მონაცემთა ბაზის ლოგიკური დიზაინი და რელაციური მოდელი

მონაცემთა ბაზის შემუშავებისას მონაცემთა ბაზის ანალიზის ფაზის ბოლოს, სისტემებისა და მონაცემთა ბაზების ანალიტიკოსებს საკმაოდ მკაფიოდ აქვთ გააზრებული მონაცემთა შენახვისა და წვდომის მოთხოვნები. ამასთან, ანალიზის დროს შემუშავებული მონაცემთა მოდელი არაა დაკავშირებული მონაცემთა ბაზის პროგრამული რეალიზაციის რომელიმე კონკრეტულ ტექნოლოგიასთან. მონაცემთა ბაზის პროგრამულ განხორციელებამდე, კონცეპტუალური მონაცემების მოდელი უნდა აისახოს მონაცემთა მოდელში, რომელიც თავსებადი იქნება ამ მიზნისთვის გამოყენებული მონაცემთა ბაზის მართვის სისტემასთან.

მონაცემთა ბაზის დაპროექტებისას განხორციელებული ქმედებები მოითხოვს მონაცემთა ბაზის ანალიზის დროს შემუშავებული მონაცემთა შენახვის მოთხოვნების გარდაქმნას სახელმძღვანელო სპეციფიკაციებად მონაცემთა ბაზის დანერგვის პროცესში. სპეციფიკაციების ორი ფორმა არსებობს:

1. ლოგიკური სპეციფიკაციები, რომლებიც ასახავს კონცეპტუალურ მოთხოვნებს მონაცემთა მოდელში, რომელიც დაკავშირებულია მონაცემთა ბაზის მართვის კონკრეტულ სისტემასთან.
2. ფიზიკური სპეციფიკაციები, სადაც მითითებულია მონაცემთა შენახვის ყველა პარამეტრი, რომლებიც შემდეგ გამოიყენება მონაცემთა ბაზის დანერგვისთვის. ამ ფაზის განმავლობაში მონაცემთა ბაზა რეალურად განისაზღვრება მონაცემთა განსაზღვრის ენის გამოყენებით.

გავიხილოთ მონაცემთა ბაზის ლოგიკური დიზაინი, კერძოდ - **რელაციური მონაცემების მოდელი.** მონაცემთა ბაზის ლოგიკური დიზაინი წარმოადგენს მონაცემების კონცეპტუალური მოდელის ლოგიკურ მონაცემთა მოდლად გარდაქმნის პროცესს, რომელიც შეესაბამება და თავსებადია მონაცემთა ბაზის სპეციფიკურ ტექნოლოგიასთან. ხშირად მონაცემთა ბაზის გამოცდილი შემმუშავებელი მონაცემთა ბაზის ლოგიკურ დიზაინს ქმნის მონაცემთა კონცეპტუალური მოდელირების პარალელურად, თუ საწყის ეტაპზევე ცნობილია მონაცემთა ბაზის შესაქმნელი ტექნოლოგიის ტიპი. ამასთან, მნიშვნელოვანია მოვახდინოთ მათი, როგორც ცალკეული ნაბიჯების განხილვა, რათა კონცენტრირება იყოს მონაცემთა ბაზის განვითარების თითოეულ მნიშვნელოვან ნაწილზე. კონცეპტუალური მონაცემების მოდელირება გულისხმობს ორგანიზაციის გაგებას - მოთხოვნების სწორად წარმოჩენას. მონაცემთა ბაზის ლოგიკური დიზაინი წარმოადგენს მონაცემთა ბაზის სტაბილური სტრუქტურების შექმნას - ტექნიკურ ენაზე მოთხოვნების სწორად გამოხატვას. ორივე მნიშვნელოვანი ნაბიჯია, რომელიც ფრთხილად უნდა შესრულდეს.

მიუხედავად იმისა, რომ არსებობს მონაცემების სხვა ლოგიკური მოდელები, ჩვენ განვიხილავთ მონაცემთა რელაციურ მოდელს ორი მიზეზის გამო: პირველი, მონაცემების რელაციური მოდელი ყველაზე ხშირად გამოიყენება მონაცემთა ბაზის თანამედროვე პროგრამებში; მეორე, მონაცემთა ბაზის ლოგიკური დაპროექტების ზოგიერთი პრინციპები რელაციური მოდელისთვის სხვა ლოგიკურ მოდელებზეც ვრცელდება.

ადრე განხილული მაგალითების საშუალებით ჩვენ არაფორმალურად განვიხილეთ მონაცემების რელაციური მოდელი. ამასთან, მნიშვნელოვანია აღინიშნოს, რომ რელაციური მონაცემების მოდელი ლოგიკური მონაცემების მოდელის ფორმაა და, როგორც ასეთი, იგი განსხვავდება მონაცემთა კონცეპტუალური მოდელებისგან. ამრიგად, E-R მონაცემთა მოდელი არ არის რელაციური მონაცემების მოდელი და E-R მოდელი შეიძლება არ დაემორჩილოს კარგად სტრუქტურირებული რელაციური მონაცემების მოდელის წესებს (რომელსაც ნორმალიზაცია ეწოდება), რადგან E-R მოდელი შეიქმნა სხვა მიზნებისთვის - მონაცემთა მოთხოვნების და ბიზნესის წესების გაგება-გაზიარებისათვის - და არა მონაცემთა სტრუქტურირებისათვის.

მონაცემთა ბაზის ლოგიკური დაპროექტების მიზანია კონცეპტუალური მოდელის (რომელიც წარმოადგენს ორგანიზაციის მოთხოვნებს მონაცემთა მიმართ) თარგმნას მონაცემთა ბაზის ლოგიკურ პროექტში, რომლის განხორციელება შესაძლებელია მონაცემთა ბაზის მართვის სისტემის მეშვეობით. შედეგად მიღებული მონაცემთა ბაზები უნდა აკმაყოფილებდეს მომხმარებლის საჭიროებებს მონაცემთა გაზიარების, მოქნილობისა და ხელმისაწვდომობის სიმარტივის უზრუნველყოფით.

რელაციური მოდელი

მონაცემების რელაციური მოდელი პირველად 1970 წელს წარმოადგინა E. F. Codd– მა, ხოლო შემდეგ IBM–მა. გაშვებული იყო ორი ადრეული კვლევითი პროექტი რელაციონალური მოდელის განხორციელებადობის და პროტოტიპი სისტემების შემუშავების შესაძლებლობის დასამტკიცებლად. პირველი System R (რელაციური DBMS-ის პროტოტიპი [RDBMS]) შემუშავებით დასრულდა IBM–ის სან ხოსეს სამეცნიერო ლაბორატორიაში 1970 – იანი წლების ბოლოს. მეორე, ბერკლის კალიფორნიის უნივერსიტეტში, აკადემიურად ორიენტირებული RDBMS სისტემა Ingres-ის შემუშავებით დაგვირგვინდა. კომერციული RDBMS პროდუქციის გავრცელება დაიწყო დაახლოებით 1980 წელს. დღეს RDBMS–ები გახდა მონაცემთა ბაზის მართვის დომინანტური ტექნოლოგია და კომპიუტერებისთვის ასობით RDBMS პროდუქტი არსებობს, სმარტფონებიდან და პერსონალური კომპიუტერიდან მაგისტრალური ჩარჩოებით დასრულებული.

ძირითადი განმარტებები

მონაცემების რელაციური მოდელი მონაცემებს წარმოადგენს ცხრილების სახით. რელაციური მოდელი ემყარება მათემატიკურ თეორიას და, შესაბამისად, მას აქვს მყარი თეორიული საფუძველი. ამასთან, ჩვენ გვჭირდება მხოლოდ რამდენიმე მარტივი ცნება რელაციონალური მოდელის აღსაწერად. ამიტომ, მისი ადვილად გააზრება და გამოყენება შეიძლება მათთვისაც, ვინც არ იცნობს ძირითად თეორიას. მონაცემების რელაციური მოდელი შედგება შემდეგი სამი კომპონენტისგან:

1. **მონაცემთა სტრუქტურა** (***Data structure)*** - მონაცემები ორგანიზებულია ცხრილების სახით, სტრიქონებითა და სვეტებით;

2. **მონაცემთა მანიპულირება** (***Data manipulation)*** - რელაციაში შენახული მონაცემების მანიპულირებისთვის გამოიყენება მძლავრი ოპერაციები (ჩვეულებრივ ხორციელდება SQL ენის გამოყენებით);
3. **მონაცემთა მთლიანობა** (***Data integrity)*** - მოდელი მოიცავს მექანიზმებს ბიზნესის წესების დასაზუსტებლად, რომლებიც ინარჩუნებს მონაცემთა მთლიანობას მათი მანიპულირებისას.

EMPLOYEE1

EmpID	Name	DeptName	Salary
100	Margaret Simpson	Marketing	48,000
140	Allen Beeton	Accounting	52,000
110	Chris Lucero	Info Systems	43,000
190	Lorenzo Davis	Finance	55,000
150	Susan Martin	Marketing	42,000

FIGURE 4-1 EMPLOYEE1 relation with sample data

მონაცემთა რელაციური სტრუქტურა

რელაცია - მონაცემთა ორგანზომილებიანი სახელდებული ცხრილია. თითოეული რელაცია (ან ცხრილი) შედგება სახელდებული სვეტებისა და განუზღვრელი რაოდენობის უსახელო სტრიქონებისაგან. ატრიბუტი, რომელიც შეესაბამება მის განმარტებას E-R მოდელის განმარტებით, არის რელაციის სახელდებული სვეტი. რელაციის თითოეული სტრიქონი შეესაბამება ჩანაწერს, რომელიც შეიცავს მონაცემთა (ატრიბუტის) მნიშვნელობებს ერთი არსისათვის. დიაგრამა 4-1 გვიჩვენებს რელაციის მაგალითს, სახელწოდებით EMPLOYEE1. ეს რელაცია შეიცავს შემდეგ ატრიბუტებს, რომლებიც აღწერს თანამშრომლებს: EmpID, Name, DeptName, Salary. ცხრილის ხუთი სტრიქონი შეესაბამება ხუთ თანამშრომელს. მნიშვნელოვანია გვესმოდეს, რომ ნახაზი 4-1-ში მოცემული მონაცემების მიზანია EMPLOYEE1 რელაციის სტრუქტურის ილუსტრაცია; ისინი თავად რელაციის ნაწილები არ არიან. მაშინაც კი, თუ ფიგურას მონაცემების სხვა სტრიქონებს დავუმატებთ ან არსებულ სტრიქონებში რაიმე მონაცემს შევცვლით, ეს მაინც იგივეა EMPLOYEE1 რელაციაა. არც სტრიქონის წაშლა ცვლის რელაციას. სინამდვილეში, ჩვენ შეგვიძლია წავშალოთ ნახაზი 4-1-ში ნაჩვენები ყველა სტრიქონი და EMPLOYEE1 რელაცია მაინც იარსებებს. სხვა სიტყვებით რომ ვთქვათ, სურათი 4-1 EMPLOYEE1 რელაციის მაგალითია.

ჩვენ შეგვიძლია გამოვხატოთ რელაციის სტრუქტურა მოკლე ჩანაწერების გამოყენებით, რომელშიც რელაციის სახელს მოჰყვება (ფრჩხილებში) ატრიბუტების სახელები ამ რელაციაში. EMPLOYEE1– სთვის გვექნებოდა:

```
EMPLOYEE1(EmpID, Name, DeptName, Salary)
```

რელაციის გასაღებები

ჩვენ უნდა შეგვეძლოს მონაცემთა სტრიქონის შენახვა და გამოთხოვა რელაციაში, ამ სტრიქონში შენახული მონაცემთა მნიშვნელობების საფუძველზე. ამ მიზნის მისაღწევად, ყველა რელაციას უნდა ჰქონდეს **პირველადი გასაღები.** პირველადი გასაღები არის ატრიბუტი ან ატრიბუტების კომბინაცია, რომელიც ცალსახად განსაზღვრავს თითოეულ სტრიქონს რელაციაში. პირველადი გასაღები ატრიბუტის სახელის (ების) აღსანიშნად გამოიყენება ხაზგასმა. მაგალითად, EMPLOYEE1 რელაციის პირველადი გასაღები არის EmpID (ატრიბუტი ხაზგასმულია ნახაზზე 4-1.) მოკლე აღნიშვნით, ამ რელაციას შემდეგნაირად გამოვხატავთ:

EMPLOYEE1(EmpID, Name, DeptName, Salary)

პირველადი გასაღების ცნება უკავშირდება ადრე განსაზღვრულ იდენტიფიკატორის ტერმინს. ატრიბუტი ან ატრიბუტების კრებული, რომელიც მითითებულია როგორც არსის იდენტიფიკატორი E-R დიაგრამაზე, შეიძლება იყოს იგივე ატრიბუტები, რომლებიც წარმოადგენს ამ არსის წარმოდგენის რელაციის პირველად გასაღებს. არსებობს გამონაკლისები: მაგალითად, ასოციაციურ არსებს არ უნდა ჰქონდეთ იდენტიფიკატორი, ხოლო სუსტი არსის (ნაწილობრივი) იდენტიფიკატორი მხოლოდ შესაბამისი რელაციის ძირითადი გასაღების მხოლოდ ნაწილს წარმოადგენს. გარდა ამისა, შეიძლება არსებობდეს არსის რამდენიმე ატრიბუტი, რომელიც შეიძლება გახდეს ასოცირებული რელაციია პირველადი გასაღები. ყველა ამ სიტუაციას განვიხილავთ მოგვიანებით.

კომპოზიტური გასაღები არის პირველადი გასაღები, რომელიც შედგება ერთზე მეტი ატრიბუტისგან. მაგალითად, DEPENDENT რელაციის პირველადი გასაღები, სავარაუდოდ, შედგებოდა EmpID და DependentName კომბინაციით.

ხშირად გვჭირდება წარმოვადგინოთ დამოკიდებულება (კავშირი) ორ ცხრილსა თუ რელაციას შორის. ეს მიიღწევა გარე გასაღებების გამოყენებით. **გარე გასაღები** არის ატრიბუტი (შესაძლოა კომპოზიტური) რელაციაში, რომელიც სხვა რელაციის პირველადი გასაღებია. მაგალითად, განვიხილოთ დამოკიდებულება (კავშირი) EMPLOYEE1 და DEPARTMENT:

EMPLOYEE1(EmpID, Name, DeptName, Salary)
DEPARTMENT(DeptName, Location, Fax)

ატრიბუტი DeptName არის გარე გასაღები EMPLOYEE1– ში. ის მომხმარებელს საშუალებას აძლევს დააკავშიროს ნებისმიერი თანამშრომელი იმ განყოფილებასთან, რომელშიც ის დასაქმებულია. ზოგიერთი იმ ფაქტს, რომ ატრიბუტი გარე გასაღებია მიუთითებს წყვეტილი ხაზის გამოყენებით, როგორიცაა:

EMPLOYEE1(EmpID, Name, DeptName, Salary)

რელაციის თვისებები

ჩვენ განვსაზღვრეთ რელაციები, როგორც მონაცემთა ორგანზომილებიანი ცხრილი. ამასთან, ყველა ცხრილი არ არის რელაცია. რელაციას აქვთ რამდენიმე თვისება, რაც მათ განასხვავებს არარაელაციური ცხრილებისგან. ეს თვისებებია;

1. მონაცემთა ბაზაში თითოეულ რელაციას (ან ცხრილს) აქვს უნიკალური სახელი.
2. თითოეული სტრიქონისა და სვეტის გადაკვეთაზე მნიშვნელობა არის ატომური (ან ერთჯერადი). ცხრილის კონკრეტულ სტრიქონზე თითოეულ ატრიბუტთან ასოცირდება მხოლოდ ერთი მნიშვნელობა; დაუშვებელია მრავალმნიშვნელოვანი ატრიბუტების არსებობს;
3. თითოეული სტრიქონი უნიკალურია; რელაციაში არც ერთი სტრიქონი არ შეიძლება იყოს იდენტური;
4. ცხრილის თითოეულ ატრიბუტს (ან სვეტს) აქვს უნიკალური სახელი;
5. სვეტების თანმიმდევრობა (მარცხნიდან მარჯვნივ) უმნიშვნელოა. რელაციაში სვეტების თანმიმდევრობა შეიძლება შეიცვალოს რელაციის მნიშვნელობის ცვლილების ან გამოყენების გარეშე.
6. სტრიქონების თანმიმდევრობა (ზემოდან ქვედა) უმნიშვნელოა. როგორც სვეტების შემთხვევაში, რელაციის სტრიქონების რიგი შეიძლება შეიცვალოს ან შეინახოს ნებისმიერი თანმიმდევრობით.

მრავალმნიშვნელოვანი ატრიბუტის გადატანა ცხრილიდან

რელაციის ჩამოთვლილი თვისებებიდან მეორე აცხადებს, რომ დაუშვებელია რელაციაში მრავალმნიშვნელოვანი ატრიბუტის არსებობა. ამრიგად, ცხრილი, რომელიც შეიცავს ერთ ან რამდენიმე მრავალმნიშვნელოვან ატრიბუტს, არ არის რელაცია. მაგალითად, დიაგრამა 4-2a აჩვენებს თანამშრომლის მონაცემებს EMPLOYEE1 რელაციიდან, რომელიც მოიცავს იმ კურსებს, რომლებიც ამ თანამშრომლებმა გაიარეს. იმის გამო, რომ მოცემულმა თანამშრომელმა შეიძლება ერთზე მეტი კურსი გაიარა, CourseTitle და DateCompleted მრავალმნიშვნელოვანი ატრიბუტებია. მაგალითად, EmpID 100–ის შესაბამისმა თანამშრომელმა გაიარა ორი კურსი, შესაბამისად, არსებობს ორი კურსი: CourseTitle (SPSS და Surveys) და DateCompleted (6/9/2015 და 10/7/2015), რომელიც ასოცირდება EmpID–ის ერთ მნიშვნელობასთან (100). თუ თანამშრომელს არ აქვს გავლილი რაიმე კურსები, CourseTitle და DateCompleted ატრიბუტის მნიშვნელობები ნულოვანია (მაგალითისთვის იხილეთ EmpID 190 თანამშრომელი).

ვაჩვენოთ, თუ როგორ უნდა აღმოიფხვრას მრავალმნიშვნელოვანი ატრიბუტები ნახაზზე 4-2 b, მოცემული მონაცემების შესაბამისი მნიშვნელობების შევსებით ნახაზი 4-2a – ში მოცემული მონაცემების შესაბამისი მნიშვნელობებით. შედეგად, 4-2 b სურათზე მოცემულ ცხრილს აქვს მხოლოდ ერთმნიშვნელოვანი ატრიბუტები და ახლა აკმაყოფილებს რელაციის

ატომურობის თვისებას. ამ რელაციას მივანიჭოთ სახელი EMPLOYEE2, რათა განვასხვაოთ იგი EMPLOYEE1– ისგან. ამასთან, როგორც ვხედავთ, ამ ახალ რელაციას აქვს არასასურველი თვისებები. ზოგიერთ მათგანს მოგვიანებით განვიხილავთ, მაგრამ ერთ-ერთი მათგანია ის, რომ პირველადი გასაღების სვეტი EmpID აღარ წარმოადგენს ცალსახად თითოეულ სტრიქონს.

FIGURE 4-2 **Eliminating multivalued attributes**

(a) Table with repeating groups

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
				Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
				C++	4/22/2015
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/16/2015
				Java	8/12/2015

(b) EMPLOYEE2 relation

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
110	Chris Lucero	Info Systems	43,000	C++	4/22/2015
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/2015
150	Susan Martin	Marketing	42,000	Java	8/12/2015

მონაცემთა ბაზის ნიმუში

რელაციური მონაცემთა ბაზა შეიძლება შედგებოდეს ნებისმიერი რაოდენობის რელაციებისაგან. მონაცემთა ბაზის სტრუქტურა აღიწერება სქემის გამოყენებით, რომელიც წარმოადგენს მონაცემთა ბაზის საერთო ლოგიკური სტრუქტურის აღწერას. სქემის გამოხატვის ორი საერთო მეთოდი არსებობს:

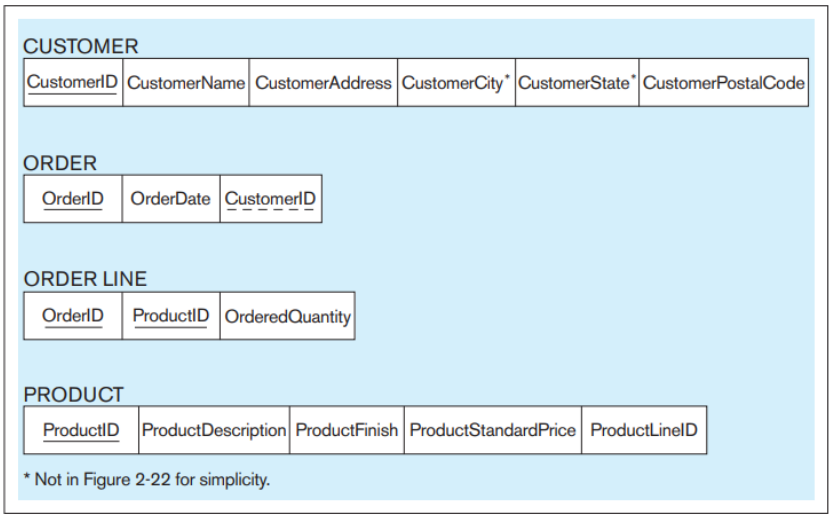
1. მოკლე ტექსტური განაცხადები, რომელშიც თითოეული რელაციაა დასახელებული და მისი ატრიბუტების სახელები ფრჩხილებში მოჰყვება;
2. გრაფიკული წარმოდგენა, რომელშიც თითოეული რელაცია წარმოდგენილია მართკუთხედით, რომელიც შეიცავს რელაციის ატრიბუტებს.

ტექსტურ განაცხადებს უპირატესობას ანიჭებენ სიმარტივის გამო. ამასთან, გრაფიკული წარმოდგენა გვთავაზობს დამოწმებითი მთლიანობის შეზღუდვის უკეთ გამოხატვას.

განვიხილოთ Pine Valley ავეჯის კომპანიის მაგალითი. ნახაზზე 4-3 ნაჩვენებია ოთხი რელაციისაგან შემდგარი სქემა. ეს რელაციებია CLIENT, ORDER, ORDER LINE და PRODUCT. ამ რელაციების გასაღები ატრიბუტები ხაზგასმულია და თითოეულ რელაციაში შედის სხვა მნიშვნელოვანი ატრიბუტები. ქვემოთ მოცემულია ამ რელაციების ტექსტური აღწერა, რათა შევძლოთ მათი შედარება:

CUSTOMER(CustomerID, CustomerName, CustomerAddress,
CustomerCity, CustomerState, CustomerPostalCode)
ORDER(OrderID, OrderDate, CustomerID)
ORDER LINE(OrderID, ProductID, OrderedQuantity)
PRODUCT(ProductID, ProductDescription, ProductFinish,
ProductStandardPrice, ProductLineID)

FIGURE 4-3 Schema for four relations (Pine Valley Furniture Company)



გაითვალისწინეთ, რომ ORDER LINE- ის პირველადი გასაღები არის კომპოზიტური გასაღები, რომელიც შედგება OrderID და ProductID ატრიბუტებისგან. ასევე, CustomerID არის გარე გასაღები ORDER რელაციაში. ეს საშუალებას აძლევს მომხმარებელს შეუკვეთოს შეკვეთა მომხმარებელთან, რომელმაც შეკვეთა წარადგინა. ORDER LINE– ს ორი გარე გასაღები აქვს: OrderID და ProductID. ეს გასაღებები საშუალებას აძლევს მომხმარებელს დაუკავშიროს შეკვეთის თითოეული სტრიქონი შესაბამის შეკვეთასთან და პროდუქტთან. იმ შემთხვევებში, როდესაც გარე გასაღები ასევე არის კომპოზიციური გასაღების ნაწილი (მაგალითად, ეს), ხშირად ატრიბუტის მხოლოდ ძირითად საკვანძო როლს აღნიშნავენ უწყვეტი ხაზგასმით.

ამ მონაცემთა ბაზის ეგზემპლარი ნაჩვენებია დიაგრამა 4-4-ზე. ამ ნახატზე მოცემულია ოთხი ცხრილი, მონაცემთა ნიმუშით. დავაკვირდეთ, როგორ გვაძლევს გარე გასაღები სხვადასხვა ცხრილების კავშირს. რელაციური სქემის მაგალითის შექმნა მონაცემთა ნიმუშთან ერთად სასარგებლოა ოთხი მიზეზის გამო:

1. მონაცემთა ნიმუში საშუალებას გვაძლევს შევამოწმოთ ჩვენი დაშვებები დიზაინთან დაკავშირებით;
2. მონაცემთა ნიმუში გვთავაზობს ჩვენი დიზაინის სიზუსტის შემოწმების მოსახერხებელ გზას;
3. მონაცემთა ნიმუში ხელს უწყობს მომხმარებლებთან კომუნიკაციის გაუმჯობესებას ჩვენი დიზაინის განხილვისას;
4. ნიმუშის მონაცემები შეიძლება გამოყენებულ იქნას პროტოტიპის პროგრამების შესაქმნელად და მოთხოვნების შესამოწმებლად.

მთლიანობის შეზღუდვები

მონაცემთა რელაციური მოდელი მოიცავს რამდენიმე სახის შეზღუდვას, ან მისაღები მნიშვნელობებისა და მოქმედებების შემზღუდველ წესებს, რომელთა მიზანია მონაცემთა ბაზაში მონაცემთა სიზუსტისა და მთლიანობის შენარჩუნება. მთლიანობის შეზღუდვის ძირითადი ტიპები არის დომენის შეზღუდვები, არსის მთლიანობა და დამოწმებითი მთლიანობა.

