**M. Nowakowska, WZiMK, PŚk. BD; notatki z wykładów**

**Zależności funkcyjne w tabeli**

Oznaczenia: R – tabela bazy Danych, K główny – klucz tabeli

* Zależność częściowa

R ={A, B, C, D}, K ={A, B} A 🡪 C to oznacza, że C jest częściowo zależny od klucza K

* Zależność przechodnia

R ={A, B, C, D}, K ={A} A 🡪 C 🡪 D to oznacza, że D jest przechodnio zależny od klucza K, poprzez C

**Relationship – relacja, powiązanie**

**Relation, table – relacja, tabela**

**Implementacja złączenia ∞ - ∞**

**Operator modulo *Mod***

**Dzięwczęta**

**Id | Osoba**

d1 | dddd1

d2 | dddd2

d3 | dddd3

d4 | dddd4

d5 | dddd5

**Chłopcy**

**Id | Osoba**

c1 | cccc11

c2 | cccc22

c3 | cccc33

c4 | cccc44

c5 | cccc55

c6 | cccc66

c7 | cccc77

c8 | cccc88

∞

∞

**Dzięwczęta**

**Id | Osoba**

d1 | dddd1

d2 | dddd2

d3 | dddd3

d4 | dddd4

d5 | dddd5

**Chłopcy**

**Id | Osoba**

c1 | cccc11

c2 | cccc22

c3 | cccc33

c4 | cccc44

c5 | cccc55

c6 | cccc66

c7 | cccc77

c8 | cccc88

**Sympatyzowanie**

**Id\_d | Id\_c**

d1 | c1

d1 | c3

d1 | c4

d2 | c6

d4 | c6

d5 | c8

1

1

∞

∞

Zwraca resztę z dzielenia dwóch liczb całkowitych

4 mod 2 --> 0

5 mod 2 --> 1

2 mod 5 --> 2

**Operatory dzielenia: / , \**

Dzielenie rzeczywiste / – zwraca liczbę rzeczywistą z częścią ułamkową

5 / 2 --> 2,5

Dzielenie całkowite \ – zwraca liczbę całkowitą, część całkowitą wyniku dzielenia

5 \ 2 --> 2

2 \5 🡪 0

<, >, <=, >=, =, !=

**Operator relacyjny <>**

operator „różny od”: <> zwraca wartość *fałsz* (*false*) jeżeli oba argumenty są takie same, w przeciwnym przypadku zwraca wartość *prawda* (*true*).

5 <> 5 --> *fałsz*

|  |  |
| --- | --- |
| **Cena** | **Cena <> 50** |
| 65, zł | prawda |
| 100,00 zł | prawda |
| 50,00 zł | fałsz |
| 45,00 zł | prawda |
| 50,00 zł | fałsz |
| 250,00 zł | prawda |

**Operacje na datach**

* stałe datowe umieszcza się pomiędzy znakami # (hash): #2005-01-01#
* operacje arytmetyczne na datach:
  + do daty można dodać liczbę całkowitą, wynik: **data** o określoną liczbę dni późniejsza:

np. #2010-11-30# +1 --> #2010-12-01#

* + od daty można odjąć liczbę całkowitą, wynik: **data** o określoną liczbę dni wcześniejsza:

np. #2010-11-30# - 2 --> #2010-11-28#

* + można odjąć od siebie dwie daty, wynik: **liczba całkowita** określająca liczbę dni jakie minęły od jednej daty do drugiej:

np. #2010-11-30# - #2010-11-28# --> 2

* funkcja pracujące na datach:
  + *Date*(), *Now*() zwraca bieżącą datę z zegara systemowego komputera (druga jeszcze godzinę)
  + *DateAdd* (jaki\_interwał , liczba\_interwałów , data )

Najlepiej stosować dla interwałów nierównej długości, Jak lata, kwartały i miesiące.

* + *DateDiff* (jaki\_interwał , data1 , data2 )

Zwraca informację ile określonych interwałów czasowych istnieje między dwiema datami. Por. też opis funkcji w Internecie.

