ОБВ	6	50	Управљање пословањем
6	Математика II	ОБВ	6	60	Управљање кадровима
7	Основе организације	ОБВ	6		
8	Социологија	ИЗБ	5		
9	Психологија	ИЗБ	5		

Релације у некој бази података могу да се поделе на базне и изведене. **Базна релација** је релација која се не може извести из осталих релација у релационој бази података, а у бази података се чува њена интензија и екстензија. **Изведена релација** (поглед, приказ) је релација која се може извести из скupa базних или изведенних релација, преко операција које се дефинишу над релацијама; у бази података се чува само њена интензија и поступак извођења, али не и подаци.

Упоредни приказ израза (термина) релационе, табеларне и датотечне обраде података:

РЕЛАЦИОНА	ТАБЕЛАРНА	ДАТОТЕЧНА
Домен	Скуп вредности	Тип
Атрибут	Колона	Поље
Н-торка	Врста	Запис / Слог / Рекорд
Примарни кључ	Идентификатор (врсте)	Јединствени кључ
Релација	Табела	Датотека података
Индекс	Индекс	Индексна датотека
Степен	Број колона	Број поља у запису
Кардиналност	Број врста	Број записа у датотеци

³³ Постоји редунданса спољних и примарних кључева.

³⁴ Преко примарних и спољних кључева.

³⁵ Појам релације се у овој дефиницији користи двојако: као схема релације (интензија; апстрактна структура података) и као вредност релације (екстензија; физички постојећи, конкретни подаци). Зато се израз “временски променљивих” користи да указаје на чињенице да се током времена: (1) схема релације може мењати и (2) вредности атрибута релације могу мењати.

3.2.5. NULL вредност

У случају недостатка податка о вредности атрибута, у базу података се уместо недостајуће вредности уписује NULL вредност. На пример, у табели СТУДЕНТ за н-торку:

< БИ: 1622, Име: Ана, Сем: 5, Смер#: NULL >

вредност NULL у атрибуту Смер# значи да је **тренутно непозната вредност** за шифру смера. Другим речима, када се сазна вредност атрибута за разматрану н-торку (вредност атрибута Смер# за студента чији је БИ = 1622) та ће вредност заменити NULL вредност, тј. биће уписана у базу података.

Прецизније, NULL вредност и није права вредност, већ је треба схватити као ознаку специјалне намене, маркер који указује на тренутно непознату вредност, чувар места (placeholder) за будућу вредност атрибута. У бази података ознака NULL има специфичан бинарни код, различит од свих других бинарних кодова. Зато се ознака NULL може користити уместо било које тренутно непознате вредности, за атрибуте било ког типа.

Проблеми са ознаком NULL могу настати у случају када је неки атрибут **неприменљиво својство** за сва појављивања објекта представљених н-торкама разматране релације. На пример, нека је релацији СТУДЕНТ додат атрибут ВозКат који значи возачку категорију коју студент има:

СТУДЕНТ			
БИ	Име	ВозКат	Смер#
1611	Јова	NULL	10
1622	Ана	Б	NULL
1522	Ана	NULL	20
1533	Ема	А	10

Међутим, атрибут ВозКат применљив је само на студенте који имају возачку категорију. Студенти који имају БИ = 1611 и БИ = 1522 немају возачку категорију, па је зато:

- 1) атрибут ВозКат за њих неприменљиво својство,
- 2) вредност атрибута ВозКат за оба студента NULL.

Овде настаје проблем, јер у овом случају није могуће разлучити да ли је ознака NULL *тренутно непозната вредност* или *неприменљиво својство*. Означавање две семантички различите ситуације са једном ознаком NULL доводи до забуне. Да би се забуна избегла ознака NULL се користи само у првом случају (*тренутно непозната вредност*).

Генерално решење за атрибуте који су *неприменљива својства* је да се релације пројектују тако да су сви атрибути применљива својства за све н-торке у релацији. У претходном примеру то се може постићи декомпозицијом табеле СТУДЕНТ на две нове релације:

СТУДЕНТ_1 (БИ, Име, Смер#) и
СТУДЕНТ_ВОЗАЧ (БИ, ВозКат).

3.3. Операциона компонента релационог модела података

Операције релационог модела могуће је исказати коришћењем релационе алгебре. Операције над релацијама могу се разврстати у:

- Конвенционалне скуповне операције: унија, пресек, разлика и Декартов производ;
- Специјалне релационе операције: селекција, пројекција, спајање и делење;
- Додатне релационе операције: полуспајање, хоризонтално скаларно рачунање у релационој алгебри (*extend*), вертикално скаларно рачунање у релационој алгебри

- (*summarize*), операције поређења релација (једнакост и неједнакост релација, подскуп и надскуп релације, прави подскуп и прави надскуп релације);
- Операције ажурирања релације: *insert* (додавање нове н-торке у релацију), *delete* (брисање н-торке из релације) и *update* (промена вредности атрибута);
 - Операције са NULL ознакама.

Пошто су операције релационог модела интуитивно јасне, овде се неће приступити њиховом формалном и детаљном објашњавању³⁶.

3.4. Интегритетна компонента релационог модела података

3.4.1. Ограничења

Ограничења релационог модела података. Сама структура релационог модела диктира два општа ограничења:

- (i) Ограничење на вредности примарног кључа,
- (ii) Ограначење на вредности спољног кључа.

Ограничења разматраног система (пословна правила). У релационом моделу *посебна ограничења* је могуће поделити на:

- (i) ограничења стања система (ограничења на могуће вредности података у бази), а која могу бити:

O1: ограничења за домене – којима се одређује које вредности постоје у домену (дозвољене вредности за домен); на пример, домен ОЦЕНА садржи вредности од 5 до 10;

O2: ограничења за атрибуте – којима се одређују дозвољене вредности неког атрибута независно од вредности других атрибута у бази (дозвољене вредности за атрибут); на пример, атрибут Оцена релације ПОЛОЖЕНИ_ИСПИТ може да садржи вредности од 6 до 10 (мада је сам атрибут Оцена дефинисан над доменом ОЦЕНА):

ПОЛОЖЕНИ_ИСПИТ. Оцена **is type** ОЦЕНА **and check value in [6..10];**

O3: ограничења за релације – којима се одређује вредност једног атрибута релације на основу вредности другог атрибута (других атрибута) у једној релацији (међузависност вредности атрибута једне релације); на пример, ако је ставка рачуна представљена следећом релационом схемом:

СТАВКА(Рачун#, С#, Производ, Количина, ЈединичнаЦена, Вредност)

онда се последњи атрибут у претходној релационој схеми израчунава по обрасцу:

СТАВКА.Вредност = СТАВКА.Количина * СТАВКА.ЈединичнаЦена;

O4: ограничења за базу – којима се одређује вредност једног атрибута релације на основу вредности атрибута из више релацији (међузависност вредности атрибута више релација); на пример, ако је дата релациониа схема:

РАЧУН(Р#, ..., Укупно)
СТАВКА(Рачун#, С#, Производ, Количина, ЈединичнаЦена, Вредност)

онда се вредност атрибута Укупно израчунава на следећи начин:

РАЧУН.Укупно = **SUM(СТАВКА.Вредност WHERE РАЧУН.Р# = СТАВКА.Рачун#)**

³⁶ Такви описи могу се наћи у: [1] стр. 62-83 и [5] стр. 139-218.

(ii) ограничења промене стања, тј. прелаза из једног стања у друго стање система (ограничења на могуће промене вредности података у бази) – којима се одређује да ли је могућ прелаз из једног скупа вредности атрибута у други скуп вредности; на пример, нека су:

- дате следеће релације:

СТУДЕНТ				СМЕР	
БИ	Име	Сем	Смер#	C#	Назив
1611	Јова	3	10	10	Софтверско инжењерство
1622	Ана	5	NULL	20	Информациони системи
1522	Ана	7	30	60	Управљање кадровима
1533	Ема	7	10		

- задато следеће правило: Студенти са смера ‘Управљање кадровима’ не могу да пређу на смер ‘Информациони системи’, а ни на смер ‘Софтверско инжењерство’. Обрнуто је дозвољено.

Онда би спецификација претходног правила могла да се обави на следећи начин³⁷:

```

CONSTRAINT [ПрелазСаСмераНаСмер]
    ON [СТУДЕНТ]
BEGIN
    LET &УК is type as SCALAR( [СМЕР.C#] );
    LET &СИИС is type as TABLE( [СМЕР.C#] );

    &УК := SELECT [C#] FROM [СМЕР]
        WHERE [НАЗИВ] LIKE ‘Управљање кадровима’;
    &СИИС := SELECT [C#] FROM [СМЕР]
        WHERE [НАЗИВ] LIKE ‘Софтверско инжењерство’
        OR [НАЗИВ] LIKE ‘Информациони системи’;

    IF ( OLD[СТУДЕНТ.Смер#] = &УК ) AND ( NEW[СТУДЕНТ.Смер#] IN &СИИС )
        THEN Rejection;
END.