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

დომენის შეზღუდვები

ყველა მნიშვნელობა, რომელიც გამოჩნდება რელაციის სვეტში, უნდა იყოს ერთი და იგივე დომენიდან. დომენი არის მნიშვნელობების ერთობლიობა, რომელიც შეიძლება მიენიჭოს ატრიბუტს. დომენის განმარტება ჩვეულებრივ შედგება შემდეგი კომპონენტებისგან: დომენის სახელი, მნიშვნელობა, მონაცემთა ტიპი, ზომა (ან სიგრძე) და დასაშვები მნიშვნელობები ან დასაშვები დიაპაზონი (ასეთის არსებობის შემთხვევაში). ცხრილი 4-1 გვიჩვენებს დომენის განმარტებებს იმ დომენებისთვის, რომლებიც ასოცირდება ატრიბუტებთან 4-3 და 4-4 ნახაზებში.

არსის მთლიანობა

არსის მთლიანობის წესი შექმნილია იმის უზრუნველსაყოფად, რომ ყველა რელაციას ჰქონდეს პირველადი გასაღები და რომ ამ პირველადი გასაღების ყველა მონაცემების მნიშვნელობები იყოს ქმედითი. კერძოდ, იძლევა გარანტიას, რომ პირველადი გასაღების ყველა ატრიბუტი არ არის ნულოვანი.

ზოგიერთ შემთხვევაში, კონკრეტულ ატრიბუტს არ შეიძლება მიენიჭოს მონაცემთა მნიშვნელობა. არსებობს ორი სიტუაცია, რომლის დროსაც ეს შეიძლება მოხდეს: ან არ არსებობს მონაცემთა შესაბამისი მნიშვნელობა, ან მნიშვნელობების მინიჭებისას მნიშვნელობები არ არის ცნობილი. დავუშვათ, მაგალითად, რომ ვავსებთ დასაქმების ფორმას, რომელსაც აქვს ფაქსის ნომერი. თუ ფაქსის ნომერი არ გაქვთ, ამ ადგილს ცარიელს დავტოვებთ, რადგან ის არ გვეხება. ან ვთქვათ მოგვეთხოვება შეავსოთ წინა დამსაქმებლის ტელეფონის ნომერი. თუ ნომერი არ გვახსოვს, შეიძლება ცარიელი დავტოვოთ, რადგან ეს ინფორმაცია არ არის ცნობილი.

რელაციური მონაცემების მოდელი საშუალებას გვაძლევს ატრიბუტს მივუთითოთ ნულოვანი მნიშვნელობა უბრალოდ აღწერილ სიტუაციებში. Null არის მნიშვნელობა, რომელიც შეიძლება მიენიჭოს ატრიბუტს, როდესაც სხვა მნიშვნელობა არ გამოიყენება ან მოქმედი მნიშვნელობა უცნობია. სინამდვილეში, null არ არის მნიშვნელობა, არამედ ის მიუთითებს მნიშვნელობის არარსებობაზე. მაგალითად, ეს არ არის იგივე, რაც რიცხვითი ნულოვანი ან პრობელების სტრიქონი. ნულოვნების ჩართვა რელაციურ მოდელში გარკვეულწილად სადავოა, რადგან მას ზოგჯერ ანომალურ შედეგებამდე მივყავართ. ამასთან, რელაციური მოდელის გამომგონებელი კოდი მხარს უჭერს null-ების გამოყენებას გამოტოვებული მნიშვნელობებისთვის.

ყველა თანახმაა, რომ პირველადი გასაღების მნიშვნელობების გადახრა დაუშვებელია. ამრიგად, *არსის მთლიანობის წესი* აცხადებს შემდეგს: პირველადი გასაღების არცერთი ატრიბუტი (ან პირველადი გასაღების ატრიბუტის კომპონენტი) არ შეიძლება იყოს ნულოვანი.

დამოწმებითი მთლიანობა

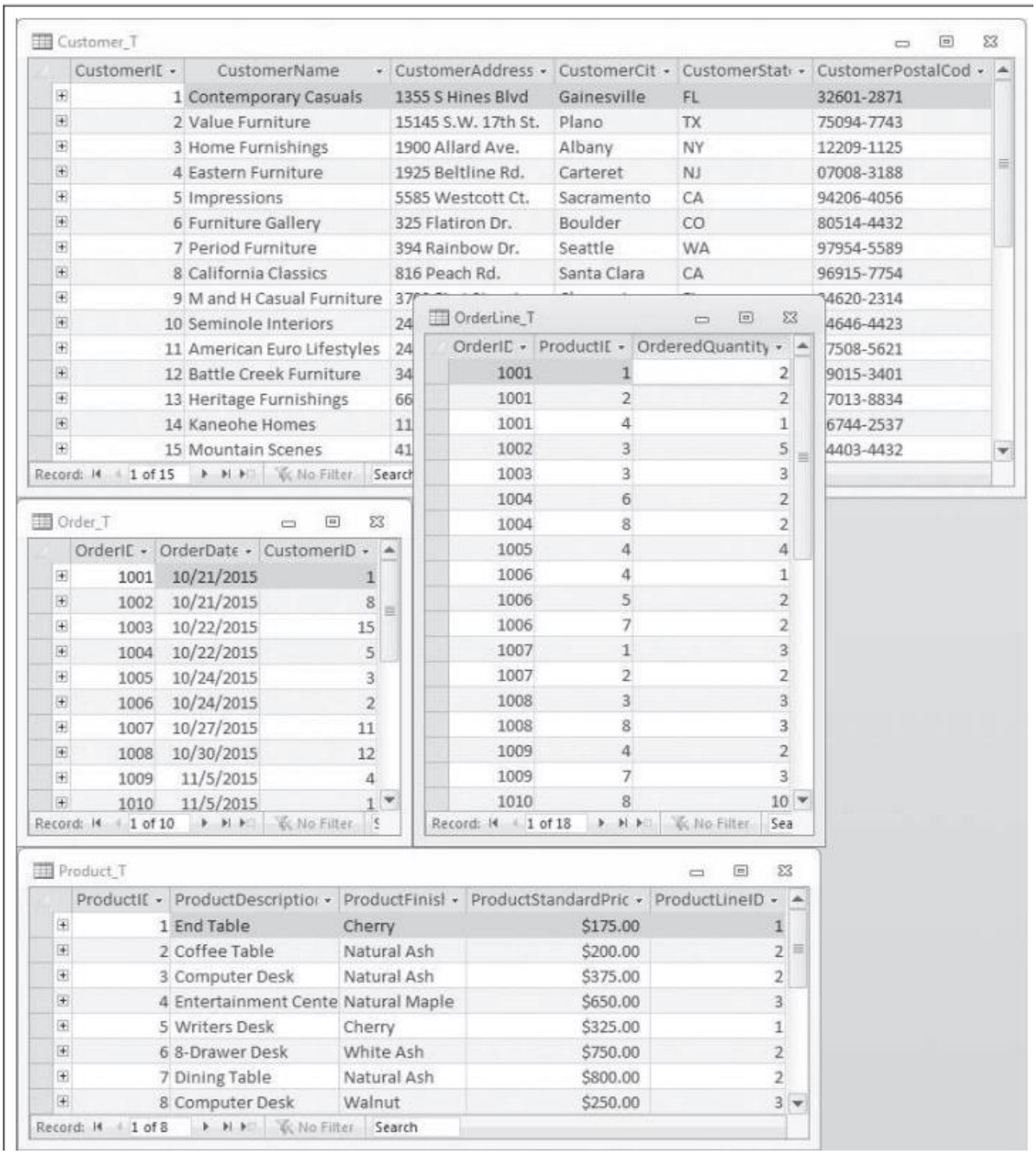
რელაციური მონაცემების მოდელში ცხრილებს შორის კავშირი განისაზღვრება გარე გასაღებების გამოყენებით. მაგალითად, დიაგრამა 4-4-ში, CUSTOMER და ORDER ცხრილებს შორის კავშირი განისაზღვრება CustomerID ატრიბუტის, როგორც გარე გასაღების ORDER- ში ჩასმით. ეს გულისხმობს იმას, რომ სანამ ORDER ცხრილში არ ჩავსვამთ ახალ სტრიქონს, ამ შეკვეთის მომხმარებელი უკვე უნდა არსებობდეს CUSTOMER ცხრილში. თუ შვისწავლით ORDER ცხრილის სტრიქონებს 4-4 ნახაზზე, ვნახავთ, რომ მომხმარებლის ყველა ნომერი შეკვეთისთვის უკვე ჩანს CUSTOMER ცხრილში.

დამოწმებითი მთლიანობის შეზღუდვა არის წესი, რომელიც ინარჩუნებს შეთანადება ორი რელაციის სტრიქონებში. წესში ნათქვამია, რომ თუ ერთ რელაციაში არის გარე გასაღები, ან თითოეული კარე გასაღების მნიშვნელობა უნდა ემთხვეოდეს პირველადი გასაღების

მნიშვნელობას სხვა რელაციაში, ან გარე გასაღების მნიშვნელობა უნდა იყოს ნულოვანი (მაგ. ნახ. 4-4-ზე მოცემული ცხრილები).

რელაციური სქემის გრაფიკული ვერსია გთავაზობთ მარტივ ტექნიკას კავშირების იდენტიფიკაციისთვის, სადაც დამოწმებითი მთლიანობა უნდა განხორციელდეს. დიაგრამა 4-5 გვიჩვენებს სქემა 4-3-ში მოცემული რელაციების. ისარი დახატულია ყოველი გარე გასაღებიდან შესაბამის პირველად გასაღებისკენ. სქემაში მოცემული თითოეული ისრისთვის უნდა განისაზღვროს დამოწმებითი მთლიანობის შეზღუდვა. გვახსოვდეს, რომ OrderID და ProductID ORDER LINE– ში არის როგორც გარე გასაღებები, ასევე კომპოზიტური ძირითადი გასაღების კომპონენტები.

FIGURE 4-4 **Instance of a relational schema (Pine Valley Furniture Company)**



Customer_T

CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
9	M and H Casual Furniture	37			4620-2314
10	Seminole Interiors	24			4646-4423
11	American Euro Lifestyles	24			7508-5621
12	Battle Creek Furniture	34			9015-3401
13	Heritage Furnishings	66			7013-8834
14	Kaneohe Homes	11			6744-2537
15	Mountain Scenes	41			4403-4432

Record: 1 of 15 No Filter Search

OrderLine_T

OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Record: 1 of 18 No Filter Sea

Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2015	1
1002	10/21/2015	8
1003	10/22/2015	15
1004	10/22/2015	5
1005	10/24/2015	3
1006	10/24/2015	2
1007	10/27/2015	11
1008	10/30/2015	12
1009	11/5/2015	4
1010	11/5/2015	1

Record: 1 of 10 No Filter

Product_T

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID
1	End Table	Cherry	$175.00	1
2	Coffee Table	Natural Ash	$200.00	2
3	Computer Desk	Natural Ash	$375.00	2
4	Entertainment Cente	Natural Maple	$650.00	3
5	Writers Desk	Cherry	$325.00	1
6	8-Drawer Desk	White Ash	$750.00	2
7	Dining Table	Natural Ash	$800.00	2
8	Computer Desk	Walnut	$250.00	3

Record: 1 of 8 No Filter Search

დამოწმებითი მთლიანობის უზრუნველყოფისასა უნდა შეგვეძლოს იმის გარკვევა ნებადართულია თუ არა გარე გასაღების მნიშნელობა იყოს NULl . თუ თითოეულ შეკვეთას უნდა ჰყავდეს მომხმარებელი (სავალდებულო ურთიერთობა), მაშინ გარე გასაღები CustomerID არ შეიძლება იყოს ნულოვანი ORDER რელაციაში. თუ კავშირი არასავალდებულოა, მაშინ გარე გასარები შეიძლება იყოს ნულოვანი. შესაძლებელია თუ არა

გარე გასაღების null მითითება, უნდა მოხდეს მონაცემთა ბაზის განსაზღვრისას, გარე გასაღების ატრიბუტის თვისების მითითებისას.

სინამდვილეში, შეიძლება თუ არა გარე გასაღების ნულოვანი მნიშვნელობა, უფრო რთულია E-R დიაგრამაზე მოდელირებისას და განსაზღვრისას, ვიდრე აქამდე ვაჩვენეთ. მაგალითად, რა ემართება შეკვეთის მონაცემებს, თუ ავირჩევთ მომხმარებელს, რომელმაც შეკვეთა უკვე წარადგინა? შეიძლება გვინდოდეს გაყიდვების ნახვა, მაშინაც კი, თუ მომხმარებელზე აღარ ვიზრუნებთ. შესაძლებელია სამი არჩევანი:

1. წავშალოთ მასთან დაკავშირებული შეკვეთები (ე.წ. კასკადური წაშლა), ამ შემთხვევაში ჩვენ ვკარგავთ არა მხოლოდ მომხმარებელს, არამედ გაყიდვების მთლიან ისტორიას;
2. ავკრძალოთ მომხმარებლის წაშლა მანამ, სანამ ყველა დაკავშირებული შეკვეთა არ წაიშლება (უსაფრთხოების შემოწმება).
3. გარე გასაღებში მოვათავსოთ null მნიშვნელობა (გამონაკლისი, რათა შეკვეთის შექმნისას შეკვეთას ჰქონდეს CustomerID მნიშვნელობა, დაკავშირებული მომხმარებლის წაშლის შემთხვევაში, CustomerID შეიძლება გახდეს null).

რელაციური ცხრილების შექმნა

განვმარტოთ რელაციური ცხრილის შექმნა SQL მონაცემთა განსაზღვრის ენის **CREATE TABLE** განაცხადის გამოყენებით. პრაქტიკაში, ცხრილის ეს განმარტებები იქმნება მონაცემთა ბაზის შემუშავების პროცესში მოგვიანებით. თუმცა ჩვენ ამ პროცედურებს აქ ვაჩვენებთ პროცესის უწყვეტობისთვის და განსაკუთრებით იმის გასაცნობად, თუ როგორ ხდება SQL–ში მთლიანობის შეზღუდვების რეალიზაცია.

SQL ცხრილის განმარტებები ნაჩვენებია 4-6 ნახაზზე. რელაციურ სქემაში ნაჩვენები ოთხი რელაციიდან თითოეული ცხრილი იქმნება ცალ-ცალკე (სურათი 4-5). შემდეგ განისაზღვრება ცხრილის თითოეული ატრიბუტი. გავითვალისწინოთ, რომ მონაცემთა ტიპი და სიგრძე თითოეული ატრიბუტისთვის აღებულია დომენის განმარტებებიდან (ცხრილი 4-1). მაგ: ატრიბუტი CustomerName Customer_T ცხრილში განისაზღვრება, როგორც VARCHAR (ცვლადი სიმბოლო) მონაცემთა ტიპი, სიგრძით 25. NOT NULL მითითებით, თითოეული ატრიბუტი შეიძლება შეიზღუდოს ნულოვანი მნიშვნელობის მინიჭებით.

პირველადი გასაღები მითითებულია თითოეული ცხრილისთვის PRIMARY KEY პირობის გამოყენებით, თითოეული ცხრილის განმარტების ბოლოს. OrderLine_T ცხრილი ასახავს, თუ როგორ უნდა მიუთითოთ პირველადი გასაღები, როდესაც ეს გასაღები არის ატრიბუტი. ამ მაგალითში OrderLine_T- ის პირველადი გასაღები არის OrderID და ProductID კომბინაცია. ოთხი ცხრილის თითოეული პირველადი გასაღები ატრიბუტი იზღუდება NOT NULL-ით. ეს ახორციელებს ადრე აღწერილ არსის მთლიანობის შეზღუდვას. გავითვალისწინოთ, რომ NOT NULL შეზღუდვა ასევე შეიძლება გამოყენებულ იქნეს პირველადი გასაღებისაგან განსხვავებული ატრიბუტებსთვისაც.

დამოწმებითი მთლიანობის შეზღუდვები მარტივად განისაზღვრება მე-4-5 ნახაზზე ნაჩვენები გრაფიკული სქემის საფუძველზე. ისარი გამოდის თითოეული გარე გასაღებიდან და

მიუთითებს კავშირში არსებული რელაციის პირველად გასაღებზე. SQL ცხრილის განსაზღვრებაში, FOREIGN KEY REFERENCES განაცხადი შეესაბამება თითოეულ ამ ისარს. ამრიგად, ცხრილისთვის Order_T, გარე გასაღები CustomerID მიუთითებს Customer_T- ის პირველად გასაღებაზე, რომელსაც ასევე უწოდებენ CustomerID. მიუხედავად იმისა, რომ ამ შემთხვევაში გარე გასაღები და პირველადი გასაღები ერთი და იგივე სახელისაა, ეს არა სავალდებულო. მაგალითად, გარე გასაღების ატრიბუტს CustomerID- ის ნაცვლად შეიძლება დაერქვას CustNo. ამასთან, გარე და პირველადი გასაღებები უნდა იყოს ერთი და იგივე დომენისგან (ანუ მათ უნდა ჰქონდეთ მონაცემთა ერთი და იგივე ტიპი).

OrderLine_T ცხრილი გვთავაზობს ცხრილის მაგალითს, რომელსაც ორი გარე გასაღები აქვს. ამ ცხრილის გარე გასაღებები მითითებულია Order_T და Product_T ცხრილებში, შესაბამისად. გვითვალისწინოთ, რომ ეს ორი გარე გასაღები ასევე OrderLine_T პირველადი გასაღების კომპონენტებია. ამ ტიპის სტრუქტურა ძალზე გავრცელებულია, „ბევრი-ბევრთან“ კავშირის განხორციელებისას.