* + *Year*(jakaś\_data) zwraca numer roku z daty
  + *Month*(jakaś\_data) zwraca numer miesiąca z daty
  + *Day*(jakaś\_data) zwraca numer dnia z daty

np. Year(#2010-11-30#) --> 2010

Month(#2010-11-30#) --> 11

Day(#2010-11-30#) --> 30

Jeżeli dzisiaj jest 2021-04-04

Date() -> 2021-04-04

Year(Date()) -> 2021

Month(Date())+10-day(date()) = Month (#2021-04-04#)+10-Day(#2021-04-04#) ->

4 +10 – 4 =10

Date()+1 -> 04/04/2021+1 (2021-04-04+1) 🡪 05/04/2021

Date()-1 -> 04/04/2021 – 1 🡪 03/04/2021

#2021-03-01# - #2021-03-15# -> -14

#2021-03-15# - #2021-03-01# -> 14

Różne rozwiązania do kwerendy Finanse pracowników

Dodatek\_staż: IIf([Staż]>20;0,2\*[Podst\_wyn];[Podst\_wyn]\*[Staż]/100)

Dodatek\_staż: [Podst\_wyn]\*IIf([Staż]>20;0,2;[Staż]/100)

Dodatek\_staż: [Podst\_wyn]\*

IIf((date()-[Data\_zat])\365,25>20;0,2; (date()-[Data\_zat])\365,25/100)

**Operacje na tekstach**

sklejanie/konkatenacja tekstów – operator konkatencji czyli sklejenia (&, +):

np. "ala" & "ma" & "asa" --> "alamaasa"

"ala " & "ma " & "asa" --> "ala ma asa"

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Nazwisko** | **Imię** | **Czytelnik: [Nazwisko] & " " & [Imię]** |
|  | aaa | zzz | aaa zzz |
|  | bbbb | xxx | bbbb xxx |
|  | ccc | yyy | ccc yyy |
|  | dddddd | uu | dddddd uu |

operator podobieństwa: *Like* stosuje się z symbolami wieloznacznymi:

\* (dowolny ciąg znaków), ? (jeden dowolny znak)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Miejscowość** |  | **Wynik działania kryterium** |
|  | Kielce |  | Wrocław |
|  | Końskie |  | Wolbrom |
|  | Kołobrzeg |  |  |
|  | Koło |  |  |
|  | Kolbuszowa |  |  |
|  | Kije |  |  |
|  | Wicie |  |  |
|  | Warszawa |  |  |
|  | Wrocław |  |  |
|  | Wolbrom |  |  |
|  |  |  |  |
| Filtr (kryterium) | like "W??????" |  | *<-- Tekst zaczynający się literą „W” i mający potem 6 dowolnych znaków* |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Miejscowość** |  | **Wynik działania kryterium** |
|  | Kielce |  | Kielce |
|  | Końskie |  | Końskie |
|  | Kołobrzeg |  | Kije |
|  | Koło |  |  |
|  | Kolbuszowa |  |  |
|  | Kije |  |  |
|  | Wicie |  |  |
|  | Warszawa |  |  |
|  | Wrocław |  |  |
|  | Wolbrom |  |  |
|  |  |  |  |
| Filtr (kryterium) | like "K\*e" |  | <-- *Tekst zaczynający się literą „K i kończący się literą „e”* |
|  | Like [Podaj nazwę miasta] &"\*" |  | <-- *Tekst zaczynający się na podany ciąg znaków* |

Nie zaleca się używać operatora Like w połączeniu z tekstami niezawierającymi symboli wieloznacznych, np. zamiast: *Like "Kielce"* należy pisać: =*"Kielce"* albo *"Kielce"* (znak przyrównania = jest operatorem domyślnym w siatce projektowej Access’a w pozycji Kryterium).

porównywanie tekstów: =, >, >=, <, <=

[Nazwisko] > "K\*"

|  |  |  |
| --- | --- | --- |
| Kije | Po uporządkowaniu rosnącym | Kielce |
| Kielce |  | Kije |

wycinanie tekstów

Do operacji aktualizacji pola:

Z0001 -> 0001 -> po konkatenacji z X -> X0001

Jest wykorzystywane wyrażenie:

"X" & Right([ID\_CZYT];Len([ID\_CZYT])-1)

Right (*tekst*; *n*) wycina z wyrażenia *tekst* *n* znaków począwszy od prawej strony, np.

Right ("*marabut*"*;* *5*) -> *rabut*

Left(*tekst*; *n*) wycina z wyrażenia *tekst* *n* znaków począwszy od lewej strony, np.

