Зертханалық жұмыс №4

MySQL ортасындағы деректер қоры кестесін құрып және байланыстыру

Екі нүкте көмегімен кез-келген шексіз және шектеулі арифметикалық прогрессияларды да анықтауға болады. Егер тізбек шектеулі болса, онда оның бірінші және соңғы элементтері беріледі. Арифметикалық прогрессияның айырмасы бірінші және екінші берілген элементтер негізінде есептеледі – жоғарыда келтірілген мысалдарда бірінші прогрессиядағы айырма – 1, ал екіншісінде – 2 болады. Сонымен, 10-ға дейінгі барлық тақ натурал сандардың тізімін анықтау үшін келесі тізімді жазу керек: [1, 3 .. 10]. Нәтижесі төмендегі тізім болады:

[1, 3, 5, 7, 9].

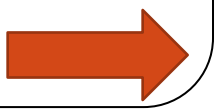
Деректердің шексіз құрылымын шексіз тізімдер негізінде де, рекурсия механизмін қолдану арқылы да анықтауға болады. Бұл жағдайда рекурсия рекурсивті функцияларға қатынау түрінде қолданылады. Шексіз деректер құрылымын құрудың үшінші тәсілі шексіз типтерді қолдану арқылы жүргізіледі.

Мысал 1. Екілік бұтақты бейнелейтін типті анықтау.

```
data Tree a

Branch
Leaf
= Leaf a
| Branch (Tree a) (Tree a)

: Tree a > Tree a > Tree a
:: a > Tree a
```



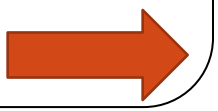
Бұл мысалда шексіз типті анықтау тәсілі көрсетілген. Рекурсия қолданылғаны көрініп тұр. Егер деректердің жаңа типін құру қажет болмаса, шексіз құрылымды функцияның көмегімен де алуға болады:

```
ones
numbersFrom n
= 1 : ones
= n : numberFrom (n + 1)

squares

= map (^2) (numbersFrom 0)
```

Бірінші функция бірліктерден тұратын шексіз тізбекті анықтайды. Екінші функция берілген саннан бастап, барлық бүтін сандар тізбегін құрады. Үшінші функция нөлмен бірге натурал сандар квадраттарының шексіз тізбегін құрады.



Функцияларды шақыру

Функцияны шақырудың математикалық қағидасы шақыру параметрлерін жақшаға алумен шектелген еді. Бұл қағиданы кейіннен барлық императивті тілдер қолданды. Бірақ функционалдық тілдерде басқа қағида қабылданған – функция атауы оның параметрінен жай бос орын арқылы ажыратылады.

Lisp'те параметрі L length функциясын шақыру келесі тізім түрінде жазылады: (length L). Мұндай қағида функционалдық тілдердегі көптеген функциялар каррирленген екенін көрсетеді. Haskell'де функцияны шақыруды тізім түріне келтірудің қажеттігі жоқ. Мысалы, екі санды қосатын функция анықталған болса:

```
add
add x y
:: Integer > Integer > Integer
= x + y
```

Онда, оны нақты параметрлерімен шақыру келесі түрде болады:

```
add 5 7
```

Бұл жерде Haskell' қағидасы абстрактілі математикалық тіл қағидасына ұқсайтыны көрініп тұр. Бірақ Haskell Lisp'тен ары асып, типі A1 □→ (A2 □→ ... (An □→ B) ...) түрінде көрсетуге болмайтын, каррирленбеген функияларды да сипаттайтын қағида бар екенін көрсетті. И эта нотация, как и в императивных языках программирования, использует круглые скобки:

```
add (x, y) = x + y
```

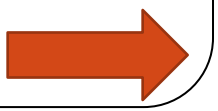
Соңғы жазу - Haskell' қағидасы бойынша жазылған бір аргументті функция екенін көруге болады. Екінші жағынан каррирленген функциялар үшін бөліктеп қолданумүмкін болатынын да байқауға болады. Яғни, екі аргументті функцияны шақыру кезінде оған тек біреуін ған берсе жеткілікті.

Бұл құбылысты берілген аргументке бірлік қосып отыратын inc функциясының мысалы арқылы нақтырақ көрсетуге болады:

```
inc :: Integer > Integer
inc = add 1
```

Бұл жағдайда бір параметрлі inc функциясын шақыру, біріншісі – 1 болатын, екі параметрлі add функциясын шақыруға әкеледі. Бұл бөліктеп қолданудың интуитивті түсінігі. Түсінікті бекіту үшін map функциясының классикалық мысалын қарастыруға болады. Келесі мысалда Haskell'де map функциясы:

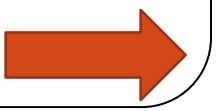
```
map
map f []
map f (x:xs)
:: (a > b) > [a] > [b]
= []
= (f x) : (map f xs)
```



Мұнда prefix - қос нүкте операциясының инфиксті жазуы қолданылғаны көрініп тұр, жұптарды құру және белгілеу үшін мұндай жазу тек Haskell' қағидасында ғана қолданылады. Жоғарыдағы анықтаманы келтіргеннен кейін ғана келесі шақыруды жүргізуге болады:

```
map (add 1) [1, 2, 3, 4]
```

Нәтижесі [2, 3, 4, 5] түріндегі тізім болады.



Мәтіндік типтер және жолдар:

- *CHAR* – тіркелген ұзындықтың жолдарын сақтау;
- *VARCHAR*– ауыспалы ұзындықтың жолдарын сақтау;
- *TEXT, BLOB* және олардың нұсқалары – мәтіннің үлкен фрагменттерін сақтау;

ENUM және *SET* – берілген тізімнен белгілерді сақтау

Күнтізбелік календар мәліметтердің күнтізбелік типтері (4-кесте):

DATE – дкүнді сақтау үшін (формат *YYYY-MM-DD* түр күндері үшін 2009-10-15 және формат *YY-MM-DD* 09-10-15 түр күндері үшін);

TIME – тәулік уақытын сақтау үшін (формат *HH:MM:SS*, бұл жерде *HH* – сағаттар, *MM* – минуттар, *SS* – секундтар, мысалы, 10:48:56);

DATETIME– тәуліктің күнін және уақытын көрсету үшін;

TIMESTAMP – егер нақты таяқшада жолдар нақты белгіні немесе *NULL* көрсетпесе, ол жерде уақыты жазылады, ол кезде сәйкес жолдар құрылады немесе соңғылары өзгереді (*DATETIME* форматы);

YEAR – тек жылды ғана сақтауға мүмкіндік жасайды.