FIGURE 4-6 SQL table definitions

```
CREATE TABLE Customer_T
        (CustomerID                 NUMBER(11,0)     NOT NULL,
        CustomerName                VARCHAR2(25)     NOT NULL,
        CustomerAddress             VARCHAR2(30),
        CustomerCity                VARCHAR2(20),
        CustomerState               CHAR(2),
        CustomerPostalCode          VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                    NUMBER(11,0)     NOT NULL,
        OrderDate                   DATE DEFAULT SYSDATE,
        CustomerID                  NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
        (ProductID                  NUMBER(11,0)     NOT NULL,
        ProductDescription          VARCHAR2(50),
        ProductFinish               VARCHAR2(20),
        ProductStandardPrice        DECIMAL(6,2),
        ProductLineID               NUMBER(11,0),
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
        (OrderID                    NUMBER(11,0)     NOT NULL,
        ProductID                   NUMBER(11,0)     NOT NULL,
        OrderedQuantity             NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```

კარგად სტრუქტურირებული რელაციები

ნორმალიზაციის შესახებ დისკუსიისთვის მოსამზადებლად საჭიროა პასუხი გაეცეს შემდეგ კითხვას: რას წარმოადგენს კარგად სტრუქტურირებული რელაცია? ინტუიციურად, კარგად სტრუქტურირებული რელაცია შეიცავს მინიმალურ სიჭარბეს და მომხმარებლებს საშუალებას აძლევს შეცდომისა და შეუსაბამობის გარეშე ჩასვან, შეცვალონ და წაშალონ ცხრილის სტრიქონები. EMPLOYEE1 (სურათი 4-1) ასეთი რელაციაა. ცხრილის თითოეული სტრიქონი შეიცავს მონაცემებს, რომლებიც აღწერს ერთ თანამშრომელს, ხოლო თანამშრომლის მონაცემებში ნებისმიერი ცვლილება (მაგალითად, ხელფასის ცვლილება) შემოიფარგლება ცხრილის ერთ რიგში განსახორციელები ცვლილებით. ამის საპირისპიროდ, EMPLOYEE2 (სურათი 4-2b) არ არის კარგად სტრუქტურირებული რელაცია. თუ შევისწავლით ცხრილში მოცემულ მონაცემთა ნიმუშს, შევამჩნევთ მნიშვნელოვან სიჭარბეს. მაგალითად, EmpID- ის, Name- ს, DeptName- ს და Salary-ს მნიშვნელობები გამოჩნდება ორ სხვადასხვა სტრიქონში 100, 110 და 150 თანამშრომლებისთვის. შესაბამისად, თუ EmpID- 100-ის შესაბამისი თანამშრომლის ხელფასი შეიცვალა, ეს ფაქტი ორ სტრიქონში უნდა ჩავწეროთ.

ცხრილში სიჭარბემ შეიძლება გამოიწვიოს შეცდომები ან შეუსაბამობები (ე.წ. ანომალიები), როდესაც მომხმარებელი ცდილობს განაახლოს ცხრილში მოცემული მონაცემები. ჩვენ, როგორც წესი, გვაწუხებს სამი სახის ანომალია:

1. *ჩასმის ანომალია* - დავუშვათ, რომ EMPLOYEE2– ს უნდა დავამატოთ ახალი თანამშრომელი. ამ რელაციის პირველადი გასაღები არის EmpID და CourseTitle კომბინაცია (როგორც ადრე აღვნიშნეთ). ამიტომ, ახალი სტრიქონის ჩასასმელად, მომხმარებელმა უნდა მიუთითოს როგორც EmpID-ის, ასევე CourseTitle-ის მნიშვნელობები, რადგან პირველადი გასარების მნიშვნელობები არ შეიძლება იყოს ნულოვანი ან არარსებული. ეს ანომალიაა, რადგან მომხმარებელს უნდა შეეძლოს შეიტანოს თანამშრომლის მონაცემები, კურსებზე მონაცემების მიწოდების გარეშე.
2. *წაშლის ანომალია* - დავუშვათ, რომ ცხრილიდან წაიშლება EmpID- 140-ის შესაბამისი პერსონალის მონაცემები. ეს გამოიწვევს ინფორმაციის დაკარგვას იმის შესახებ, რომ ამ თანამშრომელმა დაასრულა კურსი (Tax Acc) 12/8/2015. ფაქტობრივად, ეს კარგავს ინფორმაციას, რომ ამ კურსს ჰქონდა შეთავაზება, რომელიც დასრულდა ამ დღეს.
3. *მოდიფიკაციის ანომალია* - დავუშვათ, რომ EmpID- 100-ის შესაბამის თანამშრომელს გაეზარდა ხელფასი. ჩვენ უნდა დავაფიქსიროთ ეს ზრდა თითოეული ამ თანამშრომლის შესაბამის ყოველ სტრიქონში (ორი შემთხვევა ნახაზზე 4-2); წინააღმდეგ შემთხვევაში, მონაცემები არათანმიმდევრული იქნება.

ეს ანომალიები მიუთითებს, რომ EMPLOYEE2 არ არის კარგად სტრუქტურირებული რელაცია. ამ რელაცის პრობლემა ისაა, რომ იგი შეიცავს მონაცემებს ორი არსის შესახებ: EMPLOYEE და COURSE. ჩვენ გამოვიყენებთ ნორმალიზაციის თეორიას EMPLOYEE2-ის ორ რელაციად დაყოფისთვის. შედეგად მიღებული ერთ-ერთი რელაციაა EMPLOYEE1 (სურათი 4-1). მეორეს EMP COURSE- ს დავარქმევთ, რომელიც მოცემულია მონაცემების ნიმუშით 4-7 ნახაზზე. ამ რელაციის პირველადი გასაღები არის EmpID-ისა და CourseTitle-ის კომბინაცია და ამ ატრიბუტების სახელებს ხაზს ვუსვამთ დიაგრამა 4-7-ზე, ამ ფაქტის გასაზრდელად. დიაგრამა

4-7-ზე გამოსახული EMP COURSE არ შეიცავს ადრე აღწერილი ანომალიების ტიპებს და, შესაბამისად, კარგად არის სტრუქტურირებული.

FIGURE 4-7 EMP COURSE

EmpID	CourseTitle	DateCompleted
100	SPSS	6/19/2015
100	Surveys	10/7/2015
140	Tax Acc	12/8/2015
110	Visual Basic	1/12/2015
110	C++	4/22/2015
150	SPSS	6/19/2015
150	Java	8/12/2015

EER დიაგრამის გარდაქმნა რელაციურ სტრუქტურად

ლოგიკური დიზაინის დროს ჩვენ გარდაქმნით E-R (და EER) დიაგრამებს, რომლებიც შემუშავდა კონცეპტუალური დიზაინის დროს, რელაციური მონაცემთა ბაზის სქემებად. ამ პროცესის საწყის წყაროს წარმოადგენს „არსი-კავშირი“ (და გაუმჯობესებული E-R) დიაგრამები, ხოლო შედეგები წარმოადგენს რელაციურ სქემებს.

EER დიაგრამების რელაციად გარდაქმნა (ან ასახვა) შედარებით მარტივი პროცესია, კარგად განსაზღვრული წესებით. სინამდვილეში, მრავალ CASE ინსტრუმენტს შეუძლია ავტომატურად შეასრულოს გარდაქმნის მრავალი ბიჯი. ამასთან, მნიშვნელოვანია ამ პროცესში ბიჯების არსის გააზრება ოთხი მიზეზის გამო:

1. CASE ინსტრუმენტებს ხშირად არ შეუძლიათ მონაცემთა უფრო რთული კავშირების (ურთდამოკიდებულებების) მოდელირება, როგორიცაა სამეული კავშირები და სუპერტიპი/ქვეტიპი კავშირები, ამ სიტუაციებში შეიძლება დაგვჭირდს ბიჯების ხელით შესრულება;
2. ზოგჯერ არსებობს ლეგიტიმური ალტერნატივა, რომლისთვისაც უნდა ავირჩიოთ კონკრეტული გადაწყვეტა;
3. მზად უნდა ვიყოთ CASE ინსტრუმენტის საშუალებით მიღებული შედეგების ხარისხის შესამოწმებლად;
4. ტრანსფორმაციის პროცესის გაგება დაგვეხმარებათ იმის გაგებაში, თუ რატომ განსხვავდება კონცეპტუალური მონაცემების მოდელირება (რეალურ დომენზე მოდელირება) ლოგიკური მონაცემების მოდელირებისგან (ანუ წარმოადგენს მონაცემთა ერთეულებს დომენში ისე, რომ ეს შეიძლება განხორციელდეს DBMS– ით).

ჩვენ განვიხილეთ სამი სახის არსი:

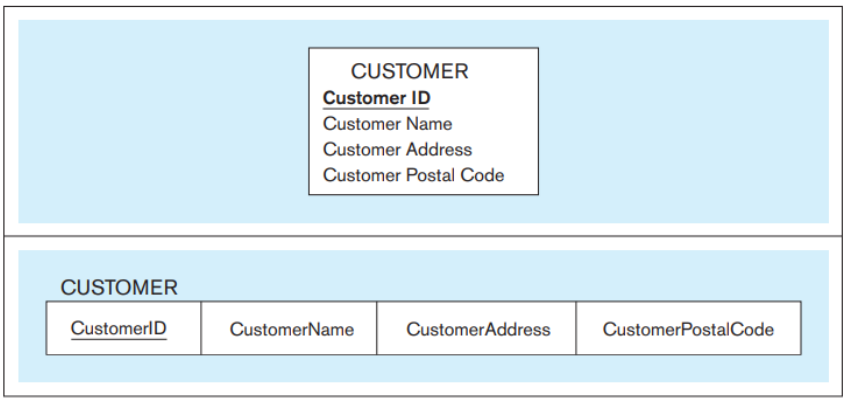
1. რეგულარული არსი (***Regular entities***) - არსები, რომლებიც დამოუკიდებელად არსებობენ და ზოგადად წარმოადგენენ რეალურ ობიექტებს, როგორიცაა პირები და პროდუქტები. არსების რეგულარული ტიპები წარმოდგენილია ერთმაგი ხაზით შემოფარგლული მართკუთხედებით.
2. სუსტი არსები (***Weak entities)*** - არსები, რომელთა არსებობა შეუძლებელია, მფლობელი (რეგულარული) არსის ტიპთან საიდენტიფიკაციო ურთიერთობის გარეშე. სუსტი არსის აღინიშნება ორმაგი ხაზით შემოფარგლული მართკუთხედით.
3. ასოციაციური არსები ***Associative entities*** (ასევე უწოდებენ გერუნდებს) - წარმოიქმნება სხვა არსების ტიპებს შორის „მრავალი-მრავალთან“ კავშირისას. ასოციაციური არსები წარმოდგენილია მომრგვალებული კუთხეებიანი მართკუთხედით.

ბიჯი 1: რეგულარული არსების ასახვა

თითოეული რეგულარული არსის ტიპი E-R სქემაში გარდაიქმნება რელაციად (ორგანზომილებიან ცხრილად, რომელსაც ხშირად „დამოკიდებულებასაც“ უწოდებენ). რელაციის სახელი, ზოგადად, იგივეა, რაც არსის ტიპის სახელი. არსის ტიპის თითოეული მარტივი ატრიბუტი ხდება რელაციის ატრიბუტი. არსის ტიპის იდენტიფიკატორი ხდება შესაბამისი რელაციის პირველადი გასაღები. აუცილებელია შეამოწმდეს ასეთი სახით წარმოქმნილი პირველადი გასაღები, რათა ის აკმაყოფილებდეს იდენტიფიკატორების თვისებებს.

დიაგრამა 4-8 ა გვიჩვენებს CUSTOMER არსის ტიპს Pine Valley ავეჯის კომპანიისთვის (იხ. სურათი 2-22) შესაბამისი CUSTOMER რელაცია გრაფიკული ფორმით არის ნაჩვენები ნახაზზე 4-8 ბ. ამ ფიგურაში და ამ მონაკვეთში მოცემულია მხოლოდ რამდენიმე ძირითადი ატრიბუტი თითოეული რელაციისათვის, რათა გავამარტივოთ ფიგურები.

FIGURE 4-8 Example of mapping a regular entity
(a) CUSTOMER entity type



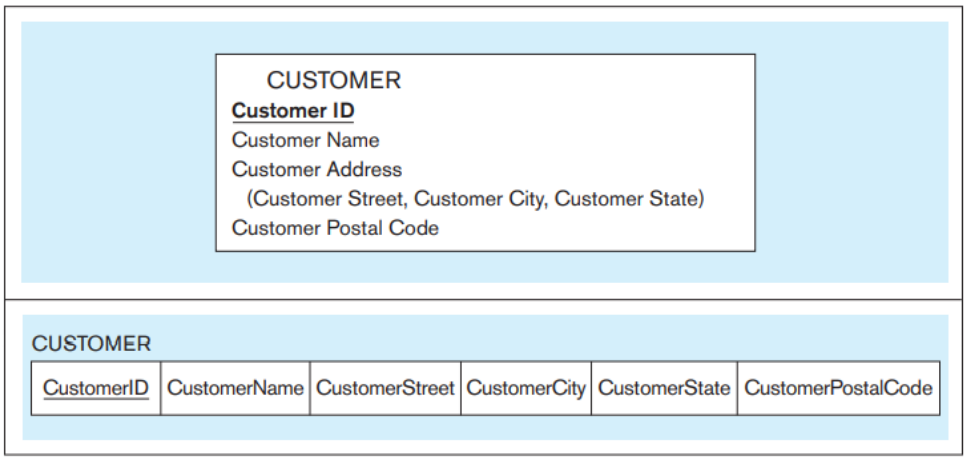
(b) CUSTOMER relation

კომპოზიტური ატრიბუტები

როდესაც ჩვეულებრივი არსის ტიპს აქვს კომპოზიტური ატრიბუტი, კომპოზიტური ატრიბუტის მხოლოდ მარტივი კომპონენტები შედის ახალ რელაციაში, როგორც მისი

ატრიბუტები. დიაგრამა 4-9 გვიჩვენებს 4-8 ნახაზზე მოყვანილი მაგალითის ვარიანტს, სადაც მომხმარებლის მისამართი წარმოდგენილია კომპოზიციური ატრიბუტი კომპონენტებით **Street, City** და **State** (იხ. სურათი 4-9a). ეს ობიექტი მონიშნულია CUSTOMER რელაციაში, რომელიც შეიცავს მარტივი მისამართის ატრიბუტებს, როგორც ეს ნაჩვენებია ნახაზზე 4-9b. მიუხედავად იმისა, რომ მომხმარებლის სახელი არის გამოსახული, როგორც მარტივი ატრიბუტი დიაგრამა 4-9 ა-ში, იგი შეიძლებოდა ყოფილიყო მოდელირებული (და, პრაქტიკულად, იქნებოდა)) როგორც კომპოზიციური ატრიბუტი კომპონენტებით: **Last Name, First Name** და **Middle Initial** (გვარი, სახელი და შუა ასო) CUSTOMER რელაციის დაპროექტებისას (სურათი 4-9 ბ), შეგვიძლიათ ავირჩიოთ ამ მარტივი ატრიბუტების გამოყენება CustomerName-ის ნაცვლად. კომპოზიციურ ატრიბუტებთან შედარებით, მარტივი ატრიბუტები აუმჯობესებს მონაცემთა ხელმისაწვდომობას და ხელს უწყობს მონაცემთა ხარისხის შენარჩუნებას. მაგალითად, მონაცემთა საანგარიშო და სხვა შედეგობრივი მიზნებისათვის მრავალად უფრო ადვილია, თუ CustomerName წარმოდგენილი იქნება მისი კომპონენტებით (გვარი, სახელი და შუა ასო ცალკე). ამ გზით, მონაცემების შესახებ ინფორმაციის წარმოდგენის ნებისმიერ პროცესს შეუძლია შექმნას მისთვის საჭირო ფორმატი.

FIGURE 4-9 Example of mapping a composite attribute
(a) CUSTOMER entity type with composite attribute



(b) CUSTOMER relation with address detail

.

მრავალმნიშვნელოვანი ატრიბუტი

როდესაც ჩვეულებრივი არისი ტიპი შეიცავს მრავალმნიშვნელოვან ატრიბუტს, იქმნება ორი ახალი რელაცია (ნაცვლად ერთის). პირველი რელაცია შეიცავს არსის ტიპის ყველა ატრიბუტს, გარდა მრავალმნიშვნელოვანი ატრიბუტისა. მეორე რელაცია შეიცავს ორ ატრიბუტს, რომლებიც ქმნიან მეორე რელაციის პირველად გასაღებს. ამ ატრიბუტებიდან პირველი არის პირველადი გასაღები პირველი რელაციიდან, რომელიც ხდება მეორე რელაციის გარე გასაღები. მეორე არის მრავალმნიშვნელოვანაი ატრიბუტი. მეორე რელაციის სახელი უნდა ასახავდეს მრავალმნიშვნელოვანი ატრიბუტის მნიშვნელობას.

რელაცია EMPLOYEE SKILL არ შეიცავს არაგასაღებ ატრიბუტებს (ასევე უწოდებენ დესკრიპტორებს). თითოეულ სტრიქონში უბრალოდ ფიქსირდება ის ფაქტი, რომ კონკრეტული თანამშრომელი ფლობს კონკრეტულ უნარს. ეს საშუალებას გვაძლევთ

მომხმარებლებს შევთავაზოთ, ამ რელაციაზე ახალი ატრიბუტების დამატების შესაძლებლობა. მაგალითად, ატრიბუტები YearsExperience და/ან CertificationDate შეიძლება იყოს შესაბამისი ახალი მნიშვნელობები, ამ რელაციაზე დამატებისთვისნ. თუ თვითონ SKILL საჭიროებს დამატებით ატრიბუტებს, შეგვიძლია შევქმნათ ცალკე SKILL კავშირი. ამ შემთხვევაში, EMPLOYEE SKILL ხდება ასოცირებული არსი EMPLOYEE– სა და SKILL– ს შორის.

მრავალმნიშვნელოვანი ატრიბუტის ასახვის მაგალითი მოყვანილია ნახ.4-10-ზე.

თუ არსის ტიპი შეიცავს ბევრ მრავალმნიშვნელოვან ატრიბუტს, თითოეული მათგანი გადაიქცევა ცალკეულ რელაციად.

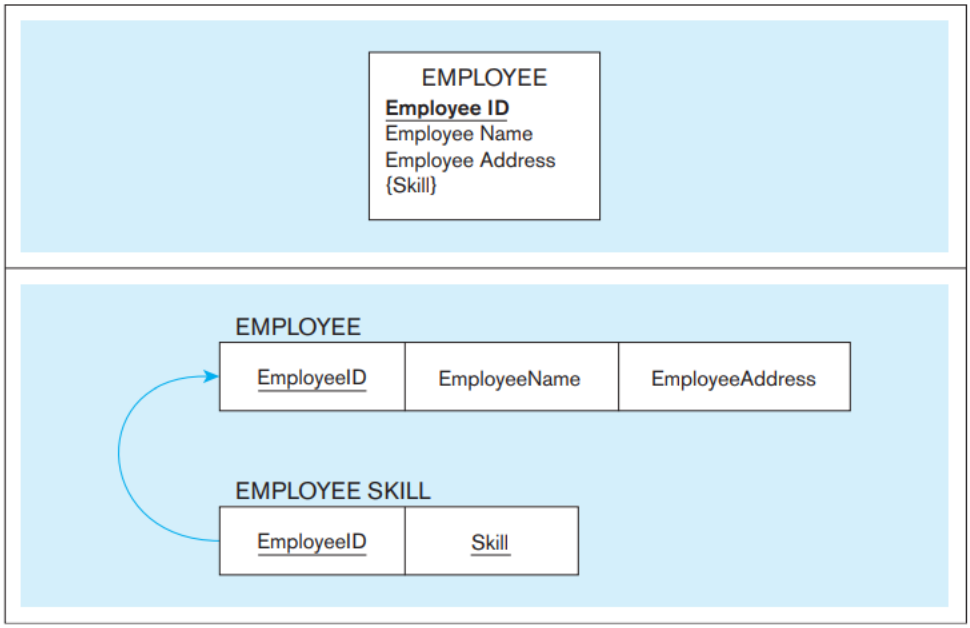


FIGURE 4-10 Example of mapping an entity with a multivalued attribute
(a) EMPLOYEE entity type with multivalued attribute

(b) EMPLOYEE and EMPLOYEE SKILL relations

ბიჯი 2: სუსტი არსის ასახვა

როგორც აღვნიშნეთ, სუსტი არსის ტიპს არ შეუძლია დამოუკიდებელი არსებობა, არსებობს მხოლოდ იდენტიფიცირების გზით სხვა არსის ტიპთან, რომელსაც ეწოდება მფლობელი. სუსტი არსის ტიპს არ აქვს სრული იდენტიფიკატორი, მაგრამ მას უნდა ჰქონდეს ატრიბუტი, სახელწოდებით ნაწილობრივი იდენტიფიკატორი, რომელიც საშუალებას იძლევა განასხვაოს სუსტი არსის სხვადასხვა შემთხვევები თითოეული მფლობელი არსის ეგზემპლარისთვის.

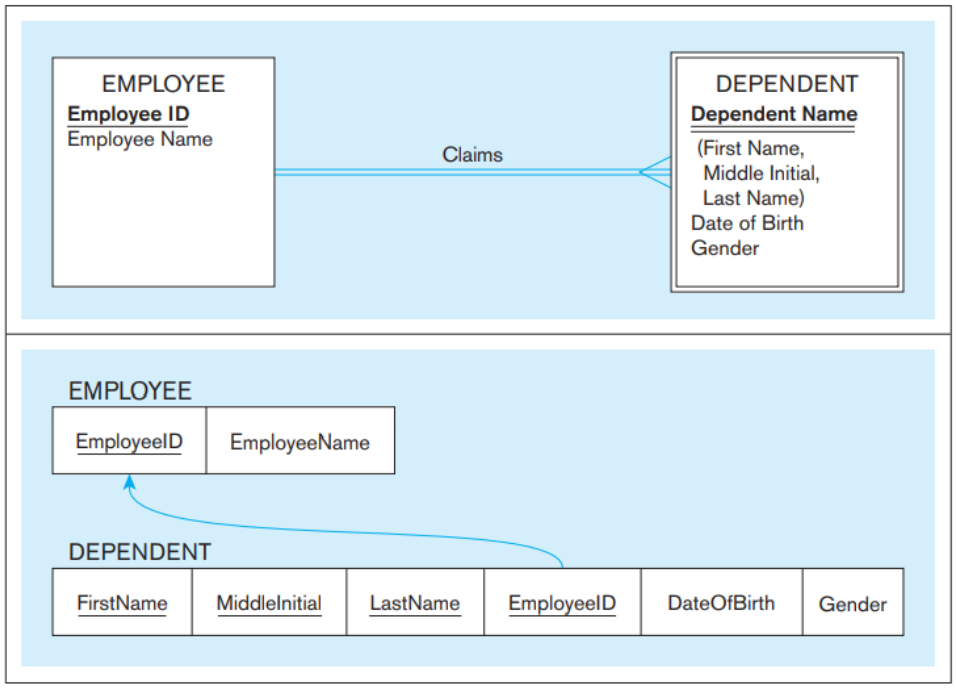
სუსტი არსის ასახვისათვის პირველ რიგში უნდა განხორციელდეს იგივე პროცედურები რაც არწერილი იყო პირველ ბიჯში - თითოეული სუსტი არსის ტიპისთვის შევქმნათ ახალი რელაცია და შეიტანეთ ყველა მარტივი ატრიბუტი (ან კომპოზიციური ატრიბუტების მარტივი

კომპონენტები), როგორც ამ რელაციის ატრიბუტები. შემდეგ ამ ახალ რელაციაში შევიტანოთ მაიდენტიფიცირებადი რელაციის პირველადი გასაღები, როგორც გარე გასაღები ატრიბუტი. ახალი რელაციის პირველადი გასაღები არის მაიდენტიფიცირებადი რელაციის პირველადი გასაღებისა და სუსტი არსის ტიპის ნაწილობრივი იდენტიფიკატორის კომბინაცია.

ამ პროცესის მაგალითი ნაჩვენებია ნახატზე 4-11. დიაგრამა 4-11a აჩვენებს სუსტი არსის ტიპის DEPENDENT და მისი მაიდენტიფიცირებელი სუბიექტის არსის EMPLOYEE შეერთებას მაიდენტიფიცირებელი კავშირით Claims (იხილეთ სურათი 2-5). გავითვალისწინოთ, რომ ატრიბუტი Dependent Name, რომელიც ამ რელაციის ნაწილობრივი იდენტიფიკატორია, არის კომპოზიციური ატრიბუტი კომპონენტებით: First Name, Middle Initial, და Last Name. ამრიგად, ვთვლით, რომ მოცემული თანამშრომლისთვის ეს ელემენტები ცალსახად განსაზღვრავს დამოკიდებულ პირს.

დიაგრამა 4-11 ბ აჩვენებს ორ რელაციას, რომლებიც წარმოიქმნება ამ E-R დიაგრამის სეგმენტიდან. DEPENDENT- ის პირველადი გასაღები ოთხი ატრიბუტისგან შედგება: EmployeeID, FirstName, MiddleInitial და LastName. DateOfBirth და Gender არის არაგასარები ატრიბუტები. გარე გასაღების კავშირი მის პირვანდელ გასაღებთან ნახატზე მითითებულია ისრით.

FIGURE 4-11 Example of mapping a weak entity
(a) Weak entity DEPENDENT



(b) Relations resulting from weak entity

პრაქტიკაში, ხშირად გამოიყენება ალტერნატიული მიდგომა DEPENDENT პირველადი გასაღების გასამარტივებლად: შევქმნათ ახალი ატრიბუტი (ე.წ. DependentID), რომელიც გამოყენებული იქნება სუროგატი პირველადი გასაღებისთვის, დიაგრამა 4-11 ბ. ამ მიდგომით რელაცია DEPENDENT აქვს შემდეგი ატრიბუტები:

DEPENDENT(DependentID, EmployeeID, FirstName, MiddleInitial, LastName, DateOfBirth, Gender)

DependentID არის უბრალოდ რიგითი ნომერი, რომელიც ენიჭება თითოეულ დასაქმებულზე დამოკიდებულ პირს. გავითვალისწინოთ, რომ ეს გამოსავალი უზრუნველყოფს უნიკალურ იდენტიფიკაციას თითოეული დამოკიდებული ადამიანისთვის.

როდის იქმნება სუროგატული გასაღები

სუროგატულიი გასაღები ჩვეულებრივ იქმნება გასაღების სტრუქტურების გამარტივების მიზნით. სუროგატი გასაღები უნდა შეიქმნას, როდესაც რომელიმე შემდეგი პირობაა დაცული:

- არსებობს კომპოზიტიური პირველადი გასაღები (ისევე როგორც DEPENDENT რელაციის შემთხვევაში, რომელიც ნაჩვენებია ადრე ოთხკომპონენტიანი პირველადი გასაღებით).
- ბუნებრივი პირველადი გასაღები (ანუ გასაღები, რომელიც გამოიყენება ორგანიზაციაში და აღიარებულია კონცეპტუალური მონაცემების მოდელირებაში, როგორც იდენტიფიკატორი) არაეფექტურია. მაგალითად, შეიძლება იყოს ძალიან გრძელი და, შესაბამისად, „ძვირადღირებული“ მონაცემთა ბაზის პროგრამული უზრუნველყოფის დამუშავებისას, თუ იგი გამოიყენებული იქნება როგორც გარე გასაღები, სხვა ცხრილებზე წვდომისას.