```

На основу претходно дефинисаних ограничења могу се дефинисати правила интегритета.

3.4.2. Правила интегритета

Присетимо се да је правило интегритета³⁸ уређена четворка³⁹:

ПИ = <Операција, Ограниччење, Акција [, Тренутак]>.

Правила интегритета дефинишу дозвољена стања и дозвољене прелазе система из једног у друго стање. У релационом моделу правила интегритета се исказују дефинисањем ограничења на вредности атрибута и акцијама које се предузимају када нека операција ажурирања наруши раније утврђено ограничење.

Правила интегритета релационог модела података:

1. интегритет ентитета (интегритет примарног кључа),
2. референцијални интегритет.

³⁷ У хипотетичком језику.

³⁸ Ово поглавље у целости је засновано на: [1] стр. 87-94 и [5] стр. 110-134. Целокупна структура овог поглавља преузета је из [1], као и најважније дефиниције.

³⁹ Средње заграде указују на опциону компоненту.

Правила интегритета разматраног система (пословна правила):

1. правила интегритета стања система,
 - (i) интегритет домена,
 - (ii) интегритет атрибута,
 - (iii) интегритет релације,
 - (iv) интегритет базе;
2. правила интегритета промене стања, тј. прелаза из једног стања у друго стање система.

Интегритет ентитета (интегритет примарног кључа): ниједан атрибут који је примарни кључ или део примарног кључа неке базне релације не може да има NULL вредност.

Примарни кључеви у базним релацијама идентификују један објекат у скупу објеката (нпр. један предмет у релацији ПРЕДМЕТ или једног студента у релацији СТУДЕНТ). Због такве њихове улоге примарни кључеви не могу имати NULL вредности. Постојање NULL вредности у примарном кључу (целом или било ком његовом делу) нарушило би саму дефиницију кључа и то како особину јединствености, тако и особину нередундантности.

Ограничење које дефинише интегритет ентитета могу да наруше операције убацивања нове н-торке у релацију или операције промене вредности. У оба случаја једино могућа акција је одбијање (*reject* = одбацивати, одбијати; *rejection* = одбацивање, одбијање) операције.

Референцијални интегритет: Ако нека базна релација R1 поседује спољни кључ R1.SK који ову релацију повезује са неком другом базном релацијом R2 преко примарног кључа R2.PK, тада свака вредност R1.SK мора бити било једнака некој вредности R2.PK или бити NULL вредност. Релације R1 и R2 не морају бити различите. Коришћењем релационе алгебре претходно ограничење референцијалног интегритета може се записати на следећи начин:

$$\pi_{\text{SK}}(R1) \subseteq \pi_{\text{PK}}(R2) \cup \{ \text{NULL} \}^{40}$$

Референцијални интегритет обезбеђује коректно повезивање н-торки релација у релационом моделу⁴¹. Нарушавањем референцијалног интегритета укидају се везе између н-торки релација и читава релационија база података се распада. Да до тога не би дошло, дефинише се правило референцијалног интегритета. Нарушавање овог правила може се десити у два случаја:

1. У релацији која садржи спољни кључ⁴² вредности атрибута које чине спољни кључ нису одговарајуће. То се може десити приликом:
 - а. Убацивања нове н-торке у релацију,
 - б. Измене вредности атрибута који чине спољни кључ.
2. У релацији која садржи примарни кључ⁴³ вредности атрибута које чине примарни кључ нису одговарајуће. То се може десити приликом:
 - а. Избацивања н-торке из релације,
 - б. Измене вредности атрибута који чине примарни кључ.

За случајеве 1.а. и 1.б. једино могућа акција је одбијање операције (*rejection*) која је изазвала нарушавање ограничења. У случајевима 2.а. и 2.б. уколико дође до нарушавања референцијалног интегритета могуће су следеће акције:

⁴⁰ Пројекција релације R1 по спољном кључу SK увек је подскуп пројекције релације R2 по примарном кључу PK проширене елементом NULL.

⁴¹ Видети 3.2.3. за пример.

⁴² Референцирајућа релација.

⁴³ Референцирана релација.

- *Rejection (Restrict)*: одбија се операција која је нарушила ограничење референцијалног интегритета;
- *Cascade*: преноси се извршавање одговарајуће операције⁴⁴ и на релацију у којој се налази спољни кључ, да би се на тај начин задовољило ограничење референцијалног интегритета. Другим речим, покреће се⁴⁵ извршење истоветне операције и на референцирајућој релацији (“продужено” избацивање или “продужена” измена).
- *SetNull*: замењује се вредност спољног кључа са NULL вредношћу за све н-торке којима одговара избачени или промењени спољни кључ.
- *SetDefault*: замењује се вредност спољног кључа са неком унапред одређеном (default) вредношћу за све н-торке којима одговара избачени или промењени спољни кључ.
- *NoAction*: обавља се захтевано ажурурање базе података без икаквих додатних акција⁴⁶.

Правила интегритета разматраног система (пословна правила) изражавају посебна ограничења и одговарајуће акције при њиховом нарушавању. Ове врсте акција се не могу схематизовати и унапред одредити као што је то урађено за референцијални интегритет. Зато свако нарушавање ограничења које је својствено разматраном систему захтева специфичну акцију, која се имплементира преко тригера⁴⁷.

Интегритет домена одређује скуп дозвољених вредности за домен и акцију која се предузима уколико разматрана вредност није у том скупу. Ограниченије за домен могу да наруше операције убацивања нове н-торке у релацију или операције промене вредности атрибута постојеће н-торке. У оба случаја једино могућа акција је одбијање операције.

ПИ_{домен} = <Домен, [Операција,] ⁴⁸ Ограниченије, [Rejection,] Пре>,
где је: Операција = { Update, Insert }.

Дефинисање правила интегритета за домен јесте друго име за креирање семантичког домена.

Интегритет атрибута одређује скуп дозвољених вредности за атрибут и акцију која се предузима уколико разматрана вредност није у том скупу. Ограниченије за атрибут могу да наруше операције убацивања нове н-торке у релацију или операције промене вредности атрибута постојеће н-торке. У оба случаја једино могућа акција је одбијање (*rejection*) операције.

ПИ_{атрибут} = <Атрибут, Домен, [Операција,] ⁴⁹ Ограниченије, [Rejection,] Пре>,
где је: Операција = { Update, Insert }.

Интегритет релације одређује:

- ограничење којим се исказује на који начин је вредност једног атрибута у релацији повезана (условљена) са вредностима других атрибута у истој релацији и
- акцију која се предузима уколико је та повезаност (условљеност) нарушена.

⁴⁴ Избацивање за избацивање и промена вредности за промену вредности.

⁴⁵ “окида се”

⁴⁶ Опрез! Ово је опасно и прихватљиво је само у ретким случајевима!

⁴⁷ Тригер (trigger) је процедуре која се извршава кад год је задовољен услов за њено активирање. Он се не може извршити као што се извршавају остале процедуре, непосредним позивом, већ само када се деси неки догађај или нека операција ажурирања базе (Update, Insert, Delete). Претходно дефинисане акције *Rejection (Restrict)*, *Cascade*, *SetNull* и *SetDefault* могу се схватити као специјализовани тригери.

⁴⁸ Навођење операције и акције је у начелу непотребно, јер је акција увек иста: било да је операција *Update* или *Insert*, уколико је ограничење нарушено акција је увек *Rejection*.

⁴⁹ Види претходну напомену.

Ограниччење повезаности вредности атрибута у једној релацији могу да наруше операције: убацивања нове н-торке, брисање постојеће н-торке или промене вредности атрибута постојеће н-торке. У свим случајевима могућа је једна од следећих акција:

- одбијања (*rejection*) извршења операције која је нарушила интегритет⁵⁰ или
- израчунавање нове вредности зависног атрибута и ажурирање зависног атрибута новосрачунатом вредношћу⁵¹.

$\text{ПИ}_{\text{релације}} = <\text{Релација}, \text{Атрибут}, \text{Операција}, \text{Ограниччење}, \text{Акција}, \text{Тренутак}>$,
где је: Операција = { *Update, Insert, Delete* }.

У *Ограниччењу* и *Акцији* дозвољен је приступ и промена података само у *Релацији* специфицираној у $\text{ПИ}_{\text{релације}}$.

