НАУЧНОЕ ИЗДАНИЕ МГТУ ИМ. Н.Э. БАУМАНА
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 48211. Государственная регистрация №0421200025. ISSN 1994-0408
электронный научно-технический журнал

Алгоритмы назначения первичных ключей в заполненных таблицах

77-48211/425188
06, июнь 2012
Брешенков А. В., Мин Т. Т.
УДК 681.3.07

Россия, МГТУ им. Н.Э. Баумана
breshenkov@rambler.ru

Введение

В классических работах, посвященных проектированию реляционных баз данных РБД [1, 2], нередко упоминается о том, что желательно формализовать выполнение большинства шагов проектирования. Это в частности касается нормализации отношений и назначения ключевых полей. Так как проектные решения в традиционной методологии проектирования РБД принимаются, как правило, на основе анализа предполагаемых схем отношений, а не на основе анализа реальных данных, формализация большинства задач проектирования РБД трудноосуществима.

Положение меняется, когда РБД проектируются на основе имеющихся данных. В работах [3, 4] рассматриваются задачи, решение которых можно формализовать при наличии данных, на основе которых проектируются РБД. В частности при назначении ключевых полей возможна разработка алгоритмов на основе анализа данных, содержащихся в таблицах. В статье рассматривается эти алгоритмы.

В рамках задачи назначения ключевых полей в существующих заполненных таблицах необходимо разработать следующие алгоритмы:

- выявления домена с уникальными значениями его элементов;
- выявления сочетания доменов с уникальными сочетаниями соответствующих им элементов;
- поиска минимальных первичных ключей, включающих в себя один атрибут;
- поиска минимальных первичных ключей, включающих в себя несколько атрибутов;

- выявления атрибутов, которые входят в первичный ключ и содержат уникальные значения;
- выявления внешних ключей.

При этом в отличие от алгоритмов, предложенных в работах [3, 7]:

- выявление первичных ключей включается в этап преобразования информации табличного вида (ИТВ) в реляционные таблиц (РТ), и тем самым обеспечивается одно из требований к РТ;

- выявление сочетания доменов с уникальными сочетаниями соответствующих им элементов выполняется более тщательно и детально;

- поиск минимальных первичных ключей, включающих в себя несколько атрибутов, выполняется в соответствии со всеми требованиями к минимальности ключей;

- выявление внешних ключей включаются в этап преобразования ИТВ в РТ и тем самым на ранних этапах проектирования РБД решается важная задача.

1. Проблема назначения ключевых полей в заполненных таблицах

В работе [7] предложен метод назначения первичных ключей в информации табличного вида, который вполне приемлем для использования. Однако в нем учтены не все особенности ключевых полей. В частности:

- рассматривается возможность включения в первичный ключ только 3-х атрибутов;

- не полностью учитывается требование минимальности первичного ключа;

- не до конца прояснены вопросы формирования первичных ключей из нескольких атрибутов;

- затруднительно понимание предложенной формализации;

- мало освещены вопросы назначения внешних ключей

- назначение первичных ключей не рассматривается как неотъемлемая задача преобразования ИТВ в РТ.

Основными требованиями к первичным ключам являются уникальность и минимальность. Формализуем эти требования, а затем используем их в качестве целевых функций при разработке соответствующих алгоритмов.

Уникальность. Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где $к$ – степень отношения; A_i –атрибут отношения.

$A_i = \{e_{i_1},...,e_{i_j},...,e_{i_n}\}$ $j = \overline{1,n}$, где n – мощность отношения, e_{i_j} - j – й элемент атрибута A_i.

$\mathrm{A_m} = \{e_{m_1},...,e_{m_j},...,e_{mn}\}$ $j = \overline{1,n}$, где n – мощность отношения, e_{m_j} - j – й элемент атрибута A_m

Необходимо найти такие атрибуты $A_i,\ldots,A_m$, чтобы обеспечилась истинность выражения: $\mathrm{concat}(e_{i_1},\ldots,e_{m_1}) \neq,...,\neq \mathrm{concat}(e_{i_j},\ldots,e_{m_j}) \neq,...,\neq \mathrm{concat}(e_{i_n},\ldots,e_{mn})$ (1)

Из выражения (1) следует, что необходимо найти такое сочетание атрибутов, чтобы конкатенация их значений была уникальна. При этом:

- проверяемый кортеж атрибутов может включать несколько атрибутов;
- число возможных сочетаний атрибутов может быть очень большим – это зависит от степени отношения (общего числа атрибутов в отношении);
- ключевой атрибут может быть только один;
- может не найтись таких атрибутов, которые обеспечивают истинность выражения (1), в этом случае назначают суррогатный ключ.