- ბუნებრივი პირველად გასაღები ხელმეორედ გამოიყენება (ანუ, გასაღები ხელახლა გამოიყენება ან მეორდება პერიოდულად, ისე რომ, ის შეიძლება რეალურად არ იყოს უნიკალური დროთა განმავლობაში); ამ პირობის ზოგადი განაცხადია - როდესაც ბუნებრივი პირველადი გასაღები, ფაქტობრივად, ვერ იქნება გარანტირებული, რომ დროთა განმავლობაში შეინარჩუნებს უნიკალურობას (მაგალითად, შეიძლება არსებობდეს დუბლიკატები, მაგალითად სახელები ან სათაურები).

სუროგატი გასაღების შექმნისას, ბუნებრივი გასაღები ყოველთვის ინახება როგორც არაგასაღები მონაცემები იმავე რელაციაში, რადგან ბუნებრივ გასაღებს აქვს ორგანიზაციული მნიშვნელობა, რომელიც უნდა იყოს დაფიქსირებული მონაცემთა ბაზაში. სინამდვილეში, სუროგატი გასაღები მომხმარებლებისთვის არაფერს ნიშნავს, ამიტომ ისინი მომხმარებელს, როგორც წესი, არასოდეს უჩანს. ამის სანაცვლოდ, ბუნებრივი გასაღებები იდენტიფიკატორად გამოიყენება ძიებებში.

ბიჯი 3: ორობითი (ბინარული) კავშირის ასახვა

კავშირის ასახვი პროცედურა დამოკიდებულია როგორც კავშირის ხარისხზე (უნიარული, ორობითი ან სამეული) ასევე კავშირის კარდინალურობაზე.

ერთი-მრავალთან ბინარული კავშირის ასახვა

თითოეული ორობითი (ბინარული) 1: M კავშირისათვის, პირველ რიგში, იქმნება რელაცია კავშირში მონაწილე ორი არსის თითოეული ტიპისთვის, ბიჯი 1-ში აღწერილი პროცედურის გამოყენებით. შემდეგ, არსის კავშირის ერთი მხარის პირველადი გასაღები ატრიბუტი (ან ატრიბუტები) შეგვაქვს, როგორც გარე გასაღები რელაციაში, რომელიც კავშირის „მრავალის“ მხარეს წარმოადგენს (ამ წესის დამახსოვრებისათვის შეგვიძლია გამოვიყენოთ მნენონიკა: პირველადი გასაღები მიემართება „მრავალის“ მხარეს).

ამ მარტივი პროცესის საილუსტრაციოდ, გამოვიყენოთ კავშირი მომხმარებლებსა და შეკვეთებს შორის Pine Valley ავეჯის კომპანიისთვის (იხ. სურათი 2-22). ეს 1:M კავშირი ილუსტრირებულია ნახაზზე 4-12a. დიაგრამა 4-12b გვიჩვენებს ამ წესის გამოყენების შედეგს არსის ტიპების წარმოსაჩენად 1: M კავშირში. CUSTOMER– ის პირველადი გასაღები CustomerID (ერთი მხარე) შედის ORDER–ში, როგორც გარე გასაღებები. გარე გასაღების კავშირი მითითებულია ისრით. გავითვალისწინოთ, რომ არ არის საჭირო გარე გასაღების ატრიბუტის CustomerID დასახელება იყოს იდენტური კავშირში მონაწილე არსის ტიპის პირველადი გასარებისა. ამასთან, არსებითია, რომ მას ჰქონდეს იგივე დომენი, რაც პირველად გასაღებს, რომელსაც ის მიმართავს.

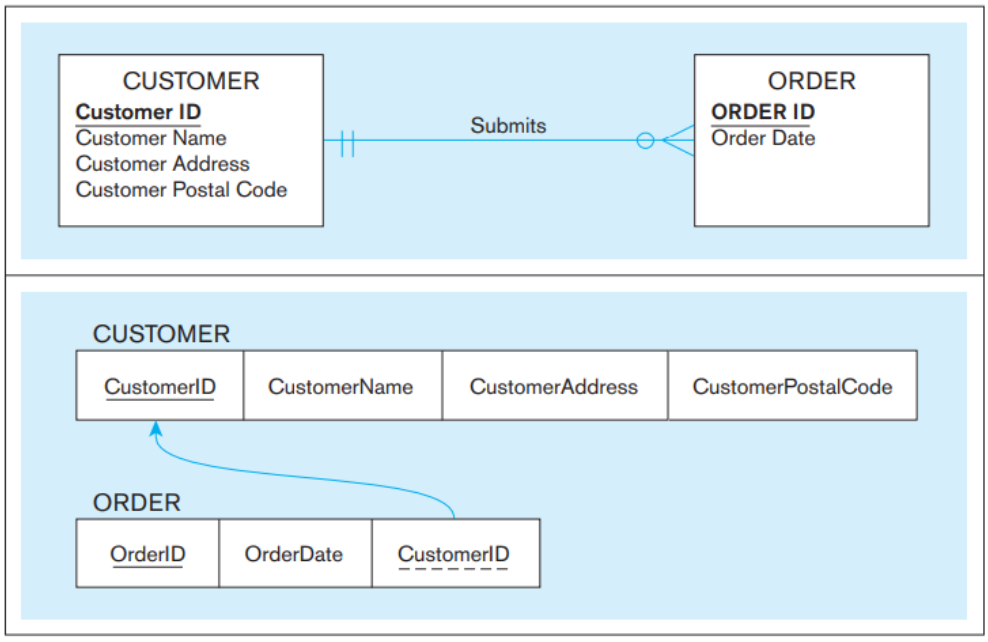


FIGURE 4-12 Example of mapping a 1:*M* relationship
(a) Relationship between CUSTOMER and ORDER entities

(b) CUSTOMER and ORDER relations with a foreign key in ORDER

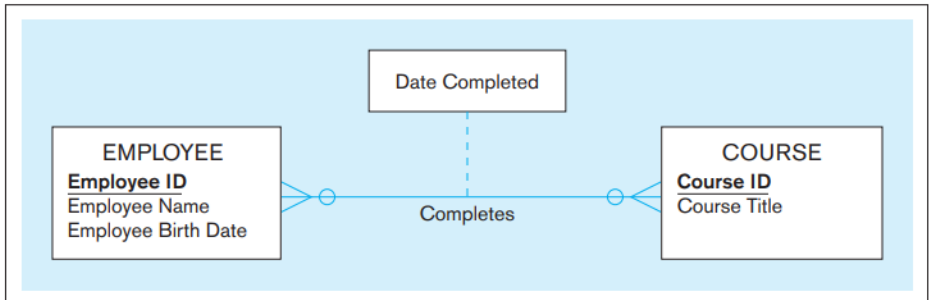
მრავალი-მრავალთან ბინარული კავშირის ასახვა

დავუშვათ, რომ არსებობს ბინარული მრავალი-მრავალთან (M:N) კავშირი ორ არსის ტიპებს შორის, A და B. ასეთი კავშირისთვის იქმნება ახალი რელაცია C. შევიტანოთ, როგორც გარე გასაღების ატრიბუტები C-ში თითოეული ორი არსის ტიპის პირველადი გასაღები, როგორც

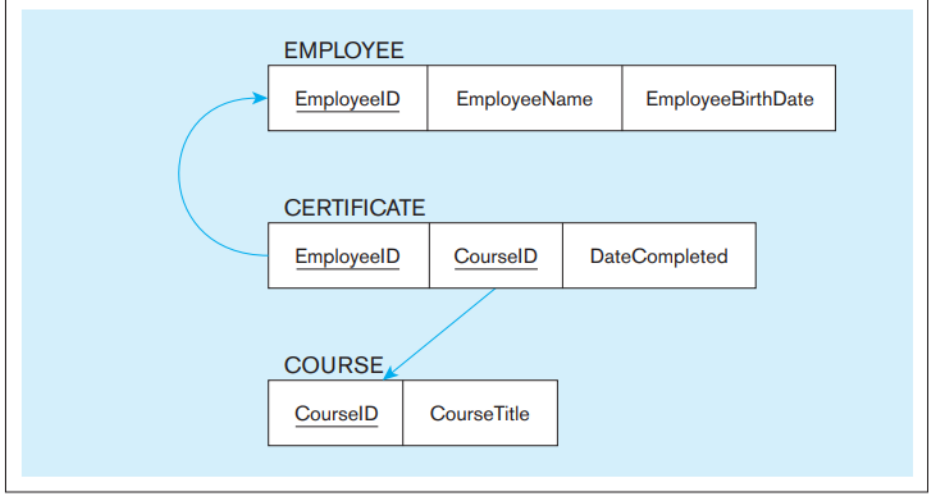
გარე გასაღები. ეს ატრიბუტები ერთად გახდება C-ს პირველადი გასაღები. ნებისმიერი არაგასაღები ატრიბუტი, რომლებიც ასოცირდება M:N კავშირში, შედის C რელაციაში.

დიაგრამა 4-13 გვიჩვენებს ამ წესის გამოყენების მაგალითს. დიაგრამა 4-13a გვიჩვენებს კავშირი Completes („დასრულებულია“) არსის ტიპებს EMPLOYEE და COURSE-ს შორის, სურათი 2-11a-დან. დიაგრამა 4-13 ბ აჩვენებს სამ რელაციას (EMPLOYEE, COURSE და CERTIFICATE), რომლებიც წარმოიქმნება არსის ტიპებისა და Complete კავშირისაგან. თუ Completes წარმოდგენილი იქნებოდა როგორც ასოციაციური არსი, როგორც ეს გაკეთებულია ნახაზზე 2-11b, მსგავსი შედეგი მოხდებოდა, (ამ საკითხს განვიხილავთ მოგვიანებით). M:N კავშირის შემთხვევაში, პირველ რიგში იქმნება რელაცია თითოეული რეგულარული არსის ტიპიდან EMPLOYEE და COURSE. შემდეგ იქმნება ახალი რელაცია (სახელად CERTIFICATE ნახაზზე 4-13b) Completes კავშირისათვის. CERTIFICATE–ის პირველადი გასაღები არის EmployeeID–ისა და CourseID–ის კომბინაცია, რომლებიც EMPLOYEE–ისა და COURSE–ის შესაბამისი პირველადი გასაღებებია. როგორც დიამაგრზეა მითითებული, ეს ატრიბუტები გარე გასაღებებია, რომლებიც შესაბამის წერტილზე გასასვლელად "მიუთითებენ". არაგასაღები ატრიბუტი DateCompleted ასევე ჩნდება CERTIFICATE რელაციაში. მართალია აქ ნაჩვენები არ არის, მაგრამ სასურველია შეიქმნას სუროგატული პირველადი გასაღები CERTIFICATE რელაციისათვის.

FIGURE 4-13 **Example of mapping a *M:N* relationship**
(a) Completes relationship (*M:N*)



(b) Three resulting relations



ერთი-ერთთან ბინარული კავშირის ასახვა

ბინარული კავშირი ერთი-ერთთან შეიძლება განიხილებოდეს, როგორც ერთი-მრავალთან კავშირის კერძო შემთხვევა. ამგვარი კავშირის რელაციად ასახვის პროცესი მოითხოვს ორ ბიჯს. პირველი, იქმნება ორი რელაცია, თითო მონაწილე არსის თითოეული ტიპებისთვის. მეორე, ერთი რელაციის პირველადი გასაღები ჩაისმის გარე გასაღებად, მეორე რელაციაში.

1:1 კავშირშირისას ერთი-ერთი მიმართულებით კავშირი თითქმის ყოველთვის არასავალდებულოა, ხოლო მეორე მიმართულებით სავალდებულო (ამ ტერმინების აღნიშვნის გადახედვა შეგიძლიათ დიაგრამაზე 2-1). კავშირის არასავალდებულო მხარეს არსებულ რელაციაში უნდა მიუთითოთ გარე გასაღები, რომელიც მიუთითებს არსის ტიპის პირველად გასაღებზე, რომელსაც აქვს სავალდებულო მონაწილეობა 1:1 კავშირში. ეს მიდგომა ხელს შეუშლის გარე გასაღების ატრიბუტში null მნიშვნელობების შენახვის აუცილებლობას. თავად კავშირთან დაკავშირებული ნებისმიერი ატრიბუტი ასევე შედის იმავე რელაციაში, სადაც გარე გასაღები (აქვე აღვნიშნოთ, რომ ნორმალიზაციის თვალსაზრისით, რასაც მოგვიანებით განვიხილავთ, თუ კავშირის შეიცავს ატრიბუტებს, უმჯობესია კავშირი წარმოავდგინოთ ასოციაციური არსის სახით).

ამ პროცედურის გამოყენების მაგალითი ნაჩვენებია ნახაზზე 4-14. დიაგრამა 4-14a აჩვენებს ორობითი 1:1 კავშირს არსის ტიპების NURSE (მედდა) და CARE CENTER (მზრუნველობის ცენტრი) შორის. თითოეულ მზრუნველობის ცენტრს უნდა ჰყავდეს ექთანი, რომელიც პასუხისმგებელია ამ ცენტრში. ამრიგად, CARE CENTER–ის კავშირი NURSE-თან სავალდებულოა, ხოლო NURSE–ს კავშირი CARE CENTER-თან არასავალდებულოა (ვინაიდან ნებისმიერი ექთანი შეიძლება იყოს ან არ იყოს პასუხისმგებელი ზრუნვის ცენტრში). ატრიბუტი Date Assigned (დანიშვნის თარიღი) თან ერთვის In Charge კავშირს.

ამ კავშირების ასახვის შედეგი რელაციების კავშირებით ნაჩვენებია ნახაზზე 4-14 ბ. ორი რელაცია NURSE და CARE CENTER იქმნება ორი არსის ტიპებისგან. იმის გამო, რომ CARE CENTER არასავალდებულო მონაწილეა, ამ რელაციაში განთავსებულია გარე გასაღები. ამ შემთხვევაში, გარე გასაღებია NurseInCharge. მას აქვს იგივე დომენი, როგორც NurseID და პირველადი გასაღების მიმართება ნაჩვენებია ნახატზე. ატრიბუტი DateAssigned ასევე მდებარეობს CARE CENTER- ში და ნებადართული არ იქნება მისი ნულოვანობა.

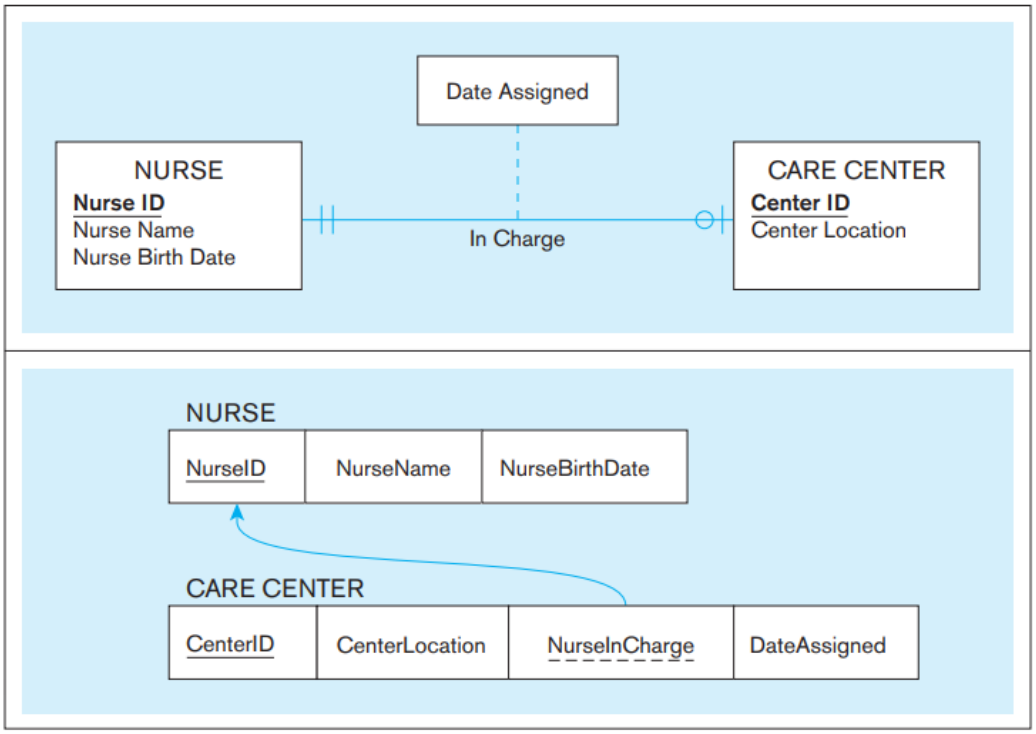


FIGURE 4-14 Example of mapping a binary 1:1 relationship
(a) In Charge relationship (binary 1:1)

(b) Resulting relations

ბიჯი 4: ასოციაციური არსების ასახვა

როგორც ადრე აღვნიშნეთ, როდესაც მონაცემთა მოდელის შექმნისას „მრავალი-მრავალთან“ კავშირთან შეჯახებისას შესაძლებელია კავშირის მოდელირებისათვის ავირჩიოთ ასოციაციური არსის გამოყენება E-R დიაგრამაზე. ეს მიდგომა ყველაზე შესაფერისია მაშინ, როდესაც საბოლოო მომხმარებელს შეუძლია უკეთესად წარმოაჩინოს კავშირის, როგორც არსის ტიპი და არა როგორც M:N კავშირი. ასოციაციური არსის ასახვა მოიცავს იგივე ნაბიჯებს, როგორიცაა M: N კავშირის ასახვა, როგორც ეს აღწერილია მე -3 ბიჯზე.

პირველი ბიჯი არის სამი რელაციის შექმნა: თითი თითო ორი მონაწილე არსის ტიპისთვის და მესამე ასოციაციური არსისთვის. ასოციაციური არსისაგან წარმოქმნილ რელაციას ასოციაციურ რელაციად ვგულისხმობთ. მეორე ბიჯი დამოკიდებულია იმაზე, იყო თუ არა E-R დიაგრამაზე ასოციაციური არსისთვის დანიშნული იდენტიფიკატორი.

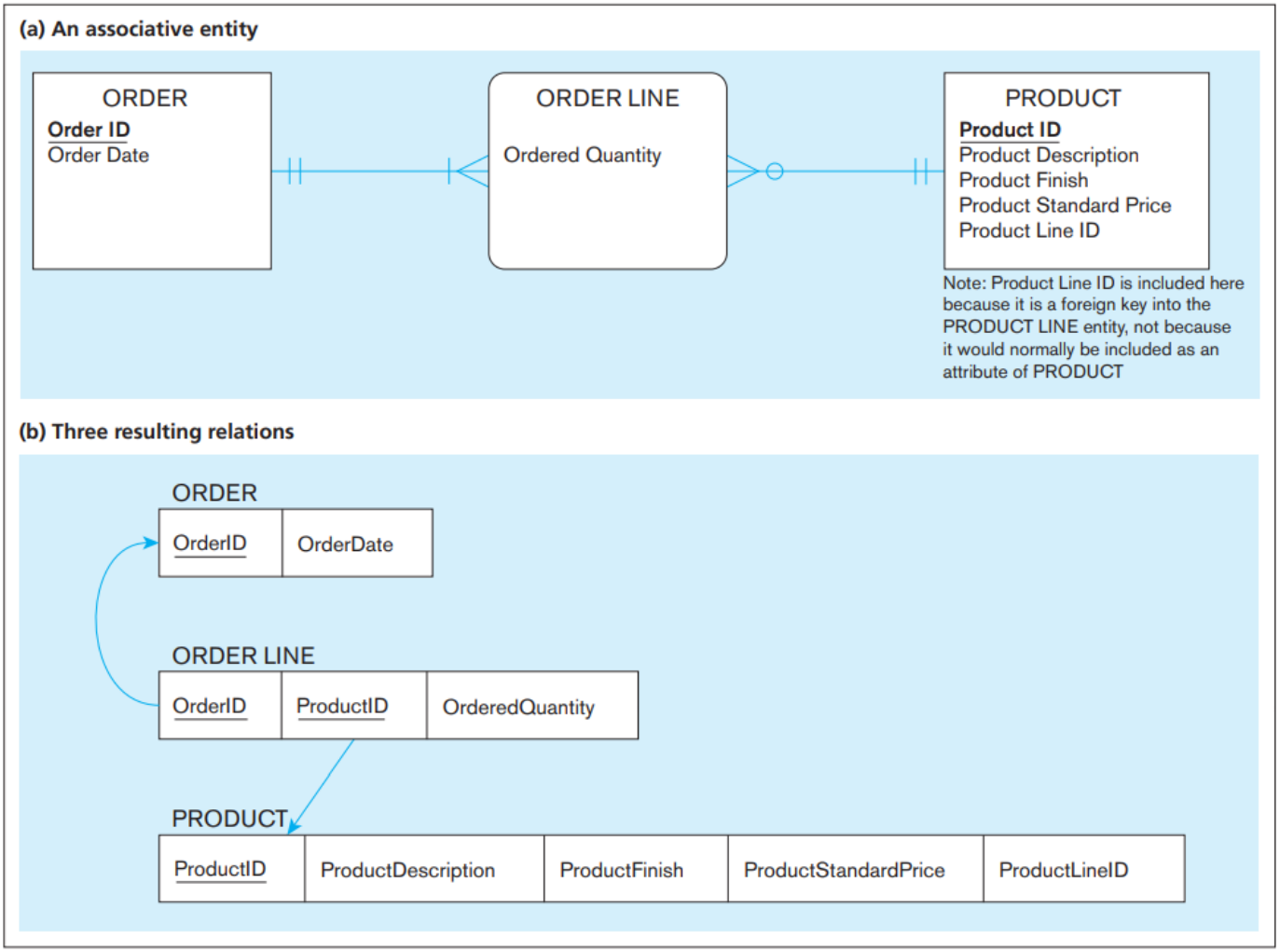
იდენტიფიკატორი არ არის დანიშნული

თუ იდენტიფიკატორი არ იყო დანიშნული, ასოციაციური რელაციის ნაგულისხმევი პირველადი გასაღები არის კომპოზიტური გასაღები, რომელიც შედგება ორი ძირითადი გასაღები ატრიბუტისგან დანარჩენი ორი რელაციიდან. ეს ატრიბუტები შემდეგ გარე გასაღებებს წარმოადგენს, რომლებიც სხვა დანარჩენ რელაციებს უკავშირდება.

ამ საქმის მაგალითი ნაჩვენებია ნახაზზე 4-15. დიაგრამა 4-15a აჩვენებს ასოციაციურ არსს ORDER LINE, რომელიც აკავშირებს ORDER და PRODUCT არსის ტიპებს Pine Valley

Furniture Company- ში (იხ. სურათი 2-22). დიაგრამა 4-15 ბ აჩვენებს სამ რელაციას, რომელიც ამ სქემიდან გამომდინარეობს. გავითვალისწინოთ ამ მაგალითის მსგავსება M:N კავშირთან, რომელიც ნაჩვენებია ნახაზზე 4-13.

FIGURE 4-15 **Example of mapping an associative entity**



იდენტიფიკატორი დანიშნულია

ზოგჯერ მონაცემთა მოდელის შემმუშავებლები ერთ-ერთ ატრიბუტს ანიჭებენ იდენტიფიკატორის ფუნქციას ასოციაციური არსის ტიპისთვია E-R დიაგრამაზე. ეს ფაქტი ორმა მიზეზმა შეიძლება განაპირობოს მონაცემთა კონცეპტუალური მოდელის შემუშავების ეტაპზე:

1. ასოციაციური არსის ტიპს გააჩნია ბუნებრივი ერთი ატრიბუტი იდენტიფიკატორი, რომელიც ნაცნობია საბოლოო მომხმარებლებისთვის.

2. ნაგულისხმევ იდენტიფიკატორს (რომელიც შედგება იდენტიფიკატორებისგან თითოეული არსის თითოეული ტიპებისთვის) არ შეუძლია ცალსახად განსაზღვროს ასოციაციური არსის ეგზემპლარები.

ეს მოტივები ავსებს სუროგატი გასაღების შექმნის ძირითადი მიზეზებს. ასოცირებული არსის ჩანაწერის პროცესი ამ შემთხვევაში იცვლება: იქმნება ახალი (ასოციაციური) რელაცია ასოციაციური არსის წარმოსადგენად; ამასთან, ამ რელაციის ძირითადი გასაღები არის E-R დიაგრამაზე მინიჭებული იდენტიფიკატორი (და არა ნაგულისხმევი გასაღები). ორი ძირითადი არსის ძირითადი გასაღები ასოცირებულ რელაციაში შედის როგორც გარე გასაღები.

ამ პროცესის მაგალითი ნაჩვენებია ნახაზზე 4-16. დიაგრამა 4-16a გვიჩვენებს ასოციაციური არსის ტიპს SHIPMENT, რომელიც აკავშირებს CUSTOMER და VENDOR არსის ტიპებს.

Shipment ID შეირჩა SHIPMENT- ის იდენტიფიკატორად ორი მიზეზის გამო:

1. Shipment ID არის ბუნებრივი იდენტიფიკატორი ამ არსისთვის, რომელიც ძალიან ნაცნობია საბოლოო მომხმარებლებისთვის.

2. ნაგულისხმევი იდენტიფიკატორი, რომელიც შედგება CustomerID და VendorID კომბინაციისგან, ცალსახად არ განსაზღვრავს გადაზიდვის შემთხვევებს. სინამდვილეში, მოცემული გამყიდველი, როგორც წესი, ბევრ გადაზიდვას ახორციელებს მოცემულ მომხმარებელზე. ატრიბუტის თარიღის დამატებაც არ იძლევა გარანტიას უნიკალურობის, რადგან მოცემული თარიღისთვის კონკრეტული გამყიდველის მიერ შეიძლება იყოს ერთზე მეტი გადაზიდვა განხორციელებული. სუროგატი გასაღებს ShorageID ცალსახად განსაზღვრავს თითოეულ გადაზიდვას.

SHIPMENT ასოციაციურ არსთან ასოცირებული ორი არაგასაღები ატრიბუტია Shipment Date და Shipment Amount.

ამ არსების რელაციაბთან შეთანადების ასახვის შედეგი ნაჩვენებია დიაგრამა 4-16 ბ. ახალ ასოციაციურ რელაციას SHIPMENT დაარქვეს. პირველადი გასაღებია ShorageID. CustomerID და VendorID მოცემულია როგორ გარე გასაღებები ამ რელაციაში და ShriageDate და ShriageAmount არიან არა გასაღები ატრიბუტები. ასევე შესაძლებელია, დიზაინერმა გადაწყვიტოს, როგორც ლოგიკური მოდელირების პროცესის ნაწილი, დაამატოს სუროგატი გასაღები იმ რელაციაში, რომელსაც ადრე არ ჰქონდა. ამ შემთხვევებში რეკომენდებულია კონცეპტუალური მოდელის განახლება, მისი მუდმივი შენარჩუნების მიზნით.

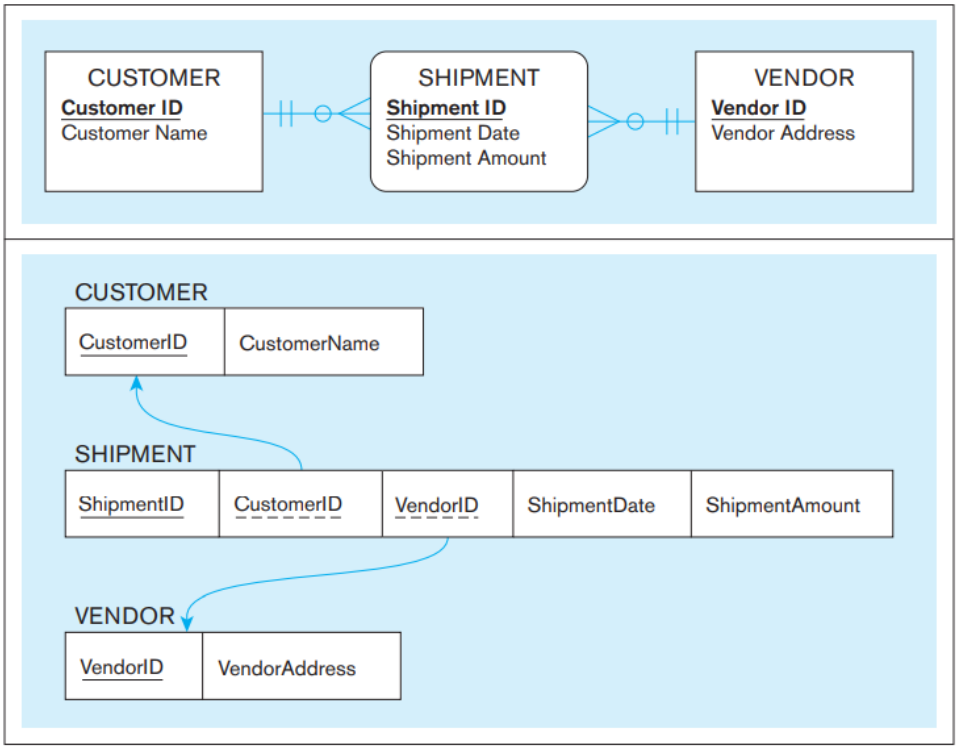


FIGURE 4-16 Example of mapping an associative entity with an identifier
(a) SHIPMENT associative entity

(b) Three resulting relations

ბიჯი 5: უნიარული კავშირების ასახვა

ადრე ჩვენ განვსაზღვრეთ უნარული კავშირი, როგორც ურთიერთობა ერთი არსის ტიპის ეგზემპლარებს შორის. უნარულ კავშირებს რეკურსიულ კავშირებსაც უწოდებენ. უნარული კავშირების ორი უმნიშვნელოვანესი შემთხვევა განიხილება - კავშირი „ერთი-მრავალთან“ და „მრავალი-მრავალთან“. ამ ორ შემთხვევას განვიხილავთ ცალკე, რადგან ასახვის მიდგომა ორი ტიპისთვის გარკვეულწილად განსხვავებულია.

უნარული კავშირი „ერთი-მრავალთან“

არსის ტიპი უნარულ კავშირში აისახება რელაციაში პროცედურით, რომელიც აღწერილია ბიჯში 1. შემდეგ, იმავე რელაციას ემატება გარე გასაღები ატრიბუტი; ეს ატრიბუტი ეფუძნება პირველადი გასაღების მნიშვნელობებს იმავე რელაციაში (ამ გარე გასაღებას იგივე დომენი უნდა ჰქონდეს, როგორც პირველად გასაღებს). გარე გასაღების ამ ტიპს **რეკურსიულ გარე გასაღებს** უწოდებენ.

დიაგრამა 4-17a გვიჩვენებს უნიარულ კავშირს, სახელწოდებით Manages, რომელიც ორგანიზაციის თითოეულ თანამშრომელს უკავშირებს სხვა თანამშრომელს, რომელიც მისი

მენეჯერია. თითოეულ თანამშრომელს შეიძლება ჰყავდეს ერთი მენეჯერი; მოცემულმა თანამშრომელმა შეიძლება მართოს ნულიდან მრავალი თანამშრომელი.

EMPLOYEE რელაცია, რომელიც წარმოიქმნება ამ არსის და კავშირის ასახვიდან გამომდინარე, ნაჩვენებია ნახაზზე 4-17b. (რეკურსიული) გარე გასაღებს რელაციაში დაერქვა ManagerID. ამ ატრიბუტს აქვს იგივე დომენი, როგორც პირველად გასაღებს EmployeeID. ამ რელაციის თითოეული სტრიქონი ინახავს შემდეგ მონაცემებს მოცემული თანამშრომლისთვის: EmployeeID, EmployeeName, EmployeeDateOfBirth და ManagerID (ანუ EmployeeID ამ თანამშრომლის მენეჯერისთვის). გავითვალისწინოთ, რომ რადგან ეს გარე გასაღებია, ManagerID მიუთითებს EmployeeID.

უნარული კავშირი „მრავალი-მრავალთან“

ამ ტიპის კავშირით იქმნება ორი რელაცია: ერთი წარმოადგენს არსის ტიპს და მეორე - ასოციაციურ არსს M:N კავშირის წარმოსადგენად. ასოციაციური რელაციის პირველადი გასაღები შედგება ორი ატრიბუტისგან. ამ ატრიბუტებს (რომელთაც არ უნდა ჰქონდეთ ერთიდაიგივე სახელი) ორივე იღებს თავის მნიშვნელობებს პირველი რელაციის პირველადი გასაღებიდან. კავშირის ნებისმიერი არაგასაღები ატრიბუტი შედის ასოციაციურ რელაციაში.

უნიარული M: N კავშირის ასახვის მაგალითი ნაჩვენებია ნახაზზე 4-18. დიაგრამა 4-18 ა აჩვენებს bill-of-materials კავშირს (კავშრი არწერს ურთიერთდამოკიდებულებას იმ ნივთებს შორის, რომლებიც სხვა ნივთებისაგან ან კომპონენტებისგან არის აწყობილი). კავშირი (ე.წ.- Contains შეიცავს) არის M:N, რადგან მოცემული ნივთი შეიძლება შეიცავდეს მრავალ კომპონენტს (ნივთს) და, პირიქით, ნივთი (კომპონენტი) გამოყენება მრავალ სხვა ნივთში (კომპონენტში).

რელაციები, რომლებიც წარმოიქმნება ამ არსის და მისი კავშირის სქემიდან ნაჩვენებია ნახაზზე 4-18 ბ. ITEM რელაცია პირდაპირ იმავე ტიპის არსისდან არის ასახული. კომპონენტი არის ასოციაციური რელაცია, რომლის პირველადი გასაღები შედგება ორი ატრიბუტისგან, რომლებსაც უწოდეს ItemNo და ComponentNo. ატრიბუტი Quantity არის ამ რელაციის არაგასაღები ატრიბუტი, რომელიც მოცემული ნივთისთვის აღრიცხავს ამ ერთეულში გამოყენებული კონკრეტული კომპონენტის ერთეულის რაოდენობას. გავითვალისწინოთ, რომ ItemNo და ComponentNo მიუთითებენ ITEM რელაციის პირველად გასაღებზე (ItemNo). ბუნებრივია ამ რელაციისთვის სუროგატი გასაღების მიცემა, რომ თავიდან იქნას აცილებული კომპოზიტურ გასაღებთან დაკავშირებული რაიმე პრაქტიკული სირთულე.

ამ რელაციის გამოთხოვნა ადვილად შეგვიძლია, მაგალითად, რათა განვსაზღვროთ, მოცემული ნივთის კომპონენტები. შემდეგი SQL მოთხოვნა ჩამოთვლის ყველა კომპონენტებს (და მათი რაოდენობას) პუნქტის 100 ნომრისთვის:

```
SELECT ComponentNo, Quantity
```

```
FROM Component_T
WHERE ItemNo = 100;
```

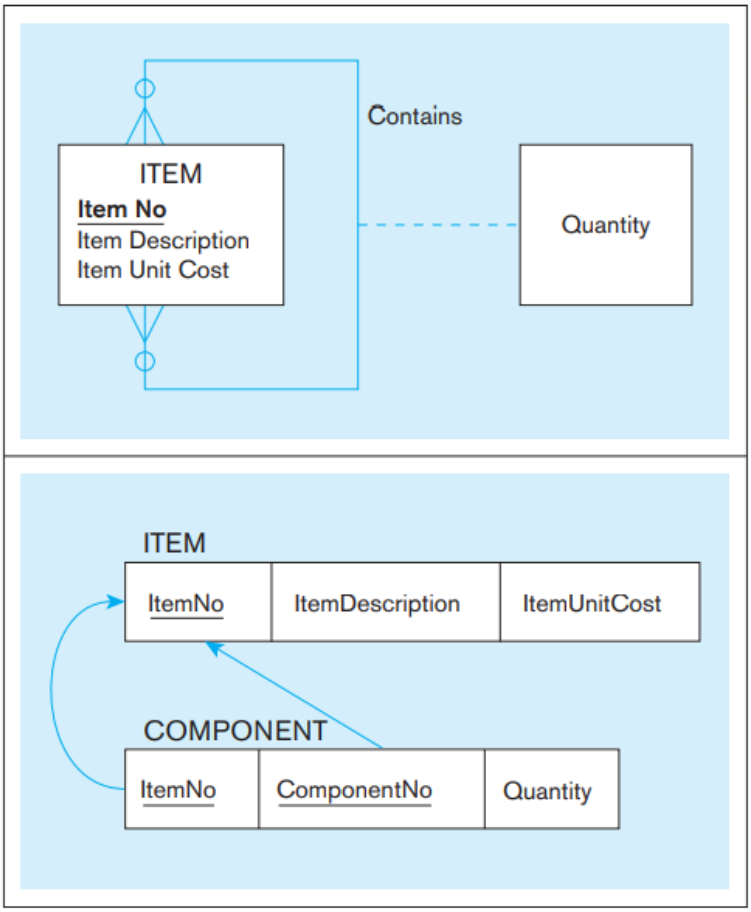


FIGURE 4-18 Example of mapping a unary *M:N* relationship
(a) Bill-of-materials relationship Contains (*M:N*)

(b) ITEM and COMPONENT relations

ბიჯი 6: სამეული (და n-ური) სავშირების ასახვა

გავიხსენოთ, რომ სამეული კავშირი არის ურთიერთობა სამ არსის ტიპებს შორის. საზოგადოდ მიღებულია სამეული კავშირის ასოციაციურ არსად გარდაქმნა, რათა უფრო ზუსტად მოხდეს მონაწილეობის შეზღუდვების წარმოდგენა.

ასოციაციური არსის ტიპის ასაგებად, რომელიც აკავშირებს სამი რეგულარული არსის ტიპს, ჩვენ ვქმნით ახალ ასოციაციურ რელაციას. ამ რელაციის ნაგულისხმევი პირველადი გასაღები შედგება სამი მონაწილე არსის ტიპების პირველადი გასაღები ატრიბუტებისაგან (ზოგიერთ შემთხვევაში საჭიროა დამატებითი ატრიბუტები უნიკალური პირველად გასაღების შესაქმნელად). ეს ატრიბუტები შემდეგ მოქმედებს გარე გასაღებების როლში, რომლებიც მიუთითებენ მონაწილე არსების ტიპების ინდივიდუალურ პირველად გასაღებებზე. ასოციაციური არსის ტიპის ნებისმიერი ატრიბუტი ხდება ახალი რელაციის ატრიბუტი.

სამეული კავშირის ასახვის მაგალითი (წარმოდგენილია როგორც ასოციაციური არსის ტიპი) ნაჩვენებია ნახაზზე 4-19. დიაგრამა 4-19a არის E-R სეგმენტი (ან ხედი), რომელიც წარმოადგენს პაციენტს, რომელსაც მკურნალობს ექიმი. ასოციაციური არსის ტიპს PATIENT TREATMENT (პაციენტის მკურნალობა) აქვს ატრიბუტები PTreatment Date, PTreatment time, და PTreatment results; მნიშვნელობები ჩაიწერება ამ ატრიბუტებისთვის თითოეული ეგზემპლარისათვის PATIENT TREATMENT-ში.

ამ წარმოდგენის ასახვის შედეგი ნაჩვენებია ნახაზზე 4-19b. პირველადი გასაღების ატრიბუტები PatientID, PhysicianID და TreatmentCode ხდებიან გასაღებები PATIENT TREATMENT-ში. TREATMENT–ის გარე გასაღებს ერქმევა PTreatmentCode PATIENT LREATMENT-ში. ამ სვეტის სახელს ვიყენებთ იმის საილუსტრაციოდ, რომ გარე გასაღების სახელი არა სავალდებულო იყოს იგივე რაც პირველადი გასაღების სახელი, რომელსაც ის უკავშირდება, რადგან მნიშვნელობები მოდის ერთი და იგივე დომენიდან. ეს სამი ატრიბუტი არის პაციენტის მკურნალობის პირველადი გასაღების კომპონენტები. ამასთან, ისინი ცალსახად არ განსაზღვრავენ მოცემულ მკურნალობას, რადგან პაციენტმა შეიძლება იგივე მკურნალობა მიიღოს იმავე ექიმისგან ერთზე მეტჯერ. შესაძლოა ატრიბუტის Date (თარიღის), როგორც პირველადი გასაღების ნაწილად (სხვა დანარჩენ სამ ატრიბუტთან ერთად) ჩართვამ წარმოქმნის პირველადი გასაღები. ეს ასე მოხდება, თუ მოცემული პაციენტი კონკრეტული ექიმისგან მხოლოდ ერთ მკურნალობას მიიღებს მოცემულ თარიღში. ამასთან, სავარაუდოდ, ეს ასე არ არის. მაგალითად, პაციენტს შეუძლია მიიღოს მკურნალობა დილით, შემდეგ ისევ იგივე მკურნალობა დღის მეორე ნახევარში. ამ საკითხის მოსაგვარებლად, ჩვენ პირველადი გასაღების ნაწილად მოვიყვანთ PTreatmentDate და PTreatmentTime. ამიტომ, პაციენტის მკურნალობის პირველადი გასაღები შედგება ხუთი ატრიბუტისგან, რომლებიც ნაჩვენებია ნახაზზე 4-19 ბ: PatientID, PhysicianID, TreatmentCode, PTreatmentDate და PTreatmentTime. ერთადერთი არაფრისმომცემი ატრიბუტი არის PTreatmentResults.

მიუხედავად იმისა, რომ ეს პირველადი გასაღები ტექნიკურად სწორია, ის რთულია, რთულია მისი მართვა და მიდრეკილია შეცდომებისკენ. უკეთესი მიდგომაა სუროგატი გასაღების შემოღება, მაგალითად, PTreatmentID, ანუ სერიული ნომერი, რომელიც ცალსახად განსაზღვრავს თითოეულ მკურნალობას. ამ შემთხვევაში, თითოეული ყოფილი პირველადი გასაღების ატრიბუტი გარდა PTreatmentDate და PTreatmentTime ხდება გარე გასაღები PATIENT TREATMENT რელაციაში. სხვა მსგავსი მიდგომაა ე.წ. საწარმოს გასაღების გამოყენება, რომელსაც მოგვიანებით აღვწერთ.

ბიჯი 7: სუპერტიპი/ქვეტიპი კავშირის ასახვა

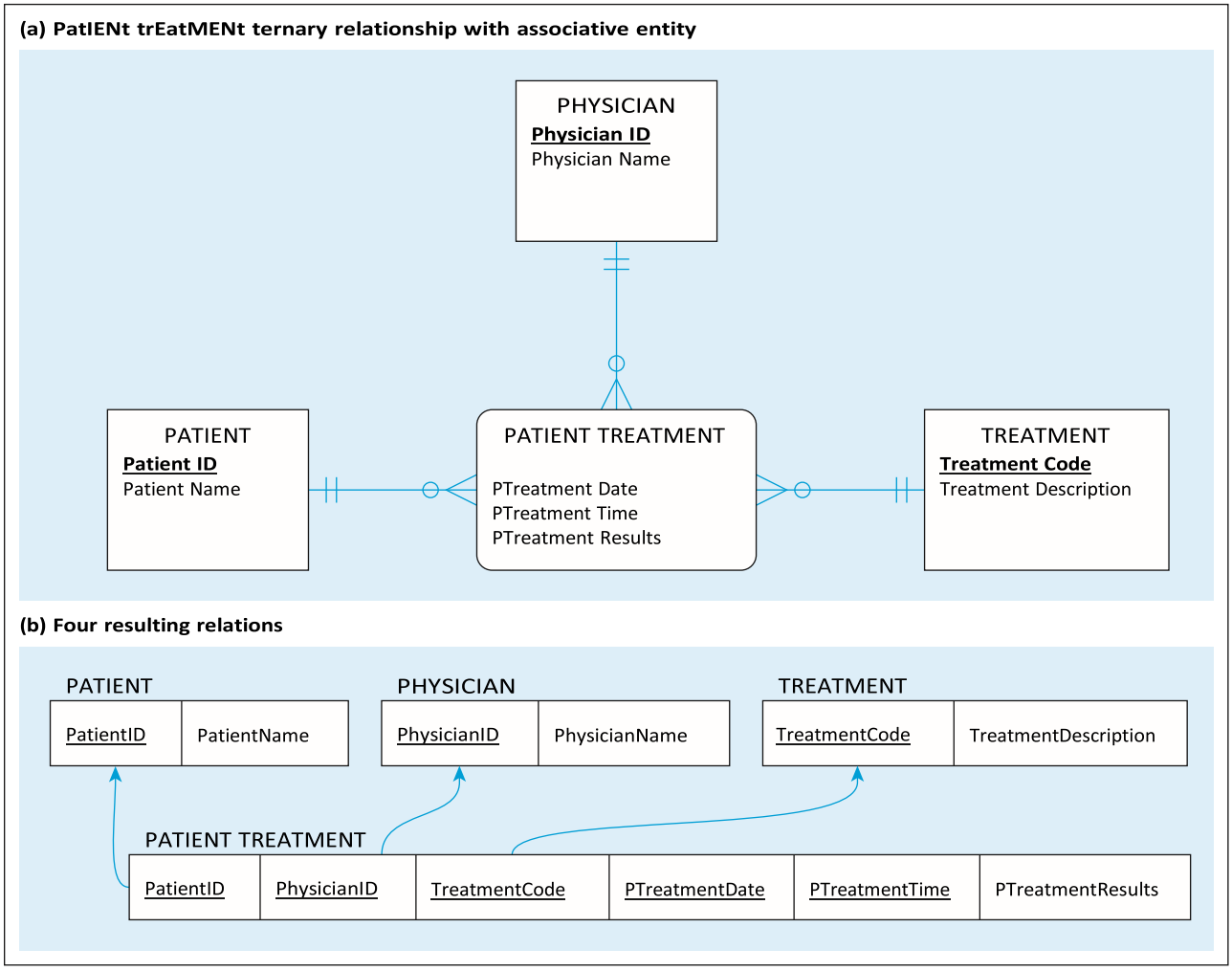
რელაციური მონაცემების მოდელი პირდაპირ არ ახდენს სუპერტიპი/ქვეტიპი კავშირის მხარდაჭერას. საბედნიეროდ, არსებობს სხვადასხვა სტრატეგია, რომელთა საშუალებით

მონაცემთა ბაზის დამპროექტებლემს შეუძლიათ გამოიყენონ ეს კავშირები რელაციური მონაცემების მოდელთან. ყველაზე ხშირად გამოიყენება შემდეგი სტრატეგია:

1. შევქმენათ ცალკე რელაციები სუპერტიპისთვის და მისი თითოეული ქვეტიპისთვის;
2. სუპერტიპისთვის შექმნილ რელაციას მივანიჭოთ ატრიბუტები, რომლებიც საერთოა სუპერტიპის ყველა წევრისთვის, პირველადი გასაღების ჩათვლით;
3. თითოეული ქვეტიპისთვის რელაციას მივანიჭოთ სუპერტიპის პირველადი გასაღები და მხოლოდ ის ატრიბუტები, რომლებიც მხოლოდ ამ ქვეტიპისთვისაა დამახასიათებელი;
4. სუპერტიპის ერთი (ან მეტი) ატრიბუტის მივანიჭოთ ქვეტიპის დისკრიმინატორის ფუნქცია (ქვეტიპის დისკრიმინატორის როლი განხილული გვქონდა ადრე).

Дру

figure 4-19 **Example of mapping a ternary relationship**



ამ პროცედურის გამოყენების მაგალითი ნაჩვენებია 4-20 და 4-21 ნახაზებზე. დიაგრამა 4-20 გვიჩვენებს სუპერტიპს EMPLOYEE ქვეტიპებით HOURLY EMPLOYEE, SALARIED EMPLOYEE და CONSULTANT (ეს მაგალითი აღწერილი იყო ადრე, ხოლო დიაგრამა 4-20 არის ფიგურა 3-8-ის გამეორება). EMPLOYEE-ის პირველადი გასაღებია EmployeeNumber, ხოლო ატრიბუტი Employee Type ქვეტიპის დისკრიმინატორია.

ამ დიაგრამის ამ წესების გამოყენებით რელაციურ სტრუტურად ასახვის შედეგი ნაჩვენებია 4-21 ნახაზზე. გვაქვს სუპერტიპისთვის (EMPLOYEE) ერთი და სამივე ქვეტიპისთვის თითოთითო რელაცია. ოთხივე რელაციის პირველადი გასაღებია EmployeeNumber. პრეფიქსი გამოიყენება თითოეული ქვეტიპის პირველადი გასაღების სახელის გასარჩევად. მაგალითად, SEmployeeNumber არის SALARIED EMPLOYEE რელაციის პირველადი გასაღების სახელი. თითოეული ეს ატრიბუტი არის გარე გასაღები, რომელიც მიუთითებს სუპერტიპის პირველად გასაღებაზე, როგორც ეს მითითებულია დიაგრამაზე ისრებით. თითოეული ქვეტიპის რელაცია შეიცავს მხოლოდ იმ ატრიბუტებს, რომლებიც უნიკალურია ქვეტიპისთვის.

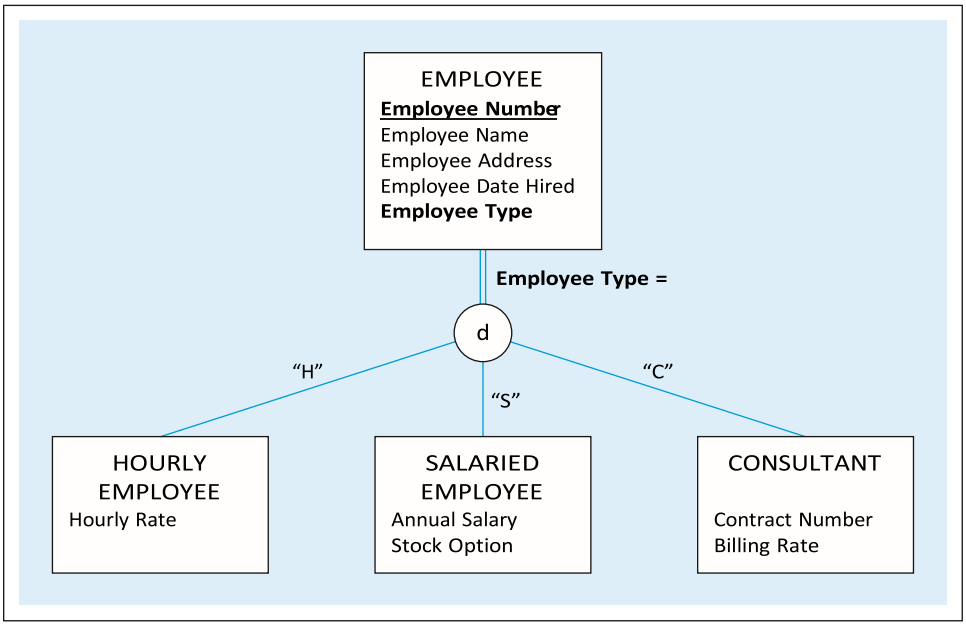


figure 4-20 Supertype/ subtype relationships

თითოეული ქვეტიპისთვის შეიძლება შეიქმნას რელაცია, რომელიც შეიცავს ამ ქვეტიპის ყველა ატრიბუტს (როგორც სპეციფიკური, ასევე მემკვიდრეობით მიღებული) SQL

ბრძანების გამოყენებით, რომელიც უერთდება ქვეტიპს თავისი სუპერტიპით. მაგალითად, დავუშვათ, რომ გვინდა ვაჩვენოთ ცხრილი, რომელიც შეიცავს ყველა ატრიბუტს SALARIED EMPLOYEE– სთვის. გამოიყენება შემდეგი ბრძანება:

```
SELECT *
FROM Employee_T, SalariedEmployee_T
WHERE EmployeeNumber = SEmployeeNumber;
```

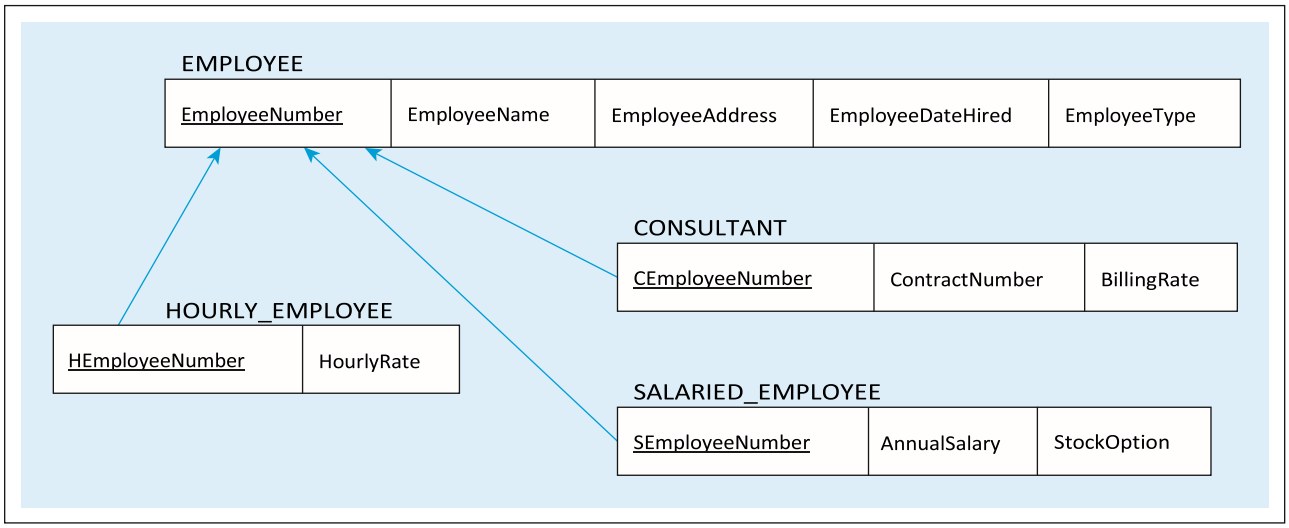


figure 4-21 **Mapping supertype/subtype relationships to relations**

tABle 4-2 Summary of EEr-to-relational transformations

EEr Structure	relational representation (Sample Figure)
Regular entity	Create a relation with primary key and nonkey attributes (Figure 4-8)
Composite attribute	Each component of a composite attribute becomes a separate attribute in the target relation (Figure 4-9)
Multivalued attribute	Create a separate relation for multivalued attribute with composite primary key, including the primary key of the entity (Figure 4-10)
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this entity depends) and nonkey attributes (Figure 4-11)
Binary or unary 1:*M* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side (Figure 4-12; Figure 4-17 for unary relationship)

Binary or unary *M:N* relationship or associative entity without its own key	Create a relation with a composite primary key using the primary keys of the related entities plus any nonkey attributes of the relationship or associative entity (Figure 4-13, Figure 4-15 for associative entity, Figure 4-18 for unary relationship)
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity; if one side of the relationship is optional, place the foreign key of the entity on the mandatory side in the relation for the entity on the optional side (Figure 4-14)
Binary or unary *M:N* relationship or associative entity with its own key	Create a relation with the primary key associated with the associative entity plus any nonkey attributes of the associative entity and the primary keys of the related entities as foreign keys (Figure 4-16)
Ternary and *n*-ary relationships	Same as binary *M:N* relationships above; without its own key, include as part of primary key of relation for the relationship or associative entity the primary keys from all related entities; with its own surrogate key, the primary keys of the associated entities are included as foreign keys in the relation for the relationship or associative entity (Figure 4-19)
Supertype/subtype relationship	Create a relation for the superclass, which contains the primary and all nonkey attributes in common with all subclasses, plus create a separate relation for each subclass with the same primary key (with the same or local name) but with only the nonkey attributes related to that subclass (Figure 4-20 and 4-21)

EER დიაგრამის რელაციურ მონაცემთა მოდელად გარდაქმნების მოკლე არწერა ბიჯებით მოცემულია ცხრილში 4-2.

რელაციური მოდელის ნორმალიზაცია

კარგად სტრუქტურირებული რელაციები

რას წარმოადგენს კარგად სტრუქტურირებული რელაცია? ინტუიციურად, კარგად სტრუქტურირებული რელაცია შეიცავს მინიმალურ სიჭარბეს და მომხმარებლებს საშუალებას აძლევს შეცდომისა და შეუსაბამობის გარეშე ჩასვან, შეცვალონ და წაშალონ ცხრილის სტრიქონები. EMPLOYEE1 (სურათი 4-1) ასეთი რელაციაა. ცხრილის თითოეული სტრიქონი შეიცავს მონაცემებს, რომლებიც აღწერს ერთ თანამშრომელს, ხოლო თანამშრომლის მონაცემებში ნებისმიერი ცვლილება (მაგალითად, ხელფასის ცვლილება) შემოიფარგლება ცხრილის ერთ რიგში განსახორციელები ცვლილებით. ამის საპირისპიროდ, EMPLOYEE2 (სურათი 4-2b) არ არის კარგად სტრუქტურირებული რელაცია. თუ შევისწავლით ცხრილში მოცემულ მონაცემთა ნიმუშს, შევამჩნევთ მნიშვნელოვან სიჭარბეს. მაგალითად, EmpID-ის, Name-ს, DeptName-ს და Salary-ს მნიშვნელობები გამოჩნდება ორ სხვადასხვა სტრიქონში 100, 110 და 150 თანამშრომლებისთვის. შესაბამისად, თუ EmpID- 100-ის შესაბამისი თანამშრომლის ხელფასი შეიცვალა, ეს ფაქტი ორ სტრიქონში უნდა ჩავწეროთ.

EMPLOYEE1

EmpID	Name	DeptName	Salary
100	Margaret Simpson	Marketing	48,000
140	Allen Beeton	Accounting	52,000
110	Chris Lucero	Info Systems	43,000
190	Lorenzo Davis	Finance	55,000
150	Susan Martin	Marketing	42,000

FIGURE 4-1 EMPLOYEE1 relation with sample data

ცხრილში სიჭარბემ შეიძლება გამოიწვიოს შეცდომები ან შეუსაბამობები (ე.წ. ანომალიები), როდესაც მომხმარებელი ცდილობს განაახლოს ცხრილში მოცემული მონაცემები. ჩვენ, როგორც წესი, გვაწუხებს სამი სახის ანომალია:

4. *ჩასმის ანომალია* - დავუშვათ, რომ EMPLOYEE2–ს უნდა დავამატოთ ახალი თანამშრომელი. ამ რელაციის პირველადი გასაღები არის EmpID და CourseTitle კომბინაცია. ამიტომ, ახალი სტრიქონის ჩასასმელად, მომხმარებელმა უნდა მიუთითოს როგორც EmpID-ის, ასევე CourseTitle-ის მნიშვნელობები, რადგან პირველადი გასაღების მნიშვნელობები არ შეიძლება იყოს ნულოვანი ან არარსებული. ეს ანომალიაა, რადგან

მომხმარებელს უნდა შეეძლოს შეიტანოს თანამშრომლის მონაცემები, კურსებზე მონაცემების მიწოდების გარეშე.

5. *წაშლის ანომალია* - დავუშვათ, რომ ცხრილიდან წაიშლება EmpID- 140-ის შესაბამისი პერსონალის მონაცემები. ეს გამოიწვევს ინფორმაციის დაკარგვას იმის შესახებ, რომ ამ თანამშრომელმა დაასრულა კურსი (Tax Acc) 12/8/2015. ფაქტობრივად, ეს კარგავს ინფორმაციას, რომ ამ კურსს ჰქონდა შეთავაზება, რომელიც დასრულდა ამ დღეს.

6. *მოდიფიკაციის ანომალია* - დავუშვათ, რომ EmpID- 100-ის შესაბამის თანამშრომელს გაეზარდა ხელფასი. ჩვენ უნდა დავაფიქსიროთ ეს ზრდა თითოეული ამ თანამშრომლის შესაბამის ყოველ სტრიქონში (ორი შემთხვევა ნახაზზე 4-2); წინააღმდეგ შემთხვევაში, მონაცემები არათანმიმდევრული იქნება.

(b) EMPLOYEE2 relation

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
110	Chris Lucero	Info Systems	43,000	C++	4/22/2015
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/2015
150	Susan Martin	Marketing	42,000	Java	8/12/2015

ეს ანომალიები მიუთითებს, რომ EMPLOYEE2 არ არის კარგად სტრუქტურირებული რელაცია. ამ რელაცის პრობლემა ისაა, რომ იგი შეიცავს მონაცემებს ორი არსის შესახებ: EMPLOYEE და COURSE. ჩვენ გამოვიყენებთ ნორმალიზაციის თეორიას EMPLOYEE2-ის ორ რელაციად დაყოფისთვის. შედეგად მიღებული ერთ-ერთი რელაციაა EMPLOYEE1 (სურათი 4-1). მეორეს EMP COURSE- ს დავარქმევთ, რომელიც მოცემულია მონაცემების ნიმუშით 4-7 ნახაზზე. ამ რელაციის პირველადი გასაღები არის EmpID-ისა და CourseTitle-ის კომბინაცია და ამ ატრიბუტების სახელებს ხაზს ვუსვამთ დიაგრამა 4-7-ზე, ამ ფაქტის გასაზრდელად. დიაგრამა 4-7-ზე გამოსახული EMP COURSE არ შეიცავს წინ აღწერილი ანომალიების ტიპებს და, შესაბამისად, კარგად არის სტრუქტურირებული.

FIGURE 4-7 EMP COURSE

EmpID	CourseTitle	DateCompleted
100	SPSS	6/19/2015
100	Surveys	10/7/2015
140	Tax Acc	12/8/2015
110	Visual Basic	1/12/2015
110	C++	4/22/2015
150	SPSS	6/19/2015
150	Java	8/12/2015

https://us02web.zoom.us/rec/share/4fz9CCtvgmMYBm7w0_vR7-MX9iYVgYN4qe8U4n-mMJIiPv4tibbCWSAxduuNPiPJ._r6deEeVdrkCrCI5?startTime=1617699864000

ნორმალიზაცია

EER დიაგრამების რელაციურ მოდელად ადრე აღწერილი ბიჯების შესაბამისად გარდაქმნას კარგად სტრუქტურირებულ რელაციურ რელაციებთან მივყავართ. ამასთან, არ არსებობს გარანტია, რომ ყველა ანომალია მოიხსნება ამ ბიჯების დაცვით. ნორმალიზება არის ფორმალური პროცესი იმის დასადგენად, რომელი ატრიბუტები უნდა დაჯგუფდეს რელაციებში ისე, რომ ყველა ანომალია მოიხსნას. მაგალითად, ჩვენ გამოვიყენეთ ნორმალიზაციის პრინციპები EMPLOYEE2 ცხრილის (მისი ზედმეტობის) EMPLOYEE1 (ნახაზი 4-1) და EMP COURSE (ნახაზი 4-7) გადასაკეთებლად. მონაცემთა ბაზის შემუშავების მთელი პროცესის დროს ძირითადი ორი შემთხვევა არსებობს, როდესაც ჩვეულებრივ შესაძლებელია სარგებლის მიღება ნორმალიზაციის გამოყენებით:

1. *მონაცემთა ბაზის ლოგიკური დაპროექტებისას* - უნდა გამოვიყენოთ ნორმალიზაციის ცნებები იმ კავშირების ხარისხის შესამოწმებლად, რომლებიც მიღებულია E-R დიაგრამებიდან;
2. *ძველი სისტემების უკუდაპროექტებისას* - უკუდაპროექტებისას ძველი სისტემების მრავალი ცხრილი და მომხმარებლის ხედები ჭარბია და მოიცავენ ანომალიებს.

ჯერჯერობით ჩვენ წარმოვადგინეთ კარგად სტრუქტურირებული რელაციების ინტუიციური განხილვა; ამასთან, ჩვენ გვჭირდება ასეთი რელაციების ფორმალური განმარტებები, მათი შემუშავების პროცესთან ერთად. ნორმალიზაცია არის ანომალიების შემცველი რელაციების თანმიმდევრული შემცირების პროცესი, უფრო მცირე ზომის, კარგად სტრუქტურირებული რრელაციების წარმოსაქმნელად. ნორმალიზაციის რამდენიმე ძირითადი მიზანია:

1. მინიმუმამდე შემცირდეს მონაცემთა სიჭარბე, რითაც თავიდან იქნას აცილებული ანომალიები და შენარჩუნდეს შენახვის ადგილი.

2. რეფერენტული მთლიანობის (დამოწმებითი ერთიანობა) შეზღუდვების გამოყენების გამარტივება.

3. გამარტივდეს მონაცემთა მხარდაჭერა (ჩასმა, განახლება და წაშლა).

4. მოხდეს უკეთესი დიზაინის მხარდაჭერა, რომელიც წარმოადგენს რეალური სამყაროს უკეთეს ასახვას და უფრო მყარ საფუძველს სამომავლო გაფართოებისათვის.

ნორმალიზება არ იძლევა ვარაუდს იმის შესახებ, თუ როგორ იქნება გამოყენებული მონაცემები დისპლეიზე, მოთხოვნებსა თუ ანგარიშებში. ნორმალიზაცია, იმის საფუძველზე, რასაც ჩვენ ნორმალურ ფორმებს და ფუნქციურ დამოკიდებულებებს დავარქმევთ, განსაზღვრავს ბიზნესის წესებს და არა მონაცემთა გამოყენებას. გარდა ამისა, მონაცემთა ნორმალიზება ხდება მონაცემთა ბაზის ლოგიკური დაპროექტების დასრულებისთანავე. ამრიგად, ნორმალიზება არანაირ შეზღუდვას არ ადებს იმაზე, თუ ფიზიკურად როგორ შეიძლება ან უნდა იყოს შენახული მონაცემები ან, შესაბამისად, ეფექტურად დამუშავებული. ნორმალიზება არის ლოგიკური მონაცემების მოდელირება, რომელიც გამოიყენება იმის უზრუნველსაყოფად, რომ მონაცემები კარგად იყოს სტრუქტურირებული ორგანიზაციის მასშტაბით.

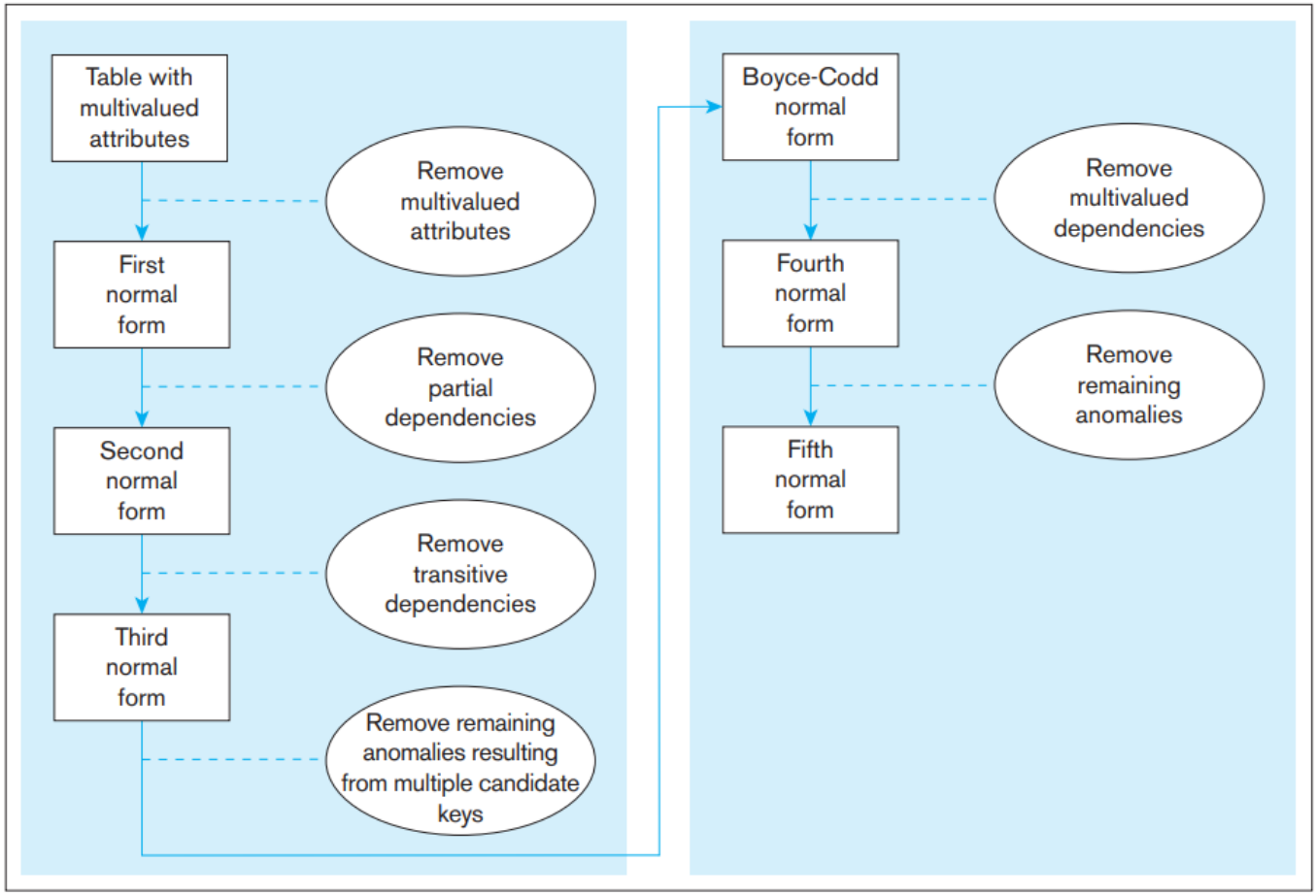
ბიჯები ნორმალიზაციაში

ნორმალიზაციის განხორციელება შესაძლებელია ეტაპობრივად, რომელთაგან თითოეული შეესაბამება ნორმალურ ფორმას (იხ. სურათი 4-22). ნორმალური ფორმა არის რელაციების მდგომარეობა, რომელიც მოითხოვს ატრიბუტებს (ან ფუნქციურ დამოკიდებულებებს) შორის ურთიერთდამოკიდებულებასთან დაკავშირებით გარკვეული წესების დაცვას. ეს ნორმალური ფორმებია:

1. *პირველი ნორმალური ფორმა* - ამოღებულია ყველა მრავალმნიშვნელოვანი ატრიბუტი (რომელსაც განმეორებად ჯგუფებს უწოდებენ), ამიტომ ცხრილის თითოეული სტრიქონისა და სვეტის გადაკვეთაზე არის ერთი მნიშვნელობა (შესაძლოა null) (როგორც სურათი 4-2 ბ).
2. *მეორე ნორმალური ფორმა* - ამოღებულია ნებისმიერი ნაწილობრივი ფუნქციური დამოკიდებულება (ანუ, არაგასაღები ატრიბუტების იდენტიფიცირება ხდება მთლიანი პირველად გასაღებით).
3. *მესამე ნორმალური ფორმა* - ყველა ტრანზიტული დამოკიდებულება ამოღებულია (მაგ., არაგასაღები ატრიბუტების იდენტიფიცირება ხდება მხოლოდ პირველადი გასაღებით).

4. *ბოის-კოდის ნორმალური ფორმა* - ყველა დარჩენილი ანომალია, რომელიც ფუნქციონალური დამოკიდებულების შედეგად წარმოიშვა, ამოღებულია (რადგან ერთიდაიმავე არაგასაღები ატრიბუტისთვის არსებობდა ერთზე მეტი შესაძლო პირველადი გასაღები).
5. *მეოთხე ნორმალური ფორმა* - ნებისმიერი მრავალჯერადი დამოკიდებულება ამოღებულია.
6. *მეხუთე ნორმალური ფორმა* - ამოღებულია დარჩენილი ანომალიები.

FIGURE 4-22 **Steps in normalization**



ფუნქციური დამოკიდებულებები და გასაღებები

ბოის-კოდის ნორმალურ ფორმამდე ნორმალიზება ემყარება ფუნქციური დამოკიდებულებების ანალიზს. ფუნქციონალური დამოკიდებულება არის შეზღუდვა ორ ატრიბუტს ან ატრიბუტის ორ ნაკრებს შორის. ნებისმიერი R დამოკიდებულებისათვის (რელაციისათვის), ატრიბუტი B ფუნქციურად არის დამოკიდებული A ატრიბუტზე, თუ A-ს ყველა მოქმედი ეგზემპლარისათვის, A-ს მნიშვნელობა ცალსახად განსაზღვრავს B-ს მნიშვნელობას. B–ს ფუნქციონალური დამოკიდებულება A–ზე გამოსახულია ისრით, შემდეგშნაირად: **A → B**. ფუნქციონალური დამოკიდებულება არ არის მათემატიკური დამოკიდებულება: B არ შეიძლება გამოითვალოს A– დან, უფრო მეტიც, თუ A–ს მნიშვნელობა ცნობილია, შეიძლება არსებობდეს მხოლოდ ერთი მნიშვნელობა B. ასევე ატრიბუტი შეიძლება ფუნქციურად იყოს დამოკიდებული ორი (ან მეტი) ატრიბუტის კომბინაციაზე. მაგალითად, განვიხილოთ დამოკიდებულება EMP COURSE (EmpID, CourseTitle, DateCompleted) ნაჩვენებია დიაგრამა 4-7. ამ რელაციაში ფუნქციურ დამოკიდებულებას წარმოვადგენთ შემდეგნაირად:

EmpID, CourseTitle → DateCompleted

მძიმით EmpID- სა და CourseTitle- ს აღინიშნება ლოგიკური „**და**“ ოპერატორი, რადგან DateCompleted ფუნქციურად დამოკიდებულია EmpID- ზე და CourseTitle- ზე.

ამ განაცხადში ფუნქციონალური დამოკიდებულება გულისხმობს, რომ კურსის დასრულების თარიღი განისაზღვრება დასაქმებულის ვინაობით და კურსის სახელწოდებით. ფუნქციური დამოკიდებულების ტიპიური მაგალითებია შემდეგი:

1. ***SSN → Name, Address, Birthdate*** (SSN → სახელი, მისამართი, დაბადების თარიღი) - პირის სახელი, მისამართი და დაბადების თარიღი ფუნქციურად არის დამოკიდებული ამ პირის პირად ნომერზე (სხვა სიტყვებით რომ ვთქვათ, თითოეული SSN- სთვის შეიძლება იყოს მხოლოდ ერთი სახელი, ერთი მისამართი და ერთი დაბადების დღე).
2. ***VIN → Make, Model, Color*** (VIN → მარკა, მოდელი, ფერი) - ავტომობილის მარკა, მოდელი და ორიგინალი ფერი ფუნქციურად დამოკიდებულია ავტომობილის საიდენტიფიკაციო ნომერზე (როგორც ზემოთ, თითოეულ VIN – სთან შეიძლება იყოს მხოლოდ ერთი მნიშვნელობის მარკა, მოდელი და ფერი).
3. ***ISBN → Title, FirstAuthorName, Publisher*** (ISBN → სათაური, პირველი ავტორის სახელი, გამომცემელი) - წიგნის სათაური, პირველი ავტორის სახელი და გამომცემელი ფუნქციურად დამოკიდებულია წიგნის საერთაშორისო სტანდარტული წიგნის ნომერზე (ISBN).

დეტერმინანტი

ატრიბუტს ისრის მარცხენა მხარეს ფუნქციურ დამოკიდებულებაში განმსაზღვრელს ანუ *დეტერმინატორს* უწოდებენ. წინა სამ მაგალითში SSN, VIN და ISBN დეტერმინატორია. EMP COURSE რელაციაში (სურათი 4-7), დეტერმინატორია EmpID და CourseTitle.

კანდიდატი გასაღები

კანდიდატი გასაღები არის ატრიბუტი, ან ატრიბუტების კომბინაცია, რომელიც ცალსახად განსაზღვრავს სტრიქონს რელაციაში. კანდიდატი გასაღები უნდა აკმაყოფილებდეს შემდეგ თვისებებს, რომლებიც ადრე განხილული ექვსი თვისების ქვესიმრავლეს წარმოადგენს:

1. *უნიკალური იდენტიფიკაცია* - თითოეული სტრიქონისათვის გასაღების მნიშვნელობამ ცალსახად უნდა განსაზღვროს ეს მწკრივი. ეს თვისება გულისხმობს, რომ თითოეული არაგასარები ატრიბუტი ფუნქციურად დამოკიდებულია ამ გასაღებაზე.
2. *არასიჭარბე* - გასაღების არცერთი ატრიბუტის წაშლა არაა შესაძლებელი უნიკალური იდენტიფიკაციის თვისების დარღვევის გარეშე.

გამოვიყენოთ წინა განმარტება წინ აღწერილ ორ რელაციაში კანდიდატი გასაღებების დასადგენად. EMPLOYEE1 რელაციას (სურათი 4-1) აქვს შემდეგი სქემა: EMPLOYEE1(EmpID, Name, DeptName, Salary). EmpID ერთადერთი დეტერემინატორია ამ მიმართებაში. ყველა სხვა ატრიბუტი ფუნქციურად დამოკიდებულია EmpID- ზე. ამიტომ, EmpID არის კანდიდატი

გასაღები და (რადგან სხვა კანდიდატი გასაღებები არ არსებობს) ასევე არის ძირითადი გასაღები.

წარმოვადგინოთ რელაციის ფუნქციური დამოკიდებულებები 4-23 ნახაზზე ნაჩვენები აღნიშვნების გამოყენებით. დიაგრამა 4-23a გვიჩვენებს EMPLOYEE1-ის ასახვას. ფიგურაში ჰორიზონტალური ხაზი ასახავს ფუნქციურ დამოკიდებულებებს. ვერტიკალური ხაზი ჩამოდის პირველადი გასაღებიდან (EmpID) და უერთდება ამ ხაზს. ვერტიკალური ისრები შემდეგში მიუთითებს თითოეულ არაგასაღებ ატრიბუტზე, რომლებიც ფუნქციურად დამოკიდებულია პირველად გასაღებზე.

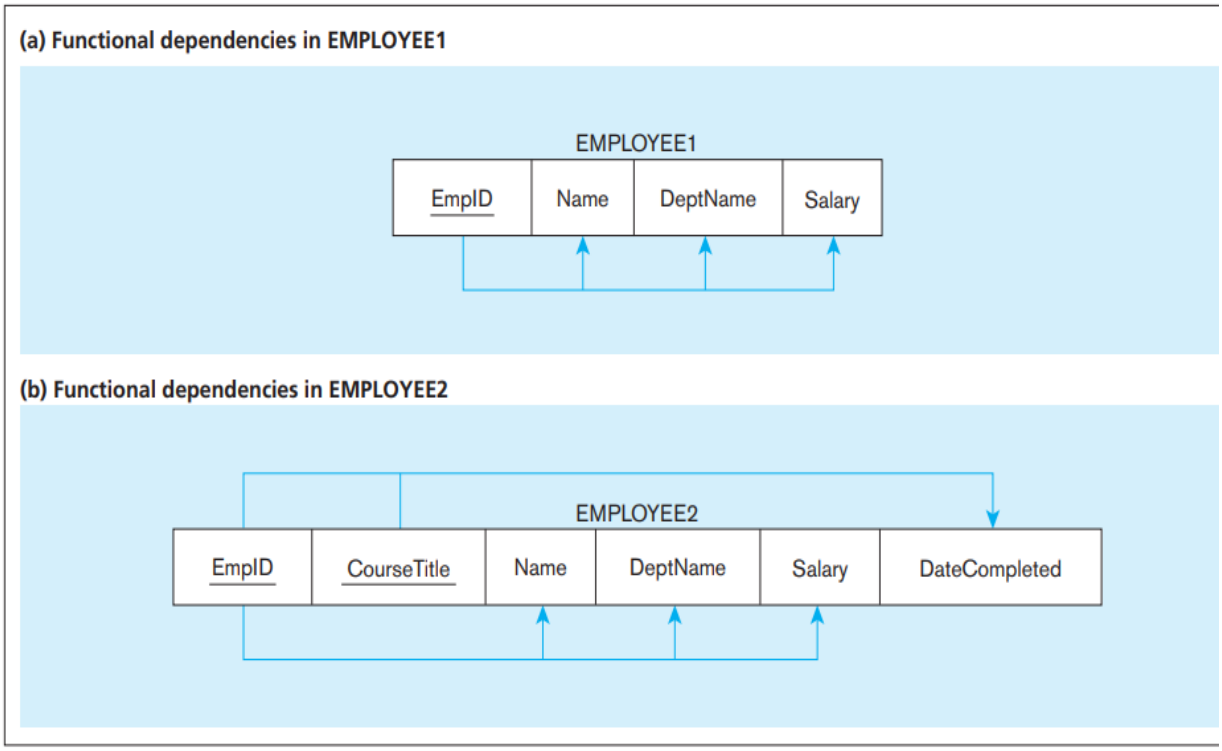


FIGURE 4-23 Representing functional dependencies

რელაციისათვის EMPLOYEE2 (სურათი 4-2b), გასათვალისწინებელია, რომ (განსხვავებით EMPLOYEE1) EmpID ცალსახად არ განსაზღვრავს სტრიქონს ამ რელაციაში. მაგალითად, EmpID-ის ნომერზე ცხრილში არის ორი სტრიქონი. ამ რელაციაში არსებობს ორი ტიპის ფუნქციური დამოკიდებულება:

1. EmpID → Name, DeptName, Salary
2. EmpID, CourseTitle → DateCompleted