Интегритет базе одређује било које сложено ограничење на вредности атрибута у бази података, ограничење које повезује вредности атрибута из више релација. Прецизније, интегритет базе одређује:

- ограничење којим се исказује на који начин је вредност једног атрибута у релацији повезана (условљена) са вредностима других атрибута у осталим релацијама и
- акцију која се предузима уколико је та повезаност (условљеност) нарушена.

Ограниччење повезаности вредности атрибута у бази могу да наруше операције: убацивања нове н-торке, брисање постојеће н-торке или промене вредности атрибута постојеће н-торке. У свим случајевима могућа је једна од следећих акција:

- одбијања (*rejection*) извршења операције која је нарушила интегритет⁵² или
- израчунавање нове вредности зависног атрибута и ажурирање зависног атрибута новосрачунатом вредношћу⁵³.

$\text{ПИ}_{\text{базе}} = <\text{Релација}, \text{Атрибут}, \text{Операција}, \text{Ограниччење}, \text{Акција}, \text{Тренутак}>$,
где је: Операција = { *Update, Insert, Delete* }.

У *Ограниччењу* и *Акцији* дозвољен је приступ подацима у свим релацијама базе података, али је дозвољена промена само *Атрибута* који припада *Релацији* специфицираној у $\text{ПИ}_{\text{базе}}$.

Интегритет прелаза из стања у стање одређује услове прелаза из једног (старог, почетног) у друго (ново, одредишно) стање. За то су потребне две помоћне привремене релације: једна која садржи измене н-торке са старим вредностима (*OLD* или *DELETED*) и друга која садржи исте те измене н-торке, али са новим вредностима (*NEW* или *INSERTED*). Поређењем вредности атрибута у старим н-торкама са вредностима атрибута у новим н-торкама (тј. старог стања са новим стањем), утврђује се да ли је ограничење за дозвољени прелаз из једног стања у друго стање нарушено. Ограниччење могу да наруше операције ажурирања (*Insert, Update* и *Delete*). Ако је ограничење нарушено, приступа се акцији која базу преводи у неко ново или враћа у старо конзистентно стање.

$\text{ПИ}_{\text{прелаза}} = <\text{Релација}, \text{Атрибут}, \text{Операција}, \text{Ограниччење}, \text{Акција}, \text{Тренутак}>$

У *Ограниччењу* и *Акцији* дозвољен је приступ и промена података у свим релацијама базе података, а не само у *Релацији* специфицираној у $\text{ПИ}_{\text{прелаза}}$.

⁵⁰ Лошији приступ.

⁵¹ Бољи приступ.

⁵² Лошији приступ.

⁵³ Бољи приступ.

3.5. Кодова релациона правила

Кодова правила⁵⁴ чини скуп од тринест правила која је предложио Едгар Ф. Код, са циљем да дефинише шта је потребно да поседује један систем за управљање базом података да би га могли сматрати релационим (Relational DBMS, RDBMS). Ова правила су срочена са намером да се спречи комерцијално “разводњавање” дефиниције релационог модела података и укаже на неодоварајуће имплементације система за управљање базама података⁵⁵. Детаљно бављење овом проблематиком довело је Е. Ф. Кода до књиге *The Relational Model for Database Management: Version 2*, у којој је изложио 333 релационих правила. Треба рећи да већина комерцијалних система никада није у потпуности задовољила ни ових овде наведених 1+12 правила.

0. Основно правило:

Да би систем назвали релационим системом за управљање базама података (РСУБП), он у свом функционисању мора користити искључиво сопствене *релационе* могућности за управљање подацима.

Следи дванаест правила⁵⁶ који су разложено и потанко објашњено *Основно правило*:

1. Правило представљања података:

Све подаци у релационој бази података су на логичком нивоу представљени само на један начин – вредностима у релацијама (табелама).⁵⁷

2. Правило обезбеђеног начина приступа подацима:

Сваки појединачни податак (атомска вредност) у релационој бази доступан је навођењем назива релације (табеле), вредности примарног кључа и назива колоне.⁵⁸

3. Систематично третирање NULL вредности:

У потпуно релационом СУБП-у за представљање податка који недостаје или који се не може применити (недостајућа вредности или неприменљиво својство) користи се NULL вредност⁵⁹. Систем подржава све операције са NULL вредностима, без обзира на тип података.

4. Динамички, увек доступан каталог (речник) података заснован на релационом моделу:

Опис базе података (мета-подаци) на логичком нивоу представљен је на исти начин као и обични подаци⁶⁰, тако да овлашћени корисник може да користи исти релациони упитни језик за приступ и мета-подацима и обичним подацима.

5. Правило свеобухватног подјезика за податке:

Релациони систем може да подржава више језика и више различитих начина крајње употребе података. Међутим, мора да постоји барем један језик чије се наредбе могу изразити као низ карактера са добро дефинисаном синтаксом и који подржава:

- Дефиницију података
- Дефиницију погледа
- Манипулацију подацима (интерактивно и кроз апликативне програме)

⁵⁴ Видети у литератури [6] и [7]. Као и линк [4].

⁵⁵ Узети у обзор напомену Дејва Вурхиса (Dave Voorhis): “Note that [...] these rules were appropriate to their time (mid 1980s), their intent (quash marketing hype at a time when any remotely DBMS-like product would probably claim to be “relational”), and their audience (readers of ComputerWorld, a popular semi-technical industry magazine) [...].”

⁵⁶ За која Е. Ф. Код каже: “All 12 rules are motivated by Rule Zero defined above, but a DBMS can be more readily checked for compliance with these 12 than with Rule Zero”.

⁵⁷ Структура базе података се на концептуалном (овде: логичком) нивоу представља искључиво релацијама (табелама).

⁵⁸ Без унапред задатих приступних путева и без рекурзије или итерације.

⁵⁹ Специфична ознака, различита од свих других вредности у бази; различита од размака, празног низа знакова, нуле или било ког броја.

⁶⁰ То јест, у облику релација (табела).

- Дефиницију правила интегритета
- Ауторизацију (сигурност података, права коришћења података)
- Дефинисања граница трансакција (begin, commit, rollback).

6. Правило ажурирања погледа:

Сви погледи који су теоријски ажурабилни, ажурабилни су и од стране система.⁶¹

7. Уношење, мењање и брисање (уклањање) података на нивоу скупова:

И над базним релацијама и над изведеним релацијама (погледима) могу се извршавати не само операције извештавања (приказа података), већ и операције ажурирања података (insertion, update and deletion of data).

8. Физичка независност података:

Апликативни програми и интерактивна комуникација остају логички непромењени кад год се промени физичка организација базе или метод приступа.⁶²

9. Логичка независност података:

Апликативни програми и интерактивна комуникација остају логички непромењени кад год се обаве теоријски могуће промене над базним релацијама (табелама), којима се не губе подаци.⁶³

10. Независност правила интегритета:

Правила интегритета базе података морају бити дефинисана независно од апликативних програма и сачувана у каталогу. Мора бити предвиђена могућност њихове измене у било ком тренутку без непотребног утицаја на постојеће апликације.

11. Независност од дистрибуције:

Све претходно наведене карактеристике су независне од дистрибуирањности базе података. Дистрибуција делова базе на различите локације мора бити невидљива за кориснике базе. Постојеће апликације морају наставити са радом:

- (а) када се дистрибуција података уводи први пут и
- (б) при било којој следећој редистрибуцији података у систему.

12. Забрана субверзије:

Ако релациони систем поседује или може да ради са неким језиком нижег нивоа (у ком се обрађује једна слог у једном тренутку), кроз тај језик се не могу подривати или заобићи правила интегритета задата преко релационог језика⁶⁴.

⁶¹ Сви погледи које је теоријски могуће ажурирати, могу се ажурирати и кроз систем.

⁶² Промене на физичком нивоу (начин на који се чувају подаци, начин на који се приступа посацима) не смеју изазивати промене у апликацијама или интерактивним комуникацијама.

⁶³ Промене на логичком нивоу (табела и колона) не смеју захтевати промене апликација (и интерактивних комуникација) које их користе.

⁶⁴ Који је језик вишег нивоа, јер обрађује више рекорда у једном тренутку.

4. Релациони упитни језик: Structured Query Language (SQL)

Релациони упитни језик *Structured Query Language* (SQL) је доменски-специфичан⁶⁵ језик осмишљен за:

- манипулацију подацима (креирање, ажурирање, заштита и одржавање података),
- очување интегритета података (специфицирање⁶⁶ или програмирање⁶⁷ правила интегритета),
- издавање података (специфицирање или програмирање приступа подацима ради њиховог приказа)

у релационим системима за управљање базама података.