В процессе назначения первичных ключей в рамках традиционной технологии РБД исходят из визуального анализа предполагаемых схем отношений, опыта разработок, особенностей предметной области. Но такой подход не всегда приводит к успеху. Действительно, в схеме отношения могут быть погрешности, степень отношения может измеряться сотнями единиц, после заполнения таблицы могут проявиться ее непредусмотренные особенности. Но альтернативного решения пока нет.

При наличии ИТВ разработчик понимает семантику таблицы, знает степень и мощность отношения, а главное имеет возможность анализа реальных данных. В этом случае процесс формирования первичного ключа вполне формализуем. И возможна разработка алгоритмов и соответствующего метода, которые обеспечат наилучшее решение.

Минимальность. Минимальность ключевого поля рассматривается в двух аспектах.

В первом случае во главу угла становится объем памяти, который необходим для хранения значений атрибутов, входящих в первичный ключ. Поэтому самая очевидная целевая функция – минимальное число атрибутов, входящих в первичный ключ:

$\min|\mathrm{A1},\ldots,\mathrm{Ai},\ldots,\mathrm{Am},\ldots,\mathrm{Ak}|$, где $i = \overline{1,k}$, где κ – число атрибутов, входящих в первичный ключ; A_i – атрибут отношения, входящий в первичный ключ.

Строго говоря, более правильная целевая функция следующая:

$\min(\mathrm{Length}(\mathrm{A_i})+\ldots+\mathrm{Length}(\mathrm{A_к}))$

Действительно, минимальность ключей определяется не только количеством атрибутов, входящих в них, но и суммарной длиной этих атрибутов. А длина атрибутов в основном определяется их типом и свойствами. В общем случае для выяснения средней длины значения атрибута знания его типа не всегда достаточно. Например, данные символьного типа могут быть представлены значениями меньшими допустимой длины для данного типа и даже меньшими установленного ограничения в свойствах атрибута.

При наличии только схемы отношения зачастую непросто выбрать атрибуты, которые составят первичные ключ. При недостаточном знании предметной области можно ошибиться, т.к. после заполнения таблицы реальная длина данных может быть больше или меньше предполагаемой.

Используя ИТВ, проектировщик РБД имеет возможность принимать решение на основе реальных данных. Более того, процесс поиска минимальных ключей можно формализовать.

Во втором случае под минимальностью первичного ключа подразумевается отсутствие в составе ключа атрибута, значения которого уникальны [7]. Пусть первичный ключ К представлен множеством атрибутов:

$K = (A_1,...,A_i,...,A_j,...,A_k)$, $i = \overline{1,k}$, где k – число атрибутов, входящих в первичный ключ;

$A_i - i$ - й атрибут отношения, входящий в первичный ключ.

$S_1 = (a_{1_1},...,a_{1_i},...,a_{1_j},...,a_{1_k})$, где S_1- часть 1-й записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

$S_n = (a_{n_1},...,a_{n_i},...,a_{n_j},...,a_{n_k})$, где S_n- часть n-й записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

$S_m = (a_{m_1},...,a_{m_i},...,a_{m_j},...,a_{m_k})$, где S_m- часть m-й, последней записи отношения, соответствующая набору атрибутов, входящих в первичный ключ.

m – мощность отношения;

a_{n_j} - значение j – го атрибута n – й записи.

Тогда для ключевого поля, которое включает в себя несколько атрибутов, должно выполняться условие:

$$\neg((a_{1_1} \neq,...,\neq a_{n_1} \neq ... \neq a_{m_1}) \vee (a_{1i} \neq,...,\neq a_{n_i} \neq ... \neq a_{m_i}) \vee$$
$$(a_{1j} \neq,...,\neq a_{n_j} \neq ... \neq a_{m_j}) \vee (a_{1_k} \neq,...,\neq a_{nk} \neq ... \neq a_{mk}))$$

Интересно отметить, что это условие, по сути, противоположно условию уникальности ключа, если он включает в себя единственный атрибут.