ფუნქციონალური დამოკიდებულებები მიუთითებს, რომ EmpID და CourseTitle კომბინაცია არის ერთადერთი კანდიდატი გასარები (და, შესაბამისად, ძირითადი გასაღები) EMPLOYEE2-ისთვის. სხვა სიტყვებით რომ ვთქვათ, EMPLOYEE2– ის პირველადი გასაღები არის კომპოზიტიური გასაღები. არც EmpID და არც CourseTitle არ განსაზღვრავს სტრიქონს ამ რელაციაში და, შესაბამისად, თავისთავად არ შეიძლება იყოს კანდიდატი გასაღები. თუ გადავხედავთ მონაცემებს ნახაზზე 4-2 ბ, დავრწმუნდებით, რომ EmpID- ისა და CourseTitle- ის კომბინაცია ცალსახად განსაზღვრავს EMPLOYEE2- ის თითოეულ სტრიქონს. ჩვენ

წარმოვადგენთ ამ რელაციაში ფუნქციონალურ დამოკიდებულებებს ნახაზზე 4-23b. გაითვალისწინეთ, რომ DateCompleted ერთადერთი ატრიბუტია, რომელიც ფუნქციურად დამოკიდებულია სრულ პირველად გასაღებზე, რომელიც შედგება ატრიბუტებისაგან EmpID და CourseTitle.

დეტერმინანტებსა და კანდიდატ გასაღებებს შორის ურთიერთდამოკიდებულების შეჯამება შეგვიძლია შემდეგნაირად: კანდიდატი გასაღები ყოველთვის არის დეტერმინანტი, ხოლო დეტერმინანტი შეიძლება იყოს ან არ იყოს კანდიდატი გასაღები. მაგალითად, EMPLOYEE2-ში, EmpID არის დეტერმინანტი, მაგრამ არა კანდიდატი გასაღები. კანდიდატი გასაღები არის დეტერმინანტი, რომელიც ცალსახად განსაზღვრავს დანარჩენ (არაგასაღებ) ატრიბუტებს რელაციაში. დეტერმინანტი შეიძლება იყოს კანდიდატი გასაღები (მაგალითად, EmpID in EMPLOYEE1), კომპოზიტიური კანდიდატი გასაღების ნაწილი (მაგალითად, EmpID in EMPLOYEE2), ან არაგასაღები ატრიბუტი.

ნორმალიზაციის მაგალითი

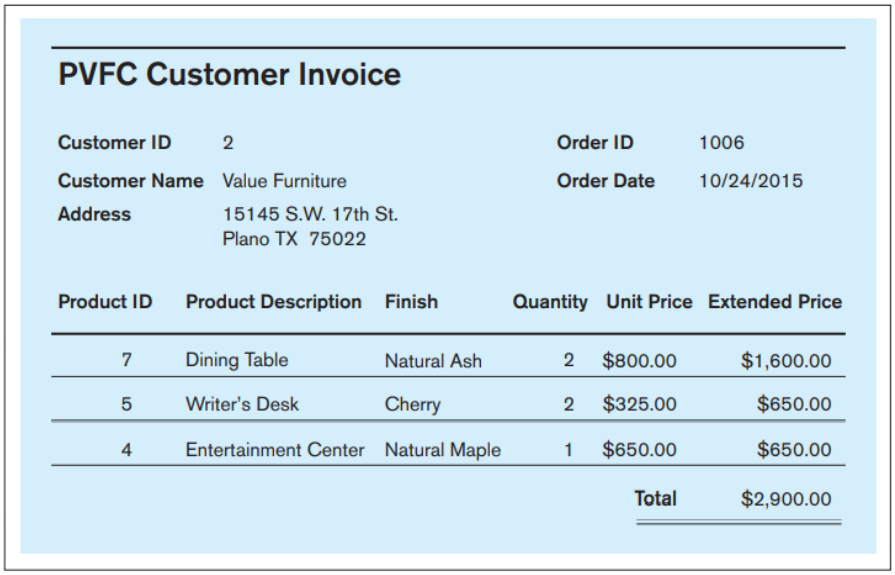
ფუნქციური დამოკიდებულებების და გასაღებების შესწავლის შემდეგ, შესაძლებელია აღვწეროთ და ავსახოთ ნორმალიზაციის ბიჯები. თუ EER მონაცემთა მოდელის ტრანსფორმაცია მოხდა მონაცემთა ბაზის რელაციების ყოვლისმომცველ ნაკრებად, საჭიროა თითოეული მათგანის ნორმალიზება. სხვა შემთხვევებში, როდესაც ლოგიკური მონაცემების მოდელი იქმნება მომხმარებლის ინტერფეისებიდან, როგორიცაა ეკრანული ასახვები, ფორმები და ანგარიშები, შესაძლოა საჭირო გახდეს რელაციური სტრუქტურის შექმნა თითოეული სამომხმარებლო ინტერფეისისათვის და შემდეგომში მოხდეს ამ რელაციების ნორმალიზება.

ილუსტრაციისთვის ვიყენებთ მომხმარებელთა ზედნადებს Pine Valley ავეჯის კომპანიისგან (იხ. სურათი 4-24.)

ბიჯი 0: წარმოადგინეთ ხედი ცხრილის სახით

პირველი ბიჯი (ნორმალიზაციის წინმსწრები) წარმოადგენს მომხმარებლის ხედვის (ამ შემთხვევაში, ინვოისის) წარმოდგენას ერთი ცხრილის ან ატრიბუტების (სვეტის სათაურების სახით) შემცველი რელაციის სახით. ნიმუშის მონაცემები უნდა ჩაიწეროს ცხრილის სტრიქონებში, მათ შორის ნებისმიერი განმეორებითი ჯგუფი, რომლებიც მოცემულია მონაცემებში. ინვოისის ცხრილი ნაჩვენებია ნახაზზე 4-25. გავითვალისწინოთ, რომ მეორე შეკვეთის მონაცემები (OrderID 1007) მოცემულია ნახაზზე 4-25 ამ მონაცემების სტრუქტურის უფრო მეტად განსამარტად.

FIGURE 4-24 Invoice (Pine Valley Furniture Company)



PVFC Customer Invoice

Customer ID	2	**Order ID**	1006
Customer Name	Value Furniture	**Order Date**	10/24/2015
Address	15145 S.W. 17th St. Plano TX 75022		

Product ID	Product Description	Finish	Quantity	Unit Price	Extended Price
7	Dining Table	Natural Ash	2	$800.00	$1,600.00
5	Writer's Desk	Cherry	2	$325.00	$650.00
4	Entertainment Center	Natural Maple	1	$650.00	$650.00
				Total	$2,900.00

ბიჯი 1: პირველ ნორმალურ ფორმად გარდაქმნა

რელაცია პირველ ნორმალურ ფორმაშია (1nF), თუ გამოიყენება ორივე შემდეგი შეზღუდვა:

1. რელაციაში არ არის განმეორებითი ჯგუფები (ამრიგად, ცხრილი თითოეული სტრიქონისა და სვეტის გადაკვეთაზე არის ერთი ფაქტი).
2. განსაზღვრულია პირველადი გასაღები, რომელიც ცალსახად განსაზღვრავს თითოეულ სტრიქონს რელაციაში.

განმეორებადი ჯგუფების ამოღება

როგორც ვხედავთ, ინვოისის მონაცემები 4-25 ნახაზზე შეიცავს განმეორებად ჯგუფს თითოეული პროდუქტისთვის, რომელიც მოცემულია კონკრეტული შეკვეთის მიხედვით. ამრიგად, OrderID 1006 შეიცავს სამ განმეორებით ჯგუფს, რომლებიც შეესაბამება ამ შეკვეთის სამ პროდუქტს.

ზემოთ ვაჩვენეთ, თუ როგორ უნდა ამოვიღოთ განმეორებითი ჯგუფები ცხრილიდან, მონაცემთა შესაბამისი მნიშვნელობების შევსებით ცხრილის ადრე არსებულ ცარიელ უჯრედებში (იხ. ნახაზები 4-2 ა და 4-2 ბ). ინვოისის ცხრილისთვის ამ პროცედურის გამოყენება იძლევა ახალ რელაციას (სახელად INVOICE), რომელიც ნაჩვენებია 4-26 ნახაზზე.

პირველადი გასაღების შერჩევა

INVOICE -ში ოთხი დეტერმინატორია და მათი ფუნქციური დამოკიდებულება შემდეგია:

OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress
CustomerID → CustomerName, CustomerAddress
ProductID → ProductDescription, ProductFinish, ProductStandardPrice OrderID, ProductID → OrderedQuantity

ამ შემთხვევაში ინფორმაცია ფუნქციური დამოკიდებულებების შესახებ ბიზნესის წესების შესაბამისად ორგანიზაციიდან მოდის (Pine Valley ავეჯის კომპანიის ბიზნესის ბუნების შესწავლის შედეგად). ასევე შეგვიძლია ვნახოთ, რომ ნახაზზე 4-26 მოცემული მონაცემები არ არღვევს რომელიმე ამ ფუნქციურ დამოკიდებულებას. მაგრამ, რადგან ამ ცხრილის ყველა შესაძლო სტრიქონს ვერ ვხედავთ, ვერ ვიქნებით დარწმუნებული, რომ არ იქნებოდა ისეთი ინვოისი, რომელიც დაარღვევდა ერთ-ერთ ამ ფუნქციურ დამოკიდებულებას. ამრიგად, დამოკიდებული უნდა ვიყოთ ორგანიზაციის წესების ჩვენს გაგებაზე.

FIGURE 4-25 **INVOICE data (Pine Valley Furniture Company)**

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

როგორც ხედავთ, INVOICE– ის ერთადერთი კანდიდატი გასაღები არის კომპოზიტური გასაღები, რომელიც შედგება ატრიბუტების OrderID და ProductID (რადგან ამ ატრიბუტების მნიშვნელობების ნებისმიერი კომბინაციისთვის ცხრილში მხოლოდ ერთი სტრიქონია). შესაბამისად, OrderID და ProductID ხაზგასმულია ნახატზე 4-26, რაც მიუთითებს იმაზე, რომ ისინი ქმნიან პირველად გასაღებს.

პირველადი გასაღების ფორმირებისას ფრთხილად უნდა ვიყოთ, რომ არ შევიტანოთ ზედმეტი (და, შესაბამისად, არასაჭირო) ატრიბუტები. ამრიგად, მიუხედავად იმისა, რომ CustomerID არის INVOICE-ისიდენტიფიკატორი, იგი არ შედის პირველადი გასაღების ნაწილად, რადგან გასაღების ყველა ატრიბუტი იდენტიფიცირებულია OrderID და ProductID კომბინაციით.

დიაგრამა, რომელიც აჩვენებს ამ ფუნქციურ დამოკიდებულებებს INVOICE რელაციისთვის ნაჩვენებია ნახაზზე 4-27. ეს დიაგრამა წარმოადგენს INVOICE- ის ყველა ატრიბუტის ჰორიზონტალურ ჩამონათვალს, პირველადი გასაღები ატრიბუტების ხაზგასმით (OrderID და ProductID). გავითვალისწინოთ, რომ ერთადერთი ატრიბუტი, რომელიც დამოკიდებულია სრულ გასაღებზე, არის OruledQuantity. ყველა სხვა ფუნქციური დამოკიდებულება ან ნაწილობრივია, ან ტრანზიტული დამოკიდებულება (ორივე განისაზღვრება შემდეგში).

ანომალიები 1nf- ში

მიუხედავად იმისა, რომ განმეორებითი ჯგუფები ამოღებულია, 4-26 ნახაზის მონაცემები კვლავ შეიცავს მნიშვნელოვან სიჭარბეს. მაგალითად, CustomerID, CustomerName და CustomerAddress Value ავეჯისთვის ჩაიწერება ცხრილში სამ სტრიქონში (მინიმუმ). ამ სიჭარბის შედეგად, ცხრილში მოცემული მონაცემებით მანიპულირებამ შეიძლება გამოიწვიოს ისეთი ანომალიები, როგორიცაა შემდეგი:

1. ***ჩასმის ანომალია*** - ამ ცხრილის სტრუქტურით კომპანიას არ შეუძლია წარუდგინოს ახალი პროდუქტი (ვთქვათ, საუზმის მაგიდა ProductID 8-ით) და დაამატოს იგი მონაცემთა ბაზაში, სანამ არ იქნება პირველი შეკვეთა: ცხრილში ჩანაწერების დამატება არ შეიძლება ProductID–ისა და OrderID– ის გარეშე. კიდევ ერთი მაგალითი, თუ მომხმარებელი დარეკავს და ითხოვს მის OrderID 1007-ს სხვა პროდუქტის დამატებას, უნდა ჩაისვას ახალი სტრიქონი, რომელშიც უნდა განმეორდეს შეკვეთის თარიღი და მომხმარებლის ყველა ინფორმაცია. ეს იწვევს მონაცემთა ტირაჟირებას და მონაცემთა შეყვანის პოტენციურ შეცდომებს (მაგალითად, მომხმარებლის სახელი შეიძლება შეიტანონ შეცდომით).
2. ***წაშლის ანომალია*** - თუ მომხმარებელი დარეკავს და ითხოვს სასადილო მაგიდის წაშლას მისი OrderID 1006–დან, ეს სტრიქონი უნდა წაიშალოს რელაციიდან და ჩვენ ვკარგავთ ინფორმაციას ამ ნივთის მოპირკეთების (Natural Ash) და ფასის (800.00 აშშ დოლარი) შესახებ.
3. ***განახლების ანომალია*** - თუ Pine Valley Furniture (როგორც ფასის კორექტირების ნაწილი) ზრდის ერთერთი პროდუქტის ფასს (ProductID 4) ფასს 750,00 დოლარამდე, ეს ცვლილება უნდა ჩაიწეროს ამ ნივთის შემცველ ყველა სტრიქონში. (სურათი 4-26-ში ორი ასეთი რიგია.)

FIGURE 4-26 **INVOICE relation (1NF) (Pine Valley Furniture Company)**

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

ბიჯი 2: მეორე ნორმალურ ფორმაში გარდაქმნა

INVOICE რელაციაში ზედმეტი სიჭარბის (და შედეგად მიღებული ანომალიები) მოშორება შესაძლებელია მეორე ნორმალურ ფორმაში გარდაქმნით. რელაცია მეორე ნორმალურ ფორმაშია (2nF), თუ ის პირველ ნორმალურ ფორმაშია და არ შეიცავს ნაწილობრივ ფუნქციურ დამოკიდებულებებს. ნაწილობრივი ფუნქციონალური დამოკიდებულება არსებობს, როდესაც არაგასაღები ატრიბუტი ფუნქციურად დამოკიდებულია პირველადი გასაღების ნაწილზე (მაგრამ არა მთლიანზე). როგორც ვხედავთ, ნახაზზე 4-27-ზე არსებობს შემდეგი ნაწილობრივი დამოკიდებულებები:

OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress
ProductID → ProductDescription, ProductFinish, ProductStandardPrice

ამ ნაწილობრივი დამოკიდებულებების პირველ ნაწილში (მაგალითად) ნათქვამია, რომ შეკვეთის თარიღი ცალსახად განისაზღვრება შეკვეთის ნომრით და მას არანაირი კავშირი არ აქვს ProductID– სთან.

ნაწილობრივი დამოკიდებულებების მქონე რელაციის მეორე ნორმალურ ფორმაში გადასაყვანად საჭიროა შემდეგი ბიჯები:

1. შეიქმნას ახალი რელაციები პირველადი გასაღების თითოეული ატრიბუტისთვის (ან ატრიბუტების კომბინაციისთვის), რომელიც განსაზღვრავს ნაწილობრივ დამოკიდებულებას. ეს ატრიბუტი იქნებიან პირველადი გასაღები ახალ რელაციაში.
2. გადაადგილდეს არაგასაღები ატრიბუტები, რომლებიც მხოლოდ ამ პირველად ატრიბუტზეა (ატრიბუტებზე) დამოკიდებული ძველი რელაციიდან ახალ რელაციაში.

ამ ბიჯების შესრულების შედეგები INVOICE რელაცია ნაჩვენებია სურათი 4-28. ნაწილობრივი დამოკიდებულების მოხსნა იწვევს ორი ახალი რელაციის შექმნას: PRODUCT და CUSTOMER ORDER. INVOICE რელაციაში ახლა დარჩა მხოლოდ პირველადი გასაღებების ატრიბუტები (OrderID და ProductID) და OrteredQuantity, რაც ფუნქციურად დამოკიდებულია მთლიან გასაღებზე. ამ ურთიერთობას დავარქვათ ORDER LINE, რადგან ამ ცხრილის თითოეული სტრიქონი წარმოადგენს შეკვეთის ერთ სტრიქონს.

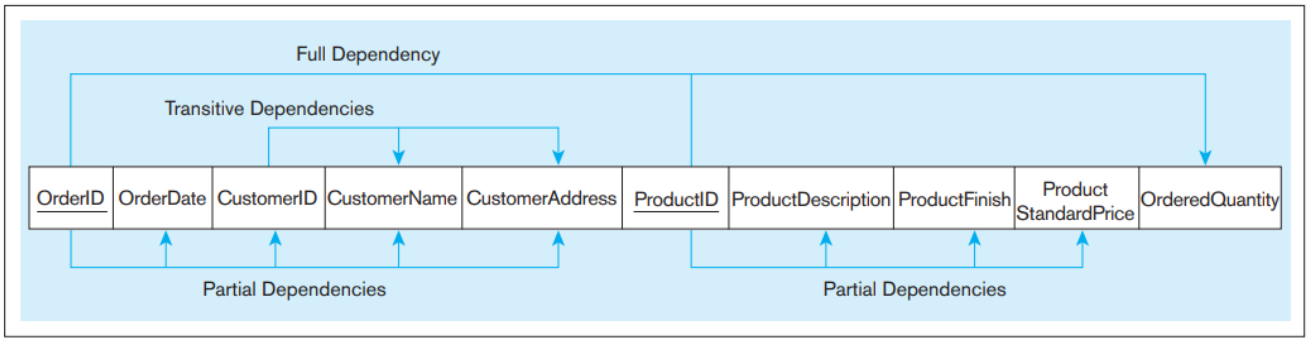


FIGURE 4-27 **Functional dependency diagram for INVOICE**

როგორც 4-28 ნახაზზეა მითითებული, რელაცია ORDER LINE და PRODUCT მესამე ნორმალური ფორმისაა. ამასთან, CUSTOMER ORDER შეიცავს ტრანზიტულ დამოკიდებულებებს და, შესაბამისად, (თუმცა მეორე ნორმალური ფორმაა) ჯერ არ არის მესამე ნორმალურ ფორმაში.

პირველ ნორმალურ ფორმაში მყოფი დამოკიდებულება მეორე ნორმალურ ფორმაში იქნება, თუ რომელიმე შემდეგი პირობებიდან ერთი მოქმედებს:

1. პირველადი გასაღები შედგება მხოლოდ ერთი ატრიბუტისგან (მაგალითად, ატრიბუტი ProductID რელაციაში PRODUCT, ნახაზი 4-28). განმარტების თანახმად, ასეთ რელაციაში ნაწილობრივი დამოკიდებულება არ შეიძლება იყოს.
2. რლლაციაში არ არსებობს არაგასაღები ატრიბუტები (ამრიგად, რელაციაში არსებული ყველა ატრიბუტი პირველადი გასაღების კომპონენტებია). ასეთ დამოკიდებულებაში ფუნქციური დამოკიდებულება არ არსებობს
3. ყველა არაგასაღების ატრიბუტი ფუნქციურად დამოკიდებულია პირველად გასაღებში შემავალი ატრიბუტების სრულ ნაკრებზე (მაგ., ატრიბუტი OrteredQuantity ORDER LINE მიმართებაში, სურათი 4-28).

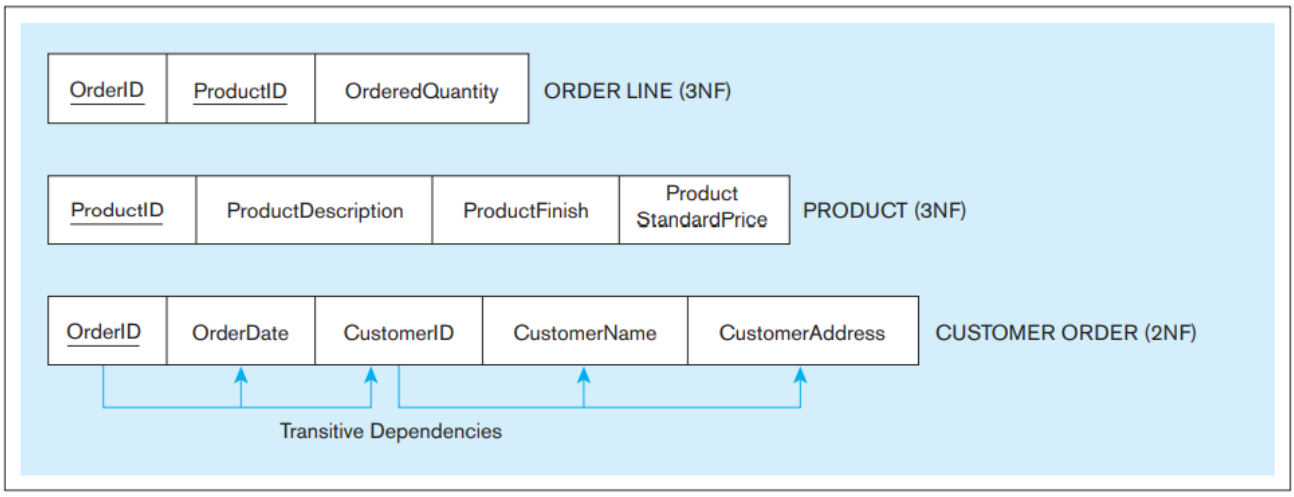


FIGURE 4-28 **Removing partial dependencies**

ბიჯი 3: მესამე ნორმალურ ფორმაში გარდაქმნა

რელაცია მესამე ნორმალურ ფორმაშია (3nF), თუ ის მეორე ნორმალურ ფორმაშია და ტრანზიტული დამოკიდებულებები არ არსებობს. რელაციაში ტრანზიტული დამოკიდებულება არის ფუნქციური დამოკიდებულება პირველადი გასაღებისა და ერთი ან მეტი არაგასაღებ ატრიბუტებს შორის, რომლებიც დამოკიდებულია პირველად გასაღებზე სხვა არაგასაღები ატრიბუტის საშუალებით. მაგალითად, CUSTOMER ORDER რელაციაში ორი ტრანზიტული დამოკიდებულებაა, ნაჩვენებია ნახაზზე 4-28:

OrderID → CustomerID → CustomerName
OrderID → CustomerID → CustomerAddress

სხვა სიტყვებით რომ ვთქვათ, მომხმარებლის სახელი და მისამართი ცალსახად იდენტიფიცირებული CustomerID– ის მიერ, მაგრამ CustomerID არ არის პირველადი გასაღების ნაწილი (როგორც ადრე აღვნიშნეთ).

ტრანზიტული დამოკიდებულება ქმნის არასაჭირო სიჭარბეს, რამაც შეიძლება გამოიწვიოს ადრე განხილული ანომალიების ტიპი. მაგალითად, CUSTOMER ORDER ტრანზიტული დამოკიდებულება (სურათი 4-28) მოითხოვს, რომ მომხმარებლის სახელი და მისამართი ხელახლა შეიტანოს ყოველ ჯერზე, როდესაც მომხმარებელი წარუდგენს ახალ შეკვეთას, მიუხედავად იმისა, რამდენჯერ არის შეყვანილი მანამდე.

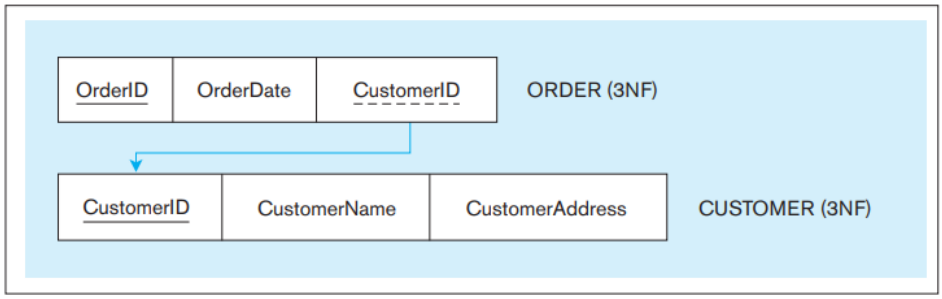
ტრანზიტული დამოკიდებულების ამოღება

რელაციიდან ტრანზიტული დამოკიდებულების ამოღება მარტივადაა შესაძლებელი სამსაფეხურიანი პროცედურის საშუალებით:

1. თითოეული არაგასაღები ატრიბუტისთვის (ან ატრიბუტების სიმრავლისთვის), რომელიც დეტერმინანტია რელაციაში, შეიქმნას ახალი რელაცია. ეს ატრიბუტი (ან ატრიბუტების ნაკრები) ხდება ახალი რელაციის პირველადი გასაღები.
2. მოხდეს ყველა ატრიბუტის, რომლებიც ფუნქციურად არის დამოკიდებული მხოლოდ ახალი რელაციის პირველად გასაღებზე, გადატანა ძველიდან ახალ რელაციაში.
3. ატრიბუტი, რომელიც წარმოადგენს პირველად გასაღებს ახალ რელაციაში, დარჩეს როგორც გარე გასაღები, რომელიც საშუალებას მოგვცენს დავაკავშიროთ ორი რელაცია.

ამ ნაბიჯების გამოყენების შედეგები CUSTOMER ORDER რელაციასთან დაკავშირებით ნაჩვენებია დიაგრამა 4-29. შეიქმნა ახალი რელაცია სახელწოდებით CUSTOMER, ტრანზიტული დამოკიდებულების კომპონენტების მისაღებად. დეტერმინანტი CustomerID ხდება ამ რელაციის პირველადი გასაღები და ატრიბუტები CustomerName და CustomerAddress გადადის რელაციაში CUSTOMER ORDER ეწოდა ORDER და ატრიბუტი CustomerID რჩება როგორც უცხოური გასაღები ამ მიმართებაში. ეს საშუალებას გვაძლევს შევასრულოთ შეკვეთა მომხმარებელთან, რომელმაც შეკვეთა წარადგინა. როგორც 4-29 გრაფიკზეა მითითებული, ეს რელაციები ახლა მესამე ნორმალურ ფორმაშია.

FIGURE 4-29 Removing transitive dependencies



INVOICE ხედვის მონაცემების ნორმალიზებამ შექმნა ოთხი რელაცია მესამე ნორმალურ ფორმაში: CUSTOMER, PRODUCT, ORDER და ORDER LINE. ამ ოთხი რელაციის და მათი ასოციაციების ამსახველი რელაციური სქემა (რომელიც შექმნილია Microsoft Visio– ს გამოყენებით) ნაჩვენებია დიაგრამა 4-30. გავითვალისწინოთ, რომ CustomerID არის გარე

გასაღები ORDER- ში, ხოლო OrderID და ProductID არის გარე გასაღებები ORDER LINE- ში (გარე გასაღებები ნაჩვენებია Visio–ში ლოგიკური, მაგრამ არა კონცეპტუალური მონაცემების მოდელისთვის). ასევე გავითვალისწინოთ, რომ მინიმალური კარდინალურობები ნაჩვენებია კავშირებზე, მიუხედავად იმისა, რომ ნორმალიზებული შედეგები არ აჩვენებს იმას, თუ რა უნდა იყოს მინიმალური კარდინალურობები.

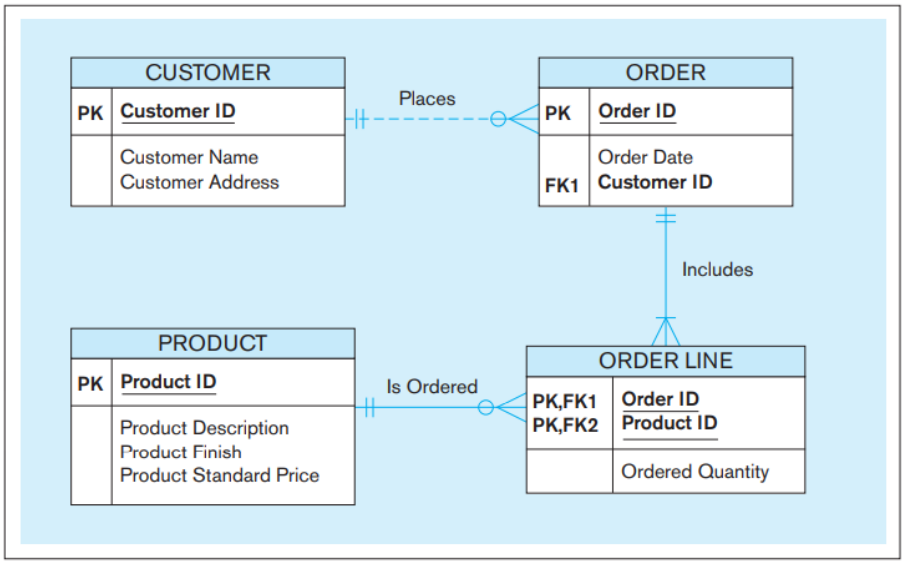


FIGURE 4-30 Relational schema for INVOICE data (Microsoft Visio notation)

დეტერმინანტები და ნორმალიზაცია

ჩვენ ვაჩვენეთ ნორმალიზაცია 3NF–ის მეშვეობით ეტაპობრივად. თუმცა არსებობს მარტივი მალსახმობიც. თუ გადახედავთ ოთხი დეტერმინანტის საწყის კომპლექტს და მასთან დაკავშირებული ფუნქციურ დამოკიდებულებებს ინვოისის მომხმარებლის ხედიდან, თითოეული მათგანი შეესაბამება ერთ–ერთ რელაციას დიაგრამა 4-30. თითოეული დეტერმინანტი არის რელაციის პირველადი გასაღები, ხოლო თითოეული რელაციის არაგასაღები არის ის ატრიბუტები, რომლებიც ფუნქციურად დამოკიდებულია თითოეულ დეტერმინანტზე. არსებობს დახვეწილი, მაგრამ მნიშვნელოვანი განსხვავება: იმის გამო, რომ OrderID განსაზღვრავს CustomerID, CustomerName და CustomerAddress და CustomerID განსაზღვრავს მის დამოკიდებულ ატრიბუტებს, CustomerID ხდება გარე გასაღები ORDER რელაციაში, სადაც წარმოდგენილია CustomerName და CustomerAddress. თუ შესაძლებელია დეტერმინანტების განსაზღვრა, რომლებსაც არ აქვთ გადამფარი დამოკიდებული ატრიბუტები, შესაძლებელია კავშირის განსაზღვრაც. ამრიგად, შესაძლებელია ნორმალიზაციის ეტაპობრივად განხორციელება, როგორც ილუსტრირებულია მოყვანილ მაგალითში „ინვოისზე“, ან შესაძლებელია შეიქმნას რელაციები პირდაპირ 3NF–ში დეტერმინანდების ფუნქციური დამოკიდებულებიდან.