SQL је скуповно базиран, декларативни програмски језик⁶⁸. Међутим, током времена су SQL-у додавана разна процедурална проширења, као што су наредбе за управљање током извршења (control-of-flow constructs). Неке од најпознатијих комерцијално расположивих проширенih верзија стандардног SQL-а су:

Назив стандарда	Скраћени назив	Пун назив
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Назив СУБП		
Interbase / Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
Microsoft / Sybase	T-SQL	Transact-SQL
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language
SAP	SAP HANA	SQL Script

Поред стандардом прописане екstenзије (SQL/PSM extensions) и бројних појединачних додатака, водећи произвођачи комерцијалних СУБП развили су могућности структурног и објектно-оријентисаног програмирања преко интеграције СУБП са другим језицима. Тако, SQL стандард дефинише SQL/JRT екstenзију (SQL Routines and Types for the Java Programming Language) за подршку Јава (Java) коду унутар SQL базе. Microsoft од верзије *SQL Server 2005* користи *SQLCLR* (SQL Server Common Language Runtime) за хостовање управљаних .NET склопова (host managed .NET assemblies) у SQL бази, док су претходне верзије SQL Server-а биле ограничene на спољне проширене усклађене

⁶⁵ Језик намењен само једном апликационом домену, тј. језик посебне намене. Идеја DCJ је близка идеји софтвера као модела разматраног система. Основна намера доменског инжењерства јесте повећати ниво апстракције, семантичко богаство и изражайност џезика у односу на описанаменске језике (C, C++, Јава, C#). Цена за то је мања свеобухватност језика, сужена област примене на само један домен, време утрошено за израду DCJ и (можда) слабије перформансе. Са прагматског гледишта, циљ је брза изградња оперативних модела који могу бити лако валидирани од стране стручњака (доменских експерата) и лако изменљиви (било због потребе да се коригује модел због лошег моделовања, било због потребе да се врши адаптација модела због промене разматраног система).

⁶⁶ Декларативан приступ правилима интегритета.

⁶⁷ Процедурални приступ правилима интегритета.

⁶⁸ За разлику од Паскала (Pascal) или Џа (C) који су императивни програмски језици.

процедуре примарно писане у C/C++. PostgreSQL омогућава програмерима да пишу функције у разним језицима (Perl, Python, Tcl и C).

4.1. Врсте SQL наредби

У стандарду ISO/IEC 9075-1:2011 у одељку 4.11.2, SQL наредбе су разврстане по функционалности у следеће врсте⁶⁹:

1. **SQL-schema** statements that can be used to create, alter, and drop schemas and schema objects.
2. **SQL-data-manipulation** statements that perform insert, update, and delete operations on tables. Execution of an SQL-data statement is capable of affecting more than one row, of more than one table.
3. **SQL-data-retrieve** statements that perform queries on tables. Execution of an SQL-data statement is capable of affecting more than one row, of more than one table.
4. **SQL-transaction** statements that set parameters for, and start or terminate transactions.
5. **SQL-control** statements that may be used to control the execution of a sequence of SQL statements.
6. **SQL-connection** statements that initiate and terminate connections, and allow an SQL-client to switch from an SQL-session with one SQL-server to an SQL-session with another.
7. **SQL-session** statements that set some default values and other parameters of an SQL-session.
8. **SQL-diagnostics** statements that get diagnostics (from the diagnostics area) and signal exceptions in SQL routines.
9. **SQL-dynamic** statements that support the preparation and execution of dynamically-generated SQL-statements, and obtaining information about them.

У средишту наше пажње биће прве три групе наредби.

4.2. SQL-schema statements

Наредбе за прављење објекта базе података.

Некада: Data Definition Language (DDL) statements

CREATE (креирај), DROP (уништи), ALTER (замени)

4.2.1. SQL типови података

Свака колона у SQL табели дефинисана је над неким типом података (data type). ISO/ANSI SQL прописује следеће основне типове.

Ниске знакова (character strings):

- CHARACTER(n) или CHAR(n): стринг фиксне дужине од n знакова
- CHARACTER VARYING(n) или VARCHAR(n): стринг променљиве дужине, максимално n знакова
- NATIONAL CHARACTER(n) или NCHAR(n): стринг фиксне дужине од n знакова који подржава ICS (international character set)
- NATIONAL CHARACTER VARYING(n) или NVARCHAR(n): стринг променљиве дужине, максимално n знакова који подржава ICS (international character set)

Ниске битова (bit strings)

- BIT(n): низ од n битова
- BIT VARYING(n): низ који има до n битова
- BYTE VARYING(n): низ од n бајтова

Бројеви (numbers)

- INTEGER, SMALLINT и BIGINT
- FLOAT, REAL и DOUBLE PRECISION

⁶⁹ Наведено у извornом облику. Навод је из радне верзије стандарда (draft version) која је аутору била доступна са линка наведеног у попису референци.

- NUMERIC(precision, scale) или DECIMAL(precision, scale)

Временски (temporal или date/time)

- DATE: за датум (нпр. 2011.11.23)
- TIME: за време (нпр. 15:51:36)
- TIME WITH TIME ZONE или TIMETZ: исто као TIME, али укључује и податке о временској зони
- TIMESTAMP: DATE и TIME заједно
- TIMESTAMP WITH TIME ZONE или TIMESTAMPTZ: DATE и TIME заједно, али укључује и податке о временској зони.

4.2.2. Домени

Домен је у SQL-у прост, кориснички дефинисан именован објекат који се може користити као алтернатива за предефинисан тип податка над којим се дефинише колона. Може имати default вредност и једно или више ограничења. Домен се креира наредбом:

```
CREATE DOMAIN <naziv domena> [AS] <predefinisani tip>
[DEFAULT <vrednost>]
[[CONSTRAINT <naziv ograničenja>] CHECK (<ograničenje>)];
```

Дефиниција домена се мења наредбом ALTER:

```
ALTER DOMAIN <naziv domena>
SET DEFAULT <vrednost> |
DROP DEFAULT |
ADD [CONSTRAINT <naziv ograničenja>] CHECK (<ograničenje>) |
DROP CONSTRAINT <naziv ograničenja>;
```

Домен се уништава наредбом:

```
DROP DOMAIN <naziv domena>;
```

4.2.3. Табеле и колоне

Креирање табеле:

```
CREATE TABLE <naziv tabele>
(<naziv kolone1> <tip podatka> NOT NULL,
<naziv kolone> <tip podatka> [NOT NULL], ...)
```

Измена дефиниције табеле:

Додавање нове колоне:

```
ALTER TABLE <naziv tabele>
[ADD COLUMN] <definicija kolone>;
```

Измена постојеће колоне:

```
ALTER TABLE <naziv tabele>
[ALTER COLUMN] <naziv kolone>
SET DEFAULT <vrednost> |
DROP DEFAULT;
```

Избацување колоне из табеле:

```
ALTER TABLE <naziv tabele>
DROP [COLUMN] <naziv kolone>;
```

Додавање или избацаивање ограничења на вредности:

```
ALTER TABLE <naziv tabele>
ADD [CONSTRAINT <naziv ograničenja> ] <ограничење табеле> |
DROP CONSTRAINT <naziv ograničenja>;
```

Уништавање табеле:

```
DROP TABLE <naziv tabele>;
```

Креирање виртуалне табеле (погледа, приказа):

```
CREATE VIEW <naziv pogleda> [(<naziv kolone1>, {<naziv kolone1>})]
AS
<SQL upit>
[WITH [LOCAL | CASCADED] CHECK OPTION];
```

Измена дефиниције виртуалне табеле:

```
ALTER VIEW <naziv pogleda> [(<naziv kolone1>, {<naziv kolone1>})}
AS
<SQL upit>
[WITH [LOCAL | CASCADED] CHECK OPTION];
```

Уништавање виртуалне табеле:

```
DROP VIEW <naziv pogleda>;
```

4.2.4. Индекси

Индекси су структуре података које олакшавају и чине ефикаснијим приступ подацима базе. Вредности индексираних колона могу бити јединствене уколико се при креирању изабере опција UNIQUE.

```
CREATE [UNIQUE] INDEX <naziv indeksa>
ON (<naziv tabele> ( <naziv kolone1> [, <naziv kolone2>, ..]) ;
```

Уништавање се врши наредбом:

```
DROP INDEX <naziv indeksa>;
```

4.2.5. Схеме

Схема (schema) је именова група логички повезаних објеката у бази; она обједињује све објекте који деле исти именски простор (простор именовања). Схема може садржати једну или више табела, а свака табела може припадати логички тачно једној схеми. Схема се креира наредбом:

```
CREATE SCHEMA <naziv sheme>;
```

Уништавање схеме може бити са опцијом CASCADE (при чему се уништава схема и сви објекти из ње) или опцијом RESTRICT (уништавање шеме која је празна) и остварује се наредбом:

```
DROP SCHEMA <naziv sheme> CASCADE | RESTRICT;
```

Пун назив објекта базе података састоји се из више делова; кад се жели навести потпуно квалификовано име објекта, наводи се троделно (у T-SQL је четвороделно) име:

```
<naziv_baze>.<naziv_sheme>.<naziv_objekta>
```

4.3. SQL-data-manipulation statements

Наредбе за ажурирање података.