На основе целевых функций, сформулированных выше, далее предлагаются неформальные, а затем формальные алгоритмы назначения первичных ключей в заполненных таблицах

2. Неформальные алгоритмы назначения первичных ключей в заполненных таблицах

Следует отметить, что в качестве ключевых полей а также в качестве атрибутов, входящих в первичный ключ, не рассматриваются атрибуты, которые имеют тип логический, MEMO, LOB, BLOB, поле объекта OLE. В связи с этим такие атрибуты необходимо исключить из рассмотрения.

П1. Поиск единственного атрибута, все значения которого уникальны.

$MKA = \varnothing$, где MKA - множество ключевых атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где κ – степень отношения; A_i –атрибут отношения.

$A_i = (e_{i_1} \ldots . e_{i_m})$, где e_{i_1} - 1-й элемент домена с атрибутом A_i, e_{i_m} - m-й элемент домена с атрибутом A_i.

Выполняется поиск такого атрибута A_i такого, что $e_{i_1} \neq \ldots . \neq e_{i_m}$.

В связи с этим перебираются все атрибуты отношения.

Если такой атрибут находится, то он запоминается: $MKA = MKA + A_i$

Если после перебора всех атрибутов $MKA = \varnothing$, то это означает, что назначить первичный ключ, включающий в себя один атрибут невозможно.

В этой связи разработчик должен принять одно из решений: назначить суррогатный ключ или сформировать первичный ключ на базе нескольких атрибутов. Если принято 2-е решение, то переход к П.2

Если $MKA \neq \varnothing$, то атрибуты с уникальными значениями нашлись и необходимо выбрать из них атрибут, удовлетворяющий требованиям минимальности. Так как на роль ключевого атрибута претендует только один из найденных атрибутов, то необходима проверка только первой части требования минимальности длины атрибута. В связи с этим необходимо найти в множестве MKA атрибут с минимальной длиной, т.е.

min(Length(A_i),…, Length($A_к$)), где (A_i ,…, $A_к$) $\in MKA$ Таким образом, найден первичный ключ. STOP.

Несмотря на то, что найден атрибут минимальной длины, разработчику должен быть предоставлен весь список возможных ключевых полей, чтобы он имел возможность выбора ключевого атрибута пусть и не минимального. Это может оказаться необходимо в связи с особенностями предметной области.

П2. Поиск атрибутов, конкатенация значений которых минимальна.

Необходимо проанализировать все возможные сочетания атрибутов. Каждое сочетание проверить на уникальность конкатенации их значений.

Все сочетания с уникальными значениями необходимо сохранить, а затем для, каждого сочетания измерить их суммарную длину. Для обеспечения возможности принятия волевого решения необходимо расположить эти сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности, и придется выбирать альтернативный ключ.

Имеет смысл сначала выполнить проверку соответствие требований к первичному ключу 2-х атрибутов, т.е. необходимо проанализировать C_N^2 сочетаний, где N степень отношения.

$MKA^2 = \varnothing$, где MKA^2 - множество пар атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где $к$ – степень отношения; A_i –атрибут отношения.

$MPA = \varnothing$

Ищутся все возможные сочетания пар атрибутов и запоминаются в массив MPA:

```
Count=0
For i = 1 to k-1
    For j = i+1 to k
        Count = Count + 1
        S = Concat (Ai , Aj)
       MPA (Count) = MPA + S
    Next i
Next j
```

Таким образом, в массиве МРА сформируются все возможные пары атрибутов, а счетчике *Count* хранится их количество.

Проверяются все пары на уникальность.

MUP = ∅ / Массив пар атрибутов, представляющих собой атрибуты, все соответствующие пары значений которых уникальны */*

Count1 = 0

For i = 1 to Count

S = MPA(Count)

/ По сути S представляет собой пару атрибутов* (A_i, A_j) $A_i = (e_{i_1} \ldots e_{i_m})$, *где* e_{i_1} *- 1-й элемент домена с атрибутом* A_i, e_{i_m} *- m-й элемент домена с атрибутом* A_i.