ბიჯი 4: შემდგომი ნორმალიზაცია

0–დან 3–ის ჩათვლთ ბიჯების დასრულების შემდეგ, ყველა არაგასაღები ატრიბუტი დამოკიდებული იქნება პირველადი გასაღების, მთლიანად პირველად გასაღებზე და სხვა

არაფერზე, გარდა ძირითადი გასაღებისა. სინამდვილეში, ნორმალური ფორმები არის ფუნქციური დამოკიდებულების წესები და, ამრიგად, შედეგია დეტერმინანტებისა და მათთან დაკავშირებული არაგასაღებების მოძებნა. ზემოთ აღწერილი ბიჯები ხელს უწყობს თითოეული დეტერმინანტისა და მასთან დაკავშირებული არაგასაღები ატრიბუტების შემცველი რელაციების შექმნას.

ნორმალიზაციის შესახებ განხილვის დასაწყისიდანვე გავიხსენოთ, რომ 3NF–ს გარდა დამატებით განმარტებულია სხვა ნორმალური ფორმებიც.

რელაციების შერწყმა

ჩვენ აღვწერეთ, თუ როგორ უნდა გარდაქიმნას EER დიაგრამები რელაციებად. ეს ტრანსფორმაცია ხდება მაშინ, როდესაც ჩვენ ზემოდან ვიღებთ მონაცემთა მოთხოვნების დაწვრილებითი ანალიზის შედეგებს და ვიწყებთ მათი სტრუქტურირებას მონაცემთა ბაზაში განსახორციელებლად. შემდეგ აღვწერეთ, თუ როგორ უნდა გადავამოწმოთ მიღებული რელაციები ისე, რომ აკმაყოფილებდნენ მესამე (ან უფრო მაღალი) ნორმალური ფორმას და საჭიროების შემთხვევაში შევასრულოთ ნორმალიზაციის ბიჯები.

ლოგიკური დიზაინის პროცესის ფარგლებში, ნორმალიზებული რელაციები შეიძლება შეიქმნას რიგი ცალკეული EER დიაგრამებიდან და (შესაძლოა) მომხმარებლის სხვა ხედვებისგან (მაგ., შეიძლება არსებობდეს მონაცემთა ბაზის ქვედა ან პარალელური განვითარების ქმედება ორგანიზაციის სხვადასხვა სფეროებისთვისაც) როგორც ზემოდან ქვემოთ). მაგალითად, ნორმალიზაციის საილუსტრაციოდ წინა განყოფილებაში გამოყენებული ინვოისის გარდა, შეიძლება არსებობდეს შეკვეთის ფორმა, ანგარიშის ბალანსის ანგარიში, წარმოების მარშრუტიზაცია და მომხმარებლის სხვა ხედვები, რომელთაგან თითოეული ნორმალიზებულია ცალკე. ადრე განხილული მონაცემთა ბაზის სამსქემიანი არქიტექტურა ხელს უწყობს მონაცემთა ბაზის განვითარების პროცესების ერთდროულად გამოყენებას. სინამდვილეში, საშუალო და მსხვილი ორგანიზაციების უმეტესობა ახდენს მრავალი დამოუკიდებელი სისტემების განვითარებას, რასაც გარკვეულ მომენტში შეიძლება დასჭირდეს გაერთიანება მონაცემთა ბაზის შესაქმნელად. შედეგიად ამ სხვადასხვა პროცესებით წარმოქმნილი ზოგიერთი რელაცია შეიძლება ზედმეტი იყოს; ანუ ისინი შეიძლება ისინი ეხებოდეს ერთიდაიმავე ობიექტებს. ასეთ შემთხვევებში, უნდა მოხდეს მათი გაერთიანება სიჭარბის აღმოსაფხვრელად. აღვწეროთ ეს პროცესი - რელაციების შერწყმა (ასევე უწოდებენ ხედვის ინტეგრაციას). ეს მნიშვნელოვანია სამი მიზეზის გამო:

1. მსხვილ პროექტებზე მუშაობა ლოგიკური დიზაინის დროს რამდენიმე ქვეჯგუფს აერთიანებს, ამიტომ ხშირად ხდება რელაციების შერწყმის საჭიროება.
2. არსებული მონაცემთა ბაზის ახალი ინფორმაციის მოთხოვნებთან ინტეგრირება ხშირად იწვევს სხვადასხვა ხედვების ინტეგრირების აუცილებლობას.
3. სასიცოცხლო ციკლის განმავლობაში შეიძლება წარმოიშვას მონაცემთა ახალი მოთხოვნები, ამიტომ საჭიროა ნებისმიერი ახალი რელაციის შერწყმა უკვე შემუშავებულთან.

მაგალითი

დავუშვათ, რომ მომხმარებლის ხედვის მოდელირება წარმოქმნის შემდეგ 3NF რელაციას:

EMPLOYEE1(EmployeeID, Name, Address, Phone)

მეორე მომხმარებლის ხედვის მოდელობამ შეიძლება მოგვცეს შემდეგი რელაცია:

EMPLOYEE2(EmployeeID, Name, Address, Jobcode, NoYears)

იმის გამო, რომ ამ ორ რელაციას აქვს ერთი და იგივე პირველადი გასაღები (EmployeeID), ისინი სავარაუდოდ აღწერს ერთსა და იმავე ობიექტს და შეიძლება გაერთიანდეს ერთ რელაციაში. რელაციების შერწყმის შედეგია შემდეგი რელაცია:

EMPLOYEE(EmployeeID, Name, Address, Phone, Jobcode, NoYears)

გავითვალისწინოთ, რომ ატრიბუტი, რომელიც ორივე რელაციაში ჩანს (მაგ., სახელი ამ მაგალითში) შერწყმულ რელაციაში მხოლოდ ერთხელ გამოჩნდება.

ინტეგრაციის პრობლემების მიმოხილვა

რელაციების ინტეგრირებისას, როგორც წინა მაგალითში, მონაცემთა ბაზის ანალიტიკოსმა უნდა გააცნობიეროს მონაცემთა მნიშვნელობა და მომზადებული უნდა იყოს ნებისმიერი პრობლემის გადასაჭრელად, რომელიც შეიძლება წარმოიშვას ამ პროცესში. ამ ნაწილში ჩვენ აღწერს და მოკლედ ასახავს ოთხ პრობლემას, რომლებიც წარმოიქმნება ინტეგრაციის თვალსაზრისით: სინონიმები, ჰომონიმები, ტრანზიტული დამოკიდებულებები და სუპერტიპის / ქვეტიპის კავშირები.

სინონიმები

ზოგიერთ სიტუაციაში, ორ (ან მეტ) ატრიბუტს შეიძლება ჰქონდეს განსხვავებული სახელები, მაგრამ იგივე მნიშვნელობა (მაგალითად, როდესაც ისინი აღწერენ არსის ერთსა და იმავე მახასიათებელს). ასეთ ატრიბუტებს სინონიმებს უწოდებენ. მაგალითად, EmployeeID და EmployeeNo შეიძლება იყოს სინონიმები. სინონიმების შემცველი რელაციების შერწყმისას, მომხმარებლებისგან უნდა მიიღოთ შეთანხმება (თუ შესაძლებელია) ატრიბუტის ერთ, სტანდარტიზებულ სახელზე და ამოღებულ იქნას სხვა სინონიმები (სხვა ალტერნატივაა მესამე სახელის არჩევა სინონიმების შესაცვლელად). მაგალითად, განვიხილოთ შემდეგი რელაციები:

STUDENT1(StudentID, Name)
STUDENT2(MatriculationNo, Name, Address)

ამ შემთხვევაში, ანალიტიკოსი ადასტურებს, რომ StudentID და MatriculationNo არის სტუდენტის პირადობის ნომრის სინონიმები და იდენტური ატრიბუტებია (კიდევ ერთი

შესაძლებლობა ის არის, რომ ეს ორივე კანდიდატი გასაღებია და მხოლოდ ერთი მათგანი უნდა იყოს არჩეული როგორც პირველადი გასაღები). ერთი შესაძლო გადაწყვეტა იქნება ორი ატრიბუტის სახელიდან ერთის სტანდარტიზაცია, მაგალითად StudentID. კიდევ ერთი ვარიანტია გამოვიყენოთ ახალი ატრიბუტის სახელი, მაგალითად StudentNo, ორივე სინონიმის შესაცვლელად. ამ უკანასკნელი მიდგომის ვარაუდით, ორი რელაციია შერწყმა შემდეგ შედეგს გამოიღებს:

STUDENT(StudentNo, Name, Address)

ხშირად, როდესაც არსებობს სინონიმები, საჭიროა მონაცემთა ბაზის ზოგიერთ მომხმარებელს მისცეს საშუალება მოახდინოს ერთიდაიმავე მონაცემებზე მითითება სხვადასხვა სახელით. მომხმარებლისთვის შეიძლება საჭირო გახდეს ნაცნობი სახელების გამოყენება, რომლებიც შეესაბამება ორგანიზაციის მათ ნაწილში არსებულ ტერმინოლოგიას. მეტსახელი (ფსევდონიმი) - არის ატრიბუტისთვის გამოყენებული ალტერნატიული სახელი. მონაცემთა ბაზის მართვის მრავალი სისტემა საშუალებას იძლევა განისაზღვროს მეტსახელი, რომელიც შეიძლება გამოყენებულ იქნას პირველადი ატრიბუტის ჭდის ნაცვლად.

ომონიმები

ატრიბუტის სახელს, რომელსაც შეიძლება ჰქონდეს ერთზე მეტი მნიშვნელობა, ჰომონიმს უწოდებენ. მაგალითად, ტერმინი *account* (ანგარიში) შეიძლება ეხებოდეს ბანკის მიმდინარე ანგარიშს, შემნახველ ანგარიშს, სესხის ანგარიშს ან სხვა სახის ანგარიშს (და შესაბამისად, ანგარიში გულისხმობს სხვადასხვა მონაცემს, მისი გამოყენების შესაბამისად)

რელაციების შერწყმისას უნდა მოხდეს ჰომონიმების მოძიება. განვიხილოთ შემდეგი მაგალითი:

.

STUDENT1(StudentID, Name, Address)
STUDENT2(StudentID, Name, PhoneNo, Address)

მომხმარებლებთან დისკუსიების დროს, ანალიტიკოსმა შეიძლება აღმოაჩინოს, რომ ატრიბუტი მისამართი STUDENT1- ში ეხება სტუდენტის უნივერსიტეტის მისამართს, ხოლო STUDENT2– ში იგივე ატრიბუტი ეხება სტუდენტის მუდმივ (ან სახლის) მისამართს. ამ კონფლიქტის მოსაგვარებლად, ალბათ დაგვჭირდება ახალი ატრიბუტების სახელების შექმნა, რომ შერწყმული რელაცია გახდეს:

STUDENT(StudentID, Name, PhoneNo, CampusAddress, PermanentAddress)

ტრანზიტული დამოკიდებულება

როდესაც ორი 3NF რელაცია შეირწყმება და ქმნის ერთ რელაციას, ამან შეიძლება გამოიწვიოს ტრანზიტული დამოკიდებულების წარმოქმნა. მაგალითად, განვიხილოთ შემდეგი ორი რელაცია:

STUDENT1(StudentID, MajorName)
STUDENT2(StudentID, Advisor)

იმის გამო, რომ STUDENT1 და STUDENT2 ერთნაირი პირველადი გასაღები აქვთ, ორი რელაციის შერწყმა შეიძლება:

STUDENT(StudentID, MajorName, Advisor)

ამასთან, ჩავთვალოთ, რომ თითოეულ MajorName (ძირითად მიმართულებას) ჰყავს ზუსტად ერთი მრჩეველი. ამ შემთხვევაში, მრჩეველი ფუნქციურად დამოკიდებულია MajorName–ზე:

MajorName → Advisor

თუ წინა ფუნქციური დამოკიდებულება არსებობს, მაშინ სტუდენტი არის 2NF, მაგრამ არა 3NF, რადგან ის შეიცავს ტრანზიტულ დამოკიდებულებას. ანალიტიკოსს შეუძლია შექმნას 3NF რელაციები ტრანზიტული დამოკიდებულების მოხსნის გზით. MajorName ხდება გარე გასაღები STUDENT-ში:

STUDENT(StudentID, MajorName)
MAJOR (MajorName, Advisor)

სუპერტიპი/ქვეტიპი კავშირები

ეს კავშირები შეიძლება დამალული იყოს მომხმარებლის ხედვებში ან კავშირებში. დავუშვათ, რომ საავადმყოფოთვის შემდეგი ორი რელაცია გვაქვს:

PATIENT1(PatientID, Name, Address)
PATIENT2(PatientID, RoomNo)

თავდაპირველად, როგორც ჩანს, ეს ორი რელაცია შეიძლება გაერთიანდეს ერთ რელაციაში PATIENT. ამასთან, ანალიტიკოსი სწორად ეჭვობს, რომ არსებობს ორი განსხვავებული ტიპის პაციენტი: რეზიდენტი პაციენტები და ამბულატორიები. PATIENT1 შეიცავს ატრიბუტებს, რომლებიც საერთოა ყველა პაციენტისათვის. PATIENT2 შეიცავს ატრიბუტს (RoomNo), რომელიც დამახასიათებელია მხოლოდ რეზიდენტი პაციენტებისათვის. ამ სიტუაციაში ანალიტიკოსმა უნდა შექმნას სუპერტიპი/ ქვეტიპი კავშირი ამ არსისთვის;

PATIENT(<u>PatientID</u>, Name, Address)
RESIDENTPATIENT(<u>PatientID</u>, RoomNo)
OUTPATIENT(<u>PatientID</u>, DateTreated)

ჩვენ შევქმენით OUTPATIENT რელაცია იმის საჩვენებლად, თუ როგორ შეიძლება გამოიყურებოდეს საჭიროების შემთხვევაში, მაგრამ ეს არ არის აუცილებელი, მხოლოდ PATIENT1 და PATIENT2 მომხმარებლის ხედების გათვალისწინებით.

დამასრულებელი ბიჯი რელაციური გასაღებების განსასაზღვრად

ჩვენ ადრე განვიხილეთ იდენტიფიკატორების შერჩევის რამდენიმე კრიტერიუმი: ისინი არ იცვლიან მნიშვნელობებს დროთა განმავლობაში და უნდა იყოვნენ უნიკალური და ცნობილი, არაინტელლექტუალური და იყენებდნენ ერთ სუროგატს კომპოზიტური ატრიბუტის იდენტიფიკატორისთვის. სინამდვილეში, ამ კრიტერიუმებიდან არც ერთი არ უნდა გამოიყენებოდეს მონაცემთა ბაზის დანერგვამდე (ანუ, როდესაც იდენტიფიკატორი გახდება პირველადი გასაღები და განისაზღვრება, როგორც ველი ფიზიკურ მონაცემთა ბაზაში). სანამ რელაციები ცხრილებად გარდაიქმნება, საჭიროების შემთხვევაში, რელაციების ძირითადი გასაღებები უნდა შეიცვალოს ამ კრიტერიუმების შესაბამისად.

მონაცემთა ბაზის ექსპერტებმა გააძლიერეს კრიტერიუმები პირველადი გასაღების სპეციფიკაციისათვის. ასევე სასურველია, რომ პირველადი გასაღები იყოს უნიკალური მთელ მონაცემთა ბაზაში (ე.წ. საწარმოს გასაღები) და არა მხოლოდ უნიკალური იმ ფარდობით ცხრილში, რომელზეც ის გამოიყენება. პირველადი გასაღების ეს კრიტერიუმი უფრო ჰგავს იმას, რასაც ობიექტზე ორიენტირებულ მონაცემთა ბაზაში ობიექტის იდენტიფიკატორი ეწოდება. ამ რეკომენდაციით, რელაციის პირველადი გასაღები ხდება მონაცემთა ბაზის სისტემის შიდა მნიშვნელობა და არ აქვს ბიზნესის მნიშვნელობა.

კანდიდატი პირველადი გასაღები, როგორიცაა EmpID ფიგურაში 4-1 EMPLOYEE1 რელაციაში ან CustomerID რელაციაში CUSTOMER (სურათი 4-29), თუ ოდესმე გამოიყენებოდა ორგანიზაციაში, ბიზნეს გასაღებს ან ბუნებრივ გასაღებს უწოდებენ და შევა რელაციაში როგორც არაგასაღები ატრიბუტი. EMPLOYEE1 და CUSTOMER რელაციებს (და მონაცემთა ბაზაში არსებულ ყველა სხვა რელაციას) შემდგომ აქვთ ახალი საწარმოს გასაღები ატრიბუტი (ე.წ. ვთქვათ, ObjectID), რომელსაც არ აქვს ბიზნესის მნიშვნელობა.

საწარმოს გასაღების გამოყენების ერთ-ერთი მთავარი მოტივაცია მონაცემთა ბაზის განვითარების შესაძლებლობა - მონაცემთა ბაზის შექმნის შემდეგ მონაცემთა ბაზაში ახალი რელაციების შერწყმა. მაგალითად, განვიხილოთ შემდეგი ორი რელაცია:

EMPLOYEE(EmpID, EmpName, DeptName, Salary)
CUSTOMER(CustID, CustName, Address)

ამ მაგალითში, საწარმოს გასაღების გარეშე, EmpID- ს და CustID- ს შეიძლება ჰქონდეთ ან არ ჰქონდეთ იგივე ფორმატი, სიგრძე და მონაცემთა ტიპი, იმის მიუხედავად იქნება ის ინტელექტუალური თუ არაინტელექტუალური. დავუშვათ, ორგანიზაცია ავითარებს ინფორმაციის დამუშავების საჭიროებებს და აცნობიერებს, რომ თანამშრომლებს ასევე შეუძლიათ იყვნენ მომხმარებლები, ამიტომ თანამშრომელი და მომხმარებელი უბრალოდ ერთი და იგივე PERSON სუპერტიპის ორი ქვეტიპია. ამრიგად, ორგანიზაციას სურს ჰქონდეს სამი რელაცია:

PERSON(PersonID, PersonName)
EMPLOYEE(PersonID, DeptName, Salary)
CUSTOMER(PersonID, Address)

ამ შემთხვევაში, იგულისხმება, რომ ერთი ადამიანისათვის PersonID უნდა იყოს ერთი და იგივე ყველა როლის შემთხვევაში. თუ EmpID და CustID მნიშვნელობები შეირჩა PERSON რელაციის შექმნამდე, EmpID და CustID მნიშვნელობები ალბათ არ დაემთხვევა. უფრო მეტიც, თუ EmpID-ის და CustID-ის მნიშვნელობებს შევცვლით ახალი PersonID-ის შესაბამისად, როგორ უნდა დავრწმუნდეთ, რომ ყველა EmpID და CustIDs უნიკალურია, თუ სხვა თანამშრომელს ან მომხმარებელს უკვე აქვს მინიჭებული PersonID მნიშვნელობა? კიდევ უფრო უარესი, თუ არსებობს სხვა ცხრილები, რომლებიც ეხება, ვთქვათ, EMPLOYEE– ს, მაშინ ამ სხვა ცხრილების გარე გასაღებები უნდა შეიცვალოს, რაც ქმნის გარე გასაღებების ცვლილებების ტალღურ ეფექტს. ერთადერთი გარანტირებული გზა, რომ რელაციების თითოეული პირველადი გასაღები უნიკალურია მონაცემთა ბაზაში, არის თავიდანვე საწარმოს გასაღების შექმნა, ასე რომ პირველადი გასაღებები არასდროს არ უნდა შეიცვალოს.

ჩვენს მაგალითში თავდაპირველი მონაცემთა ბაზა (PERSON–ის გარეშე) საწარმოს გასაღებით ნაჩვენებია ნახატებში 4–31 ა (რელაციები) და 4–31 ბ (მონაცემების ნიმუში). ამ ფიგურაში, EmpID და CustID ახლა ბიზნესის გასაღებია და OBJECT ყველა სხვა რელაციების სუპერტიპია. OBJECT–ს შეიძლება ჰქონდეს ატრიბუტები, როგორიცაა ობიექტის ტიპის სახელი (ამ მაგალითში შედის როგორც ატრიბუტი ObjectType), შექმნის თარიღი, ბოლოს ცვლილების თარიღი ან ნებისმიერი სხვა შიდა სისტემის ატრიბუტი არსის ეგზემპლარისათვის. შემდგომ, როდესაც საჭიროა PERSON, მონაცემთა ბაზა ვითარდება დაპროექტების თვალსაზრისით, როგორც ნაჩვენებია 4-31c ნახაზებზე (რელაციები) და 4-31d (მონაცემთა ნიმუში). მონაცემთა ბაზაში PERSON- ის განვითარება კვლავ მოითხოვს არსებულ ცხრილებში გარკვეულ ცვლილებებს, მაგრამ არა პირველადი გასაღებების მნიშვნელობების შეცვლას. ატრიბუტი name გადადის PERSON-ში, რადგან ის საერთოა ორივე ქვეტიპისთვის და EMPLOYEE- ს და CUSTOMER- ს ემატება გარე გასაღები, რათა მოხდეს „საერთო“ პირის ეგზემპლარზე მითითება. ცხრილში ადვილია არაგასაღები ატრიბუტების შესაბამისი სვეტების, თუნდაც გარე გასაღებების, დამატება და წაშლა. ამის საპირისპიროდ, რელაციის პირველადი გასაღების შეცვლა დაუშვებელია მონაცემთა ბაზის მართვის სისტემების უმრავლესობის მიერ, გარე გასაღების ტალღური ეფექტების დიდი გავლენის (ბაზაზე ზემოქმედების) გამო.

FIGURE 4-31 Enterprise key

(a) Relations with enterprise key

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, EmpName, DeptName, Salary)
CUSTOMER (OID, CustID, CustName, Address)

(b) Sample data with enterprise key

OBJECT

OID	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

EMPLOYEE

OID	EmpID	EmpName	DeptName	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

OID	CustID	CustName	Address
2	100	Fred's Warehouse	Greensboro, NC
3	101	Bargain Bonanza	Moscow, ID
6	102	Jasper's	Tallahassee, FL
7	103	Desks 'R Us	Kettering, OH

(c) Relations after adding PERSON relation

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, DeptName, Salary, PersonID)
CUSTOMER (OID, CustID, Address, PersonID)
PERSON (OID, Name)

FIGURE 4-31 ***(continued)***

(d) Sample data after adding the PERSON relation

OBJECT

OID	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER
8	PERSON
9	PERSON
10	PERSON
11	PERSON
12	PERSON
13	PERSON
14	PERSON

PERSON

OID	Name
8	Jennings, Fred
9	Fred's Warehouse
10	Bargain Bonanza
11	Hopkins, Dan
12	Huber, Ike
13	Jasper's
14	Desks 'R Us

EMPLOYEE

OID	EmpID	DeptName	Salary	PersonID
1	100	Marketing	50000	8
4	101	Purchasing	45000	11
5	102	Accounting	45000	12

CUSTOMER

OID	CustID	Address	PersonID
2	100	Greensboro, NC	9
3	101	Moscow, ID	10
6	102	Tallahassee, FL	13
7	103	Kettering, OH	14