Некада: Data Manipulation Language (DML) statements

INSERT (убаци), UPDATE (промени), DELETE (избаци)

Додавање нових редова у табелу:

```
[ WITH <common_table_expression> [ ,...n ] ]
INSERT
{
    [ TOP ( expression ) [ PERCENT ] ]
    [ INTO ]
    { <object> | rowset_function_limited
        [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]
    }
    {
        [ ( column_list ) ]
        [ <OUTPUT Clause> ]
        { VALUES ( { DEFAULT | NULL | expression } [ ,...n ] ) [ ,...n ]
        | derived_table
        | execute_statement
        | <dml_table_source>
        | DEFAULT VALUES
        }
    }
} [ ; ]
```

Краће:

```
INSERT INTO <naziv_tabele> VALUES (<vrednost1>, <vrednost2> ...);
```

Промена вредности у табели:

```
[ WITH <common_table_expression> [...n] ]
UPDATE
    [ TOP ( expression ) [ PERCENT ] ]
    { { table_alias | <object> | rowset_function_limited
        [ WITH ( <Table_Hint_Limited> [ ...n ] ) ]
    }
    | @table_variable
    }
    SET
        { column_name = { expression | DEFAULT | NULL }
        | { udt_column_name. { { property_name = expression | field_name = expression } | method_name ( argument [ ,...n ] ) }
        }
        | column_name { .WRITE ( expression , @Offset , @Length ) }
        | @variable = expression
        | @variable = column = expression
        | column_name { += | -= | *= | /= | %= | &= | ^= | |= } expression
        | @variable { += | -= | *= | /= | %= | &= | ^= | |= } expression
        | @variable = column { += | -= | *= | /= | %= | &= | ^= | |= }
    expression
} [ ,...n ]

[ <OUTPUT Clause> ]
[ FROM{ <table_source> } [ ,...n ] ]
[ WHERE { <search_condition>
    | { [ CURRENT OF
        { { [ GLOBAL ] cursor_name
            | cursor_variable_name
        }
    ]
}
} ]
```

```

        ]
        [ OPTION ( <query_hint> [ ,...n ] ) ]
[ ; ]

<object> ::= {
    [ server_name . database_name . schema_name .
    | database_name .[ schema_name ] .
    | schema_name .
    ]
    table_or_view_name}

```

Краће:

```

UPDATE TABLE <naziv_tabele>
SET <naziv_kolone1> = <izraz1> [ , <naziv_kolone2> = <izraz2>, ... ]
[WHERE <uslov>];

```

Брисање редова у табели:

```

[ WITH <common_table_expression> [ ,...n ] ]
DELETE
    [ TOP ( expression ) [ PERCENT ] ]
    [ FROM ]
    { { table_alias
        | <object>
        | rowset_function_limited
        [ WITH ( table_hint_limited [ ...n ] ) ]
        | @table_variable
    }
    [ <OUTPUT Clause> ]
    [ FROM table_source [ ,...n ] ]
    [ WHERE { <search_condition>
        | { [ CURRENT OF
            { { [ GLOBAL ] cursor_name }
            | cursor_variable_name } ] } }
    ]
    [ OPTION ( <Query Hint> [ ,...n ] ) ]
[; ]

<object> ::=
{
    [ server_name.database_name.schema_name.
    | database_name. [ schema_name ] .
    | schema_name .
    ]
    table_or_view_name
}

```

Краће:

```

DELETE TABLE <naziv_tabele>
[WHERE <uslov>];

```

4.4. SQL-data-retrieve statements

Наредбе за издавање података. Алтернативно: наредбе за приказ података.
Некада: Data Retrieve Language (DRL) statements

SELECT (прикажи)

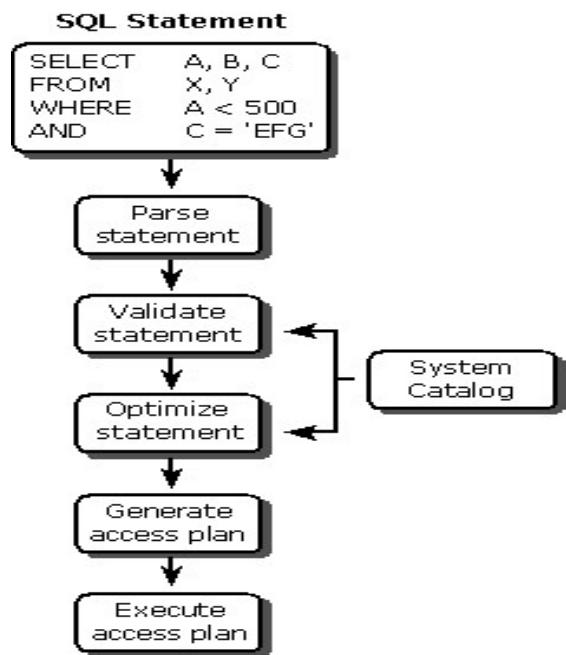
4.4.1. Поступак извршавања SQL упита

Приликом извршавања SQL упита (SQL statement, SQL query), систем за управљање базама података (СУБП) извршава следећих пет корака:

1. СУБП прво парсира SQL упит: дели упит у поједине речи (које называемо токени), проверава њихове међусобне синтаксне односе, проверава да ли се упит састоји из обавезних делова, да ли садржи кључне речи и да ли су оне тачно записане итд. У овом кораку се могу детектовати синтаксне грешке.
2. Потом СУБП врши валидацију SQL упита. При томе се користи каталог система (системски каталог, речник података). Да ли све табеле наведене у SQL упиту постоје у бази података? Да ли све наведене колоне постоје и да ли су имена колона недвосмислена? Да ли корисник има потребне привилегије да изврши овај упит? Одређене семантичке грешке могу се детектовати у овом кораку.
3. Ако је у претходном кораку све уреду, СУБП генерише план извршења SQL упита (*execution plan, (exp)*; алтернативни назив:

приступну план, *access plan*). План извршења SQL упита је бинарна презентација акција које су потребне за обављање упита; то је СУБП еквивалент извршног кода програмских језика (*executable code*), тј. $(\text{exp})_{\text{DBMS}} = (\text{exe})_{\text{PL}}$.

4. СУБП оптимизује план извршења. Испитују се различити начине његовог извршења: Може ли индекс да се користи за убрзавање претраге? Да ли да се прво изврши рестрикција над Табелом А (уклоне редови који не одговарају услову претраге), па да се онда изврши спајање са табелом Б, или би било боље обрнуто (прво спајање, па онда рестрикција)? Може ли се секвенцијално претраживање целе табеле табеле избеги? Ако не може, како га ограничити на неки подскуп података из табеле?
- Након процењивања различитих планова извршења, СУБП бира најефикаснији.
5. СУБП приступа извршењу SQL упита по изабраном плану. Добијени резултат⁷⁰ се прослеђује кориснику.



Слика 1. Кораци у извршењу SQL наредбе

Претходни кораци у извршавању SQL упита се разликују по приступу различитим објектима базе (каталог, индекси, кластери, табеле, погледи итд) и њиховој учесталости, па сходно томе и по количини времена које троше. Парсирање SQL упита не захтева приступ бази података и може да се уради веома брзо. Оптимизација је, с друге стране, процесорски веома захтевна активност и неопходан јој је приступ каталогу система. За сложене упите над више табела, у поступку оптимизацију, испитује се више хиљада

⁷⁰ Резултат = подаци и/или порука

различитих начина извођења истог упита. Међутим, трошкови неефикасног извршавања упита су обично толико високи, да се време проведено у оптимизација више него исплати, јер значајно повећава брзину извршења упита. Резултати оптимизације су још значајнији ако се изабрани (а то значи и оптимизовани) план извршења вишеструком извршава⁷¹.