$A_j = (e_{j_1} \ldots e_{j_m})$, *где* e_{j_1} *- 1-й элемент домена с атрибутом* A_j, e_{j_m} *- m-й элемент домена с атрибутом* A_j, *m – мощность отношения. */*

For n = 1 to m

Для каждой пары атрибутов (A_i, A_j) *выполняется проверка условия* $Concat(e_{i_1}, e_{j_1}) \neq \ldots \neq Concat((e_{j_1}, e_{j_m})$

Next n

Если текущая пара атрибутов имеет все соответствующие пары значений уникальными, то эта пара добавляется к массиву пар с уникальными значениями:

Count1 = Count1 + 1

MUP(Count1) = S

Next i

Если претенденты на ключевой атрибут найдены, т.е. , $MUP \neq \varnothing$, то для проверки второго требования минимальности выполняется переход к П3. Для обеспечения возможности принятия волевого решения необходимо расположить все найденные сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности и придется выбирать альтернативный ключ.

Если не найдено таких двух атрибутов, которые удовлетворяют требованиям к первичному ключу, то разработчик может назначить суррогатный ключ или попытаться найти такие 3-и атрибута, которые удовлетворяют требованиям к первичному ключу, т.е. проанализировать C_N^3 сочетаний.

П3. Поиск первичного ключа на основе 3-х атрибутов

$MKA^3 = \varnothing$, где MKA^3 - множество троек атрибутов, которые претендуют на роль первичного ключа.

Пусть имеется отношение R:

$R=(A_1, \ldots, A_i, \ldots, A_m, \ldots, A_k)$, $i = \overline{1,k}$, где $к$ – степень отношения; A_i –атрибут отношения.

$MPA = \varnothing$

Ищутся все возможные сочетания троек атрибутов и запоминаются в массив MPA:

```
Count=0
For i = 1 to k-2
    For j = i+1 to k
        For r = j+1 to k
            Count = Count + 1
            S = Concat (Ai , Aj, , Ar)
            MPA (Count) = MPA + S
        Next r
    Next i
Next j
```

Таким образом, в массиве MPA сформируются все возможные тройки атрибутов, а счетчике *Count* хранится их количество.

Проверяются все тройки на уникальность.

```
MUP = ∅ /* Массив троек атрибутов, представляющих собой атрибуты, у которых соответствующие тройки значений которых уникальны */
Count1 = 0
For i = 1 to Count
    S = MPA(Count)
     /* По сути S представляет собой тройку атрибутов (Ai, Aj, , Ar )
     Ai = (ei1 ....eim ), где ei1 - 1-й элемент домена с атрибутом Ai , eim - m-й элемент домена с атрибутом Ai .
     Aj = (ej1 ....ejm ), где ej1 - 1-й элемент домена с атрибутом Aj , erm - m-й элемент домена с атрибутом Aj , m – мощность отношения.
```

*A_r = (e_{r_1}...e_{r_m}), где e_{r_1} - 1-й элемент домена с атрибутом A_r, e_{j_m} - m-й элемент домена с атрибутом A_r, m – мощность отношения. */*

For n = 1 to m

Для каждой тройки атрибутов (A_i, A_j, A_r) выполняется проверка условия Concat(e_{i_1}, e_{r_1}, e_{j_1}) ≠ .... ≠ Concat((e_{j_1},.e_{r_m}, e_{j_m}))

Next n

Если текущая тройка атрибутов имеет все соответствующие тройки значений уникальными, то эта тройка добавляется к массиву троек с уникальными значениями:

Count1 = Count1 + 1

MUP(Count1) = S

Next i

Если претенденты на ключевой атрибут найдены, т.е., $MUP \neq \varnothing$,то для проверки второго требования минимальности выполняется переход к П3.

Для обеспечения возможности принятия волевого решения необходимо расположить все найденные сочетания в порядке возрастания. Более того, найденный ключ может не удовлетворять второму требованию минимальности и придется выбирать альтернативный ключ.

Аналогичный подход распространяется и на 4 и на 5 атрибутов, но как показывает практика, в ключевом поле очень редко задействуют более 4-х атрибутов.

Может сложиться впечатление, что этот процесс займет немало машинного времени, которое напрямую зависит от степени отношения. Ведь число возможных комбинаций может оказаться очень большим. Но это не совсем так.

Из комбинаторики известно, что число возможных сочетаний C_n^k = n!/(n-k)!/k!, где n - число элементов множества, а к – количество проверяемых значений. Например, для множества из 4-х элементов (a, b, c, d) возможны следующие сочетания пар элементов (a, b), (a, c), (a, d), (b, c), (b, d) , (c, d).

Подсчитаны числа возможных сочетаний для различных n и k, которые представляют наибольший интерес. Результаты сведены в таблицу 1.

Таблица 1

C_{10}^2	C_{10}^3	C_{10}^4	C_{20}^2	C_{20}^3	C_{20}^4	C_{30}^2	C_{30}^3	C_{30}^4
45	120	210	190	1140	4845	435	4060	27405

Степени отношений (10, 20, 30) наиболее близки к распространенным степеням в реальных БД, а число атрибутов (2, 3 , 4) обычно достаточно для первичного ключа. Подсчитанное число сочетаний вполне может быть обработано на современных компьютерах без существенной потери машинного времени.

П4. Поиск первичных ключей, у которых нет атрибутов, входящих в первичный ключ, значения которого уникальны.

После выполнения П2 настоящего алгоритма будет получен массив пар или массив троек атрибутов, представляющих собой атрибуты, у которых соответствующие тройки значений уникальны MUP $\neq \varnothing$. Необходимо проверить все элементы массивов на предмет наличия атрибутов, которые входят в элементы массива и которые имеют уникальные значения. Если такие элементы (атрибуты) найдутся, то они не могут в соответствии со вторым правилом минимальности первичного ключа претендовать на составную часть первичного ключа.

Проверка массива пар атрибутов.

В этом случае $MUP = (P_1, \ldots, P_i, \ldots P_n)$, где P_i - i-я пара атрибутов, n – число пар атрибутов с уникальными конкатенациями значений.

$$P_i = (A_{i1}, A_{i2})$$

$A_{i1} = (e_{i1_1}, \ldots, e_{i1_m})$, *где* e_{i1_1} - 1-й элемент домена с атрибутом A_{i1}, m – мощность отношения.

$A_{i2} = (e_{i2_1}, \ldots, e_{i2_m})$, *где* e_{i2_1} - 1-й элемент домена с атрибутом A_{i2}, m – мощность отношения.

При этом должно выполняться условие:

$$\neg \left((e_{i1_1} \neq, \ldots, \neq e_{i1_m}) \vee (e_{i2_1} \neq, \ldots, \neq e_{i2_m}) \right)$$

Подобные условия должны выполняться для всех элементов массива $MUP = (P_1, \ldots, P_i, \ldots P_n)$, то есть для всех пар атрибутов, претендующих на ключевые.

Проверка массива троек атрибутов.

В этом случае $MUP = (P_1, \ldots, P_i, \ldots P_n)$, где P_i - i-я тройка атрибутов, n – число пар атрибутов с уникальными конкатенациями значений.

$P_i = (A_{i1}, A_{i2}, A_{i3})$

$A_{i1} = (e_{i1_1},...,e_{i1_m})$, *где* e_{i1_1} - 1-й элемент домена с атрибутом A_{i1}, m – мощность отношения.

$A_{i2} = (e_{i2_1},...,e_{i2_m})$, *где* e_{i2_1} - 1-й элемент домена с атрибутом A_{i2}, m – мощность отношения.

$A_{i3} = (e_{i3_1},...,e_{i3_m})$, *где* e_{i3_1} - 1-й элемент домена с атрибутом A_{i3}, m – мощность отношения.

При этом должно выполняться условие:

$$\neg ((e_{i1_1} \neq,...,\neq e_{i1_m}) \vee (e_{i2_1} \neq,...,\neq e_{i2_m}) \vee e_{i3_1} \neq,...,\neq e_{i3_m}))$$

Подобные условия должны выполняться для всех элементов массива $MUP = (P_1, \ldots, P_i, \ldots P_n)$, то есть для всех троек атрибутов, претендующих на ключевые.

Для четверок атрибутов, претендующих на роль первичного ключа, выполняются аналогичные пункты алгоритма. STOP.

В конечном итоге будет получен список пар или троек атрибутов, которые претендуют на первичный ключ. Из этого списка разработчик должен выбрать единственный кортеж исходя из своих прикладных соображений. Список может состоять и из одного элемента. Если список окажется пустым, то назначается суррогатный ключ. Однако такая ситуация скорее всего будет отсечена на шаге поиска первичного ключа, включающего в себя один атрибут.