4.4.2. Синтакса наредбе SELECT

Наредба SELECT убраја се у најсложеније SQL наредбе. Овде се наводи њен општи облик, без детаљних објашњења:

```
SELECT [ ALL | DISTINCT ]
[ TOP ( expression ) [ PERCENT ] [ WITH TIES ] ]
<select_list>
<select_list> ::= 
{
  *
  | { table_name | view_name | table_alias }.*
  | {
    [ { table_name | view_name | table_alias }. ]
    { column_name | $IDENTITY | $ROWGUID }
    | udt_column_name [ { . | :: } { { property_name | field_name }
      | method_name ( argument [ ,...n] ) } ]
    | expression
    [ [ AS ] column_alias ]
  }
  | column_alias = expression
} [ ,...n ]
```

Краће, основни облик упита у SQL-у :

```
SELECT <lista kolona>
FROM <llista tabela>
WHERE <uslov>;
```

Листом колона задаје се операција *пројекције* жељених колона; листом табела одређује се извор података (који се добија *спајањем*, најчешће); условом се задаје услов *рестрикције (селекције)* жељених редова. Клаузуле SELECT и FROM су обавезне, а клаузула WHERE није.

⁷¹ То се дешава приликом: (1) вишеструког извршавања истих упита, (2) извршавања параметризованих упита и (3) извршавања упита чија се структура не мења, већ се мењају само вредности у WHERE клузули.

5. Референце

ЛИТЕРАТУРА

- [1] Lazarević Branislav i dr., *Baze podataka*, FON, Beograd, 2003.
- [2] Codd Edgar F., *Data Models in Database Management*, објављено у: *Proceedings of Workshop on Data Abstraction, Databases, and Conceptual Modelling*, Michael L. Brodie & Stephen N. Yilles, eds., Pingree Park, Colo., June 1980.
- [3] Могин Павле и Луковић Иван, *Принципи база података*, ФТН и МП "Stylos", Нови Сад, 1996.
- [4] Codd Edgar F., *A Relational Model of Data for Large Shared Data Banks*, *Communications of the ACM*, Association for Computing Machinery, 13 (6): 377–87, June 1970.
- [5] Date Christopher J., *An Introduction to Database Systems*, 6th edition, Addison-Wesley, 1995.
- [6] Codd Edgar F., *Is Your DBMS Really Relational?*, *ComputerWorld*, 14. October 1985.
- [7] Codd Edgar F., *Does Your DBMS Run By the Rules?*, *ComputerWorld*, 21. October 1985.
- [8] Darwen Hugh and Date C. J., *Databases, Types, and The Relational Model: The Third Manifesto*, 3rd edition, Addison-Wesley, 2006.
- [9] Codd Edgar F., *The Relational Model for Database Management: Version 2*, Addison-Wesley, 1990.

ИНТЕРНЕТ

- [1] ISO/IEC FDIS 9075-1 Information technology - Database languages - SQL, Part 1: Framework
http://www.jtc1sc32.org/doc/N2151-2200/32N2153T-text_for_ballot-FDIS_9075-1.pdf
- [2] Data Types (Transact-SQL)
<https://msdn.microsoft.com/en-us/library/ms187752.aspx>
- [3] Proposed foundation for future (relational) database systems
<http://thethirdmanifesto.com>
- [4] Codd's 12 Rules (for a relational database product)
<http://computing.derby.ac.uk/c/codds-twelve-rules/>

6. Прилог

У прилогу је изложен студијски примера: (скраћени) поступак развоја базе података која се користи за постављање упита. Имплементација је урађена у *Transact-SQL-y* (DBMS: *SQL Server 2012*, IDE: *SQL Server Management Studio*).

6.1. Концептуализација

6.1.1. Вербални модел

Неопходно је направити софтвер који омогућава рачунарски подржану евиденцију о основним ентитетима факултетског информационог система. Факултет има два одсека (студијска програма): Менаџмент и Информатика. На сваком од одсека постоје по три смера (студијске групе). На Информатицу то су: Софтверско инжењерство, Информациони системи и Информационе технологије. На Менаџменту то су: Управљање производњом, Управљање пословањем и Управљање кадровима. Прва година студија је заједничка за све студенте (тј. студенти не бирају смер на првој години), а од друге године сваки студент се одлучује за један смер; није дозвољено студирање на више смерова истовремено. За сваког студента се води евиденција о следећим подацима:

- Број индекса (обавезно; прве две цифре значе годину уписа, а преостале три редни број),
- Презиме и име (обавезно),
- Средње име (није обавезно),
- Датум рођења (обавезно),
- Година студија (обавезно; од 1 до 4),
- Да ли студент поседује возачку дозволу и које је она категорије (није обавезно),
- Слика студента (није обавезна),
- Износ студентског кредита (обавезно, мада неки студенти не примају кредит),
- Износ стипендије (није обавезно).

Предмети на факултету имају свој назив, врсту (обавезни или изборни) и ЕСПБ. Сваки предмет у надлежности је једног од два наведена одсека („припада“ одсеку). Студент полаже предмете на испитима. На испиту се бележи датум полагања и оцена; петице се не уписују (не води се евиденција о неположеним испитима); у случају поништења испита и поновног полагања, уписује се само последња оцена (не води се историја полагања испита). Основни подаци о студентима (име, презиме, средње име), предметима (назив), смеровима (назив) и одсечима (назив), због законских обавеза, записани су ћириличним писмом српскога језика.

Врло често су потребни:

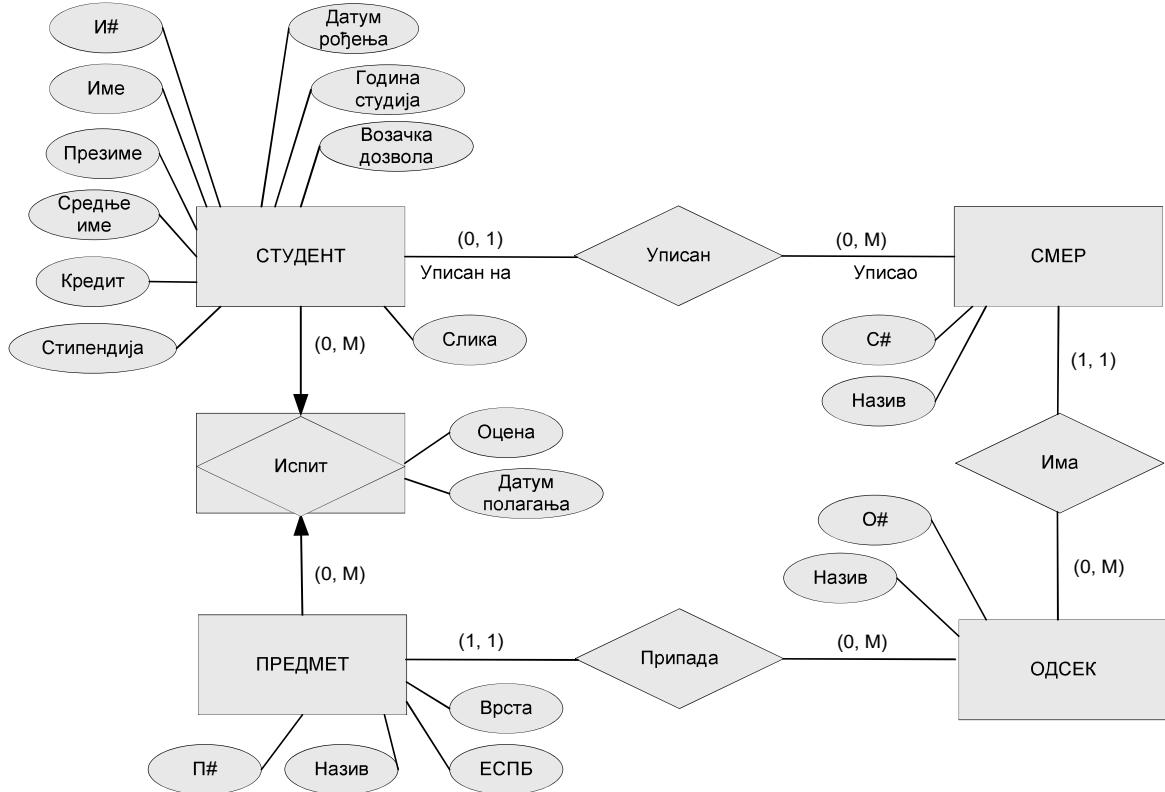
- Подаци о студентима са одсека Информатика (број индекса, име, презиме, година студија, одсек, а може и смер),
- Подаци о студентима са одсека Менаџмент (број индекса, име, презиме, година студија, одсек, а може и смер),
- Подаци о положеним испитима (који студент је положио, који предмет, када и са којом оценом),
- Подаци о студентским „примањима“ (стипендије, студентски кредити, укупно).

Сем претходно наведених, нема додатних посебних (пословних) ограничења.