Заключение

В статье определен состав алгоритмов, которые необходимо разработать для автоматизированного назначения первичных ключей в заполненных реляционных таблицах. Сформулирована проблема назначения первичных ключей. Предложены неформальные алгоритмы назначения первичных ключей в заполненных таблицах.

Литература

1. Монографии, брошюры и т.п.:

1. Дейт К. Дж. Введение в системы баз данных: Пер. с англ. - М.: Наука, 1980.- 464 с.

2. Дейт К. Дж. Введение в системы баз данных. 6-е изд.: Пер. с англ. - Киев: Диалектика, 1998. - 784 с.

3. Брешенков А.В. Методы решения задач проектирования реляционных баз

данных на основе использования существующей информации табличного вида. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. – 154 с.

4. Брешенков А.В. Базы данных. Проектирование баз данных на основе информации табличного вида. LAP LAMBERT Academic Publishing GmbH & Co. KG Dudweiler, rbr, 66123 Saarbrucken, Germany,2011, 394 с.

5. Розмахов О.Г. Основы проектирования баз данных. - М.: Московский авиационный институт, 1993. - 24 с.

2. Диссертации и авторефераты:

6. Брешенков А.В. Методология проектирования реляционных баз данных с использованием данных табличного вида. Дис. доктор техн. наук (05.25.05) – М., 2007

3. Электронные издания:

7. Брешенков А.В., Белоус В.В. Метод назначения первичных ключей в информации табличного вида. Наука и образование. Инженерное образование: Эл. науч. издание. – 2010. (Номер гос. регистрации 0321000195)

SCIENTIFIC PERIODICAL OF THE BAUMAN MSTU
SCIENCE and EDUCATION
EL № FS 77 - 48211. №0421200025. ISSN 1994-0408
electronic scientific and technical journal

Algorithms for assignment of primary keys in filled tables

77-48211/425188
06, June 2012

Breshenkov A.V., Min Thet Tin

Russia, Bauman Moscow State Technical University
breshenkov@rambler.ru

In the article the authors define a set of algorithms required for automated assignment of primary keys in filled relational tables. The problem of assigning primary keys is formulated. Informal algorithms are proposed for assignment of primary keys in filled tables.

Publications with keywords: data bases, primary key, algorithms, relational database, attribute, key fields, uniqueness, minimality, surrogate key
Publications with words: data bases, primary key, algorithms, relational database, attribute, key fields, uniqueness, minimality, surrogate key

References

1. Date C.J. *An introduction to database systems*. 2nd ed. Reading, MA, Addison-Wesley, 1979. (Russ. ed.: Deit K.Dzh. *Vvedenie v sistemy baz dannykh*. Moscow, Nauka Publ., 1980. 464 p.).

2. Date C.J. *An introduction to database systems*. 6th ed. Reading, MA, Addison-Wesley, 1997. 839 p. (Russ. ed.: Deit K.Dzh. *Vvedenie v sistemy baz dannykh*. 6th ed. Kiev, Dialektika Publ., 1998. 784 p.).

3. Breshenkov A.V. *Metody resheniia zadach proektirovaniia reliatsionnykh baz dannykh na osnove ispol'zovaniia sushchestvuiushchei informatsii tablichnogo vida* [Methods for solving problems of designing relational databases using the existing tabular form information]. Moscow, Bauman MSTU Publ., 2007. 154 p.

4. Breshenkov A.V. *Bazy dannykh. Proektirovanie baz dannykh na osnove d ctablichnogo vida* [Databases. Database design based on the tabular form information]. Saarbrucken, Lambert Academic Publ. GmbH & Co., 2011. 394 p.

5. Rozmakhov O.G. *Osnovy proektirovaniia baz dannykh* [Fundamentals of database design]. Moscow, MAI Publ., 1993, 24 p.

6. Breshenkov A.V. *Metodologiia proektirovaniia reliatsionnykh baz dannykh s ispol'zovaniem dannykh tablichnogo vida*. *Diss. dokt. tekhn. nauk* [The methodology of designing relational databases using tabular type data. Dr. tech. sci. diss.]. Moscow, 2007.

7. Breshenkov A.V., Belous V.V. Metod naznacheniia pervichnykh kliuchei v informatsii tablichnogo vida [Method of appointment of primary keys in the information of a tabular kind]. *Nauka i obrazovanie*, 2009, no. 12. Available at: http://technomag.edu.ru/doc/134299.html.