6.1.2. Концептуални модел

6.1.2.1. Структура

А) Дијаграм објекта и веза



Концептуални модел: ДОВ Студент положио предмет

Б) Речник података

ЕНТИТЕТ	АТРИБУТ	НАПОМЕНА
ОДСЕК	О#: integer, not null Назив: nvarchar(12), not null	Шифра одсека
СМЕР	С#: integer, not null Назив: nvarchar(24), not null	Шифра смера
СТУДЕНТ	БИ: integer, not null Име: nvarchar(7), not null Презиме: nvarchar(7), not null СредњеИме: nvarchar(7), null ДатумРођења: date, not null Слика: varbinary(max), null ГодинаСтудија: integer, not null, [1..4] ВозачкаКатегорија: null, ('А', 'Б', 'Ц', 'Д', 'Е') Кредит: integer, not null, Кредит >= 0, default=0 Стипендија: integer, null Стипендија > 0	Број индекса
ПРЕДМЕТ	П#: integer, not null Назив: nvarchar(22), not null Врста: nchar(3), not null, ('ИЗБ', 'ОБВ') ЕСПБ: integer, null, [2..8]	Шифра предмета Изборни, Обавезни Евро.сис.пренос.бодова
ИСПИТ	ДатумПолагања: date, not null, Оцена: integer, not null, [6..10]	

6.1.2.2. Ограничења

Нема.

6.2. Спецификација

6.2.1. Схема релационе базе података: базне релације

ОДСЕК (О#, Назив)
СМЕР (С#, Назив, Одсек#)
СТУДЕНТ (БИ, Име, Презиме, СредњеИме, ДатумРођења, Слика, ГодинаСтудија,
ВозачкаКатегорија, Кредит, Стипендија, Смер#)
ПРЕДМЕТ (П#, Назив, Врста, ЕСПБ, Одсек#)
ИСПИТ (БИ, П#, ДатумПолагања, Оцена)

6.2.2. Схема релационе базе података: изведене релације (погледи)

ИНФ_СТУДЕНТ (БИ, Име, Презиме, ГодинаСтудија, Одсек), Одсек = `Информатика`
МЕН_СТУДЕНТ (БИ, Име, Презиме, ГодинаСтудија, Одсек), Одсек = `Менаџмент`
СТУД_ПОЛОЖИО (БИ, Име, Презиме, Предмет, Врста, ЕСПБ, ДатумПолагања, Оцена)
СТУД_ФИНАНС (БИ, Име, Презиме, Кредит, Стипендија, Укупно)

6.2.3. Правила интегритета

Нема.

6.3. Имплементација

6.3.1. DDL

DDL = Data Definition Language

```
-- *****
-- Креирање базе података
-- *****

CREATE DATABASE fakultet
    COLLATE Serbian_Cyrillic_100_CS_AI;
USE fakultet;

-- *****
-- Креирање базних табела
-- *****

CREATE TABLE ODSEK(
    O#          tinyint not null primary key,
    Naziv      nvarchar(12) not null
);

CREATE TABLE SMER(
    S#          tinyint not null primary key,
    Naziv      nvarchar(24) not null,
    Odsek#     tinyint not null foreign key references ODSEK(O#)
);

CREATE TABLE STUDENT(
    I#          smallint not null primary key,           -- број индекса
    Ime        nvarchar(7) not null,
    Prezime    nvarchar(7) not null
        check (Prezime LIKE '[А-Ш]%' ),
    Slika      varbinary(max) null,
    DatRod    date not null,                            -- датум рођења
    GodStud   tinyint not null                         -- година студија
);
```

```

VozKat    check ((GodStud >= 1) and (GodStud <= 4)),
           nchar(1) null                                -- возачка категорија
Kredit     check (VozKat in ('А','Б','Ц','Д','Е')),
           tinyint not null                           -- студентски кредит
           default 0
           check (Kredit >= 0),
Stip       tinyint null                            -- стипендија
           check ((Stip > 0) or (Stip is NULL)),
Smer#      tinyint null
           foreign key references SMER(S#)
);

ALTER TABLE STUDENT
DROP COLUMN Slika;

ALTER TABLE STUDENT
ADD SredIme nvarchar(7) null;                      -- средње име

CREATE TABLE PREDMET(
P#        tinyint not null primary key,
Naziv    nvarchar(22) not null unique,
Vrsta    nchar(3) not null check (Vrsta in ('ИЗБ', 'ОБВ')),
           -- (ИЗБ)орни или (ОБ)а(В)езни предмет
ESPB     tinyint null,   -- (Е)вропски (С)истем (П)реносних (Б)одова
Odsek#   tinyint not null
           foreign key references ODSEK(O#)
);

CREATE TYPE tocena
FROM tinyint not null;

CREATE TABLE ISPIT(
Indeks#   smallint not null,          -- број индекса
Predmet#   tinyint not null,          -- шифра предмета
DatPol    date not null,             -- датум полагања испита
Ocena     tocena
           check ((Ocena >= 6) and (Ocena <= 10)),
constraint PK_Ispit primary key (Indeks#, Predmet#),
constraint FK_Ispit_Student foreign key (Indeks#) references STUDENT,
constraint FK_Ispit_Predmet foreign key (Predmet#) references PREDMET
);

-- *****
-- Креирање виртуелних табела (погледа)
-- *****

CREATE VIEW INF_STUDENT AS
SELECT
I# as [Број индекса],
Ime as [Име],
Prezime as [Презиме],
GodStud as [Година студија],
SM.Naziv as [Смер],
O.Naziv as [Одсек]
FROM STUDENT ST
JOIN SMER SM ON ST.Smer# = SM.S#
JOIN ODSEK O ON SM.Odsek# = O.O#
WHERE O.Naziv LIKE N'Информатика';

```

```

CREATE VIEW MEN_STUDENT AS
SELECT
    I# as [Број индекса],
    Ime as [Име],
    Prezime as [Презиме],
    GodStud as [Година студија],
    SM.Naziv as [Смер],
    O.Naziv as [Одсек]
FROM STUDENT ST
    JOIN SMER SM ON ST.Smer# = SM.S#
    JOIN ODSEK O ON SM.Odsek# = O.O#
WHERE O.Naziv LIKE N'Менаџмент';

CREATE VIEW STUDENT_POLOZIO AS
SELECT
    ST.I# as [Број индекса],
    Ime as [Име],
    Prezime as [Презиме],
    Naziv as [Предмет],
    Vrsta as [Врста],
    ESPB as [ЕСПБ],
    DatPol as [Датум испита],
    Ocena as [Оцена]
FROM STUDENT ST
    JOIN ISPIT I ON ST.I# = I.Indeks#
    JOIN PREDMET P ON I.Predmet# = P.P#;

CREATE VIEW STUDENT_FINANS AS
SELECT
    I# as [Број индекса],
    Ime as [Име],
    Prezime as [Презиме],
    Kredit as [Кредит],
    ISNULL(Stip, 0) as [Стипендија],
    [Укупно] = (Kredit + ISNULL(Stip, 0))
FROM STUDENT;

```

6.3.2. DML

DML = Data Manipulation Language

```

USE fakultet;

INSERT INTO ODSEK VALUES
    (100, N'Информатика'),
    (200, N'Менаџмент');

INSERT INTO SMER VALUES
    (10, 'Софтверско инжењерство',      100),
    (20, 'Информациони системи',          100),
    (30, 'Информационе технологије',       100),
    (50, 'Управљање производњом',        200),
    (60, 'Управљање пословањем',        200),
    (70, 'Управљање кадровима',         200);

INSERT INTO STUDENT
    (I#, Ime, Prezime, SredIme, DatRod, GodStud, VozKat, Kredit, Stip, Smer#)
VALUES
    (17002, 'Ана',     'Костић', NULL,      '1998.07.27', 1, NULL,      100, NULL, NULL),

```

```

(17014, 'Ана', 'Марић', 'Јова', '1998.05.16', 1, 'Б',      100, NULL, NULL),
(17008, 'Анка', 'Анић', 'Саша', '1998.05.23', 1, NULL,      100, 50, NULL),
(16002, 'Аница', 'Барић', NULL, '1997.09.23', 2, 'Б',      150, 20, 10),
(16014, 'Мара', 'Илић', 'Мита', '1998.09.11', 2, 'А',      150, NULL, 20),
(16008, 'Мила', 'Јовић', 'Саша', '1998.07.27', 2, 'Ц', default, 50, 60),
(15002, 'Аца', 'Костић', 'Јова', '1996.06.17', 3, NULL,      200, NULL, 10),
(15014, 'Мома', 'Којић', NULL, '1996.06.17', 3, 'Б',      200, 20, 20),
(15008, 'Јова', 'Кун', 'Саша', '1996.12.12', 3, NULL,      200, NULL, 50),
(14002, 'Лаза', 'Марић', 'Раша', '1995.02.21', 4, NULL,      220, 20, 10),
(14014, 'Јова', 'Киш', NULL, '1995.03.23', 4, 'Ц',      220, 20, 50);

```

>>> . . .

6.3.3. RDB

RDB = Relational Database (релациона база података)

ОДСЕК

О# Назив

100	Информатика
200	Менаџмент

СМЕР

С#	Назив	Одсек#
10	Софтверско инжењерство	100
20	Информациони системи	100
30	Информационе технологије	100
50	Управљање производњом	200
60	Управљање пословањем	200
70	Управљање кадровима	200

СТУДЕНТ

И#	Име	Презиме	СредИме	ДатРод	ГодСтуд	ВозКат	Кредит	Стип	Смер#
14002	Лаза	Марић	Раша	1995-02-21	4	NULL	220	20	10
14014	Јова	Киш	NULL	1995-03-23	4	Ц	220	20	50
15002	Аца	Костић	Јова	1996-06-17	3	Б	200	NULL	10
15008	Јова	Кун	Саша	1996-12-12	3	NULL	200	NULL	50
15014	Мома	Којић	NULL	1996-06-17	3	Б	200	20	20
16002	Аница	Барић	NULL	1997-09-23	2	Б	150	20	10
16008	Мила	Јовић	Саша	1998-07-27	2	Ц	0	50	60
16014	Мара	Илић	Мита	1998-09-11	2	А	150	NULL	20
17002	Ана	Костић	NULL	1998-07-27	1	NULL	100	NULL	NULL
17008	Анка	Анић	Саша	1998-05-23	1	Б	100	50	NULL
17014	Ана	Марић	Јова	1998-05-16	1	Б	100	NULL	NULL

ПРЕДМЕТ

П#	Назив	Врста	ЕСПБ	Одсек#
1	Математика 1	ОБВ	6	100
2	Програмирање 1	ОБВ	6	100

3 Економија	ОБВ	8	200
4 Увод у ИС	ОБВ	6	100
5 Основи организације	ОБВ	6	200
6 Менаџмент	ОБВ	6	100
7 Базе података	ОБВ	6	100
8 Социологија	ИЗБ	4	200
9 Психологија	ИЗБ	4	200
10 Логика	ИЗБ	4	200
11 Философија	ИЗБ	4	200
12 Оперативни системи	ОБВ	5	100
13 Основи квалитета	ОБВ	6	200
14 Архитектура рачунара	ОБВ	5	100
15 Производно инжењерство	ОБВ	6	200
16 Математика 2	ОБВ	5	100
17 Програмирање 2	ОБВ	4	100
18 Маркетинг	ОБВ	6	200
19 Финансије	ОБВ	6	200
20 Алгоритми	ИЗБ	4	100
21 Структуре података	ОБВ	6	100
22 Управљање залихама	ОБВ	6	200

ИСПИТ

И#	П#	Дат	Пол	Оцена
14002	1	2014-06-26		9
14002	2	2016-01-28		6
14002	3	2014-01-28		6
14002	5	2014-01-28		6
14002	16	2014-09-21		8
14002	17	2015-01-28		7
14014	1	2014-06-21		6
14014	3	2014-06-27		7
...	
17008	1	2017-09-29		8
17008	5	2017-01-28		10
17014	1	2017-06-29		8
17014	2	2016-06-29		8
17014	5	2017-06-28		8
17014	16	2017-09-29		8

6.3.4. ADS

ADS = Application Data Submodels

INF_STUDENT

Број	индекса	Име	Презиме	Година студија	Смер	Одсек
	14002	Лаза	Марић	4	Софтверско инжењерство	Информатика
	15002	Аца	Костић	3	Софтверско инжењерство	Информатика
	15014	Мома	Којић	3	Информациони системи	Информатика
	16002	Аница	Барич	2	Софтверско инжењерство	Информатика
	16014	Мара	Илић	2	Информациони системи	Информатика

MEN_STUDENT

Број индекса	Име	Презиме	Година студија	Смер	Одсек
14014	Јова	Киш	4	Управљање производњом	Менаџмент
15008	Јова	Кун	3	Управљање производњом	Менаџмент
16008	Мила	Јовић	2	Управљање пословањем	Менаџмент

STUDENT_POLOZIO

Број индекса	Име	Презиме	Предмет	Врста	ЕСПБ	Датум испита	Оцена
14002	Лаза	Марић	Математика 1	ОБВ	6	2014-06-26	9
14002	Лаза	Марић	Програмирање 1	ОБВ	6	2016-01-28	6
14002	Лаза	Марић	Економија	ОБВ	8	2014-01-28	6
14002	Лаза	Марић	Основи организације	ОБВ	6	2014-01-28	6
14002	Лаза	Марић	Математика 2	ОБВ	5	2014-09-21	8
14002	Лаза	Марић	Програмирање 2	ОБВ	5	2015-01-28	7
14014	Јова	Киш	Математика 1	ОБВ	6	2014-06-21	6
14014	Јова	Киш	Економија	ОБВ	8	2014-06-27	7
14014	Јова	Киш	Основи организације	ОБВ	6	2014-06-28	7
14014	Јова	Киш	Математика 2	ОБВ	5	2014-09-21	6
15002	Аца	Костић	Математика 1	ОБВ	6	2015-09-09	10
15002	Аца	Костић	Програмирање 1	ОБВ	6	2016-09-26	7
15002	Аца	Костић	Економија	ОБВ	8	2015-09-26	7
15002	Аца	Костић	Основи организације	ОБВ	6	2015-09-28	7
15002	Аца	Костић	Математика 2	ОБВ	5	2016-09-09	10
15002	Аца	Костић	Програмирање 2	ОБВ	5	2016-09-26	7
15008	Јова	Кун	Математика 1	ОБВ	6	2015-09-09	9
15008	Јова	Кун	Економија	ОБВ	8	2015-01-28	8
15008	Јова	Кун	Основи организације	ОБВ	6	2015-01-28	6
15008	Јова	Кун	Математика 2	ОБВ	5	2015-09-29	9
15008	Јова	Кун	Управљање залихама	ОБВ	6	2016-01-28	6
15014	Мома	Којић	Математика 1	ОБВ	6	2015-09-09	8
15014	Мома	Којић	Програмирање 1	ОБВ	6	2016-06-27	9
15014	Мома	Којић	Економија	ОБВ	8	2015-06-27	7
15014	Мома	Којић	Основи организације	ОБВ	6	2015-06-28	8
15014	Мома	Којић	Математика 2	ОБВ	5	2016-09-09	6
15014	Мома	Којић	Програмирање 2	ОБВ	5	2016-06-27	8
16002	Аница	Барил	Математика 1	ОБВ	6	2016-06-29	9
16002	Аница	Барил	Програмирање 1	ОБВ	6	2016-09-26	10
16002	Аница	Барил	Економија	ОБВ	8	2016-09-26	8
16002	Аница	Барил	Основи организације	ОБВ	6	2016-09-28	9
16002	Аница	Барил	Програмирање 2	ОБВ	5	2017-09-26	10
16008	Мила	Јовић	Математика 1	ОБВ	6	2016-06-27	6
16008	Мила	Јовић	Економија	ОБВ	8	2016-01-24	9
16008	Мила	Јовић	Основи организације	ОБВ	6	2016-01-28	9
16008	Мила	Јовић	Математика 2	ОБВ	5	2016-09-27	6
16014	Мара	Илић	Математика 1	ОБВ	6	2016-01-28	7
16014	Мара	Илић	Програмирање 1	ОБВ	6	2016-06-23	8
16014	Мара	Илић	Основи организације	ОБВ	6	2016-06-28	9
16014	Мара	Илић	Математика 2	ОБВ	5	2016-06-28	7
16014	Мара	Илић	Програмирање 2	ОБВ	5	2017-01-24	8
17002	Ана	Костић	Математика 1	ОБВ	6	2017-06-24	6
17002	Ана	Костић	Програмирање 1	ОБВ	6	2016-09-22	9
17002	Ана	Костић	Економија	ОБВ	8	2017-09-22	10
17002	Ана	Костић	Основи организације	ОБВ	6	2017-09-28	8
17002	Ана	Костић	Програмирање 2	ОБВ	5	2016-09-22	9
17008	Анка	Анић	Математика 1	ОБВ	6	2017-09-29	8
17008	Анка	Анић	Основи организације	ОБВ	6	2017-01-28	10
17014	Ана	Марић	Математика 1	ОБВ	6	2017-06-29	8
17014	Ана	Марић	Програмирање 1	ОБВ	6	2016-06-29	8
17014	Ана	Марић	Основи организације	ОБВ	6	2017-06-28	8
17014	Ана	Марић	Математика 2	ОБВ	5	2017-09-29	8