Left ("*marabut*"*;* *5*) -> *marab*

Len (*tekst*) zwraca długość (liczba znaków) wyrażenia *tekst*, np.

Len ("*ma rabut*") -> 8

Right("*marabut*"*;* Len("*marabut*")-1) -> *arabut*

**Funkcja warunkowa**

**iif**(*warunek*; *wartość dla prawdy*; *wartość dla fałszu*)

jeżeli *warunek* jest prawdziwy, funkcja zawraca *wartość dla prawdy*

jeżeli *warunek* jest nieprawdziwy, funkcja zawraca *wartość dla fałszu*

Zakładając, że dzisiaj jest 2020/04/17:

iif(month(date())=12 and day(date())=24; "piękny dzień"; "inny dzień") -> "inny dzień"

W wigilię dowolnego roku:

iif(month(date())=12 and day(date())=24; "piękny dzień"; "inny dzień") -> "piękny dzień"

IIf([Staż]>20;0,2;[Staż]/100)\*[Podst\_wyn]

Upust: iif([cena]>100; 20%; 10%)

PoUpuscie: [cena] - iif([cena]>100; 20%; 10%) \* [Cena]

PoUpuscie: [Cena] \*(1-iif([cena]>100; 20%; 10%))

|  |  |  |  |
| --- | --- | --- | --- |
| Towar | Cena | Upust: iif([cena]>100; 20%; 10%) | PoUpuscie: [Cena]-iif([cena]>100; 20%; 10%) \* [Cena] |
| AAA | 100 | 0,1 | 90 |
| BBB | 50 | 0,1 | 45 |
| CCC | 150 | 0,2 | 120 |
| DDD | 200 | 0,2 | 160 |
| EEE | 80 | 0,1 | 72 |
| FFF | 75 | 0,1 | 67,5 |

Przetrzymywanie: IIf((Date()-[WYPOŻYCZENIA]![DATA\_WYP])/7>[KATEGORIE]![Limit\_cz]; "!!!"; "")

**Korespondencja seryjna**

Dane czytelnika przetrzymującego książki (informacja nadrzędna) – kwerdena wybierająca

Dane adresata

Wykaz książek przetrzymywanych

1.

2.

3.

4.

Zestawienie książek przetrzymywanych (informacja podrzędna) – kwerenda wybierająca

**Agregaty SQL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Towar | Cena | Cena | Towar | Towar\_prim |
|  | AAA | 100 | 100 | AAA | AAA1 |
|  | BBB | 50 | 50 | BBB | BBB1 |
|  | CCC | 150 | 150 | CCC |  |
|  | DDD | 200 | 200 | DDD | DDD1 |
|  | EEE | 80 | 80 | EEE |  |
|  | AAB | 75 | 75 | FFF | FFF1 |
| Agregaty->> | Ostatni | Suma | Policz | Policz | Policz |
| Wynik | AAB | 655 | 6 | 6 | 4 |

W pustych komórkach jest wartość *Null*.

Zaokrąglenia wyrażeń w kwerendach podsumowujących

Agregat *Średnia* wprowadzony bezpośrednio do pola w kwerendzie podsumowującej, którego wartość jest zaokrąglona (funkcja: *Round*(*wyrażenie; liczba miejsc*)) do 2 miejsc po separatorze części ułamkowej:

Czas\_wyp: Round(Średnia(([ZWROTY]![Data\_zw]-[ZWROTY]![Data\_wyp])\7); 2)

**Filtrowanie w kwerendach podsumowujących**

1. Kolejność: najpierw podsumowania, potem filtr (ma sens, gdy jest grupowanie)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Towar | Cena | Cena | Towar | Towar\_prim |
|  | AAA | 100 | 100 | AAA | AAA |
|  | BBB | 50 | 50 | BBB | BBB |
|  | CCC | 150 | 150 | CCC |  |
|  | DDD | 100 | 100 | DDD | DDD |
|  | EEE | 50 | 50 | EEE |  |
|  | FFF | 100 | 100 | FFF | FFF |
| 1) Agregaty->> | Min | Suma | Grupuj wg | Policz | Policz |
| Wynik | BBB | 100 | 50 | 2 | 1 |
|  | AAA | 300 | 100 | 3 | 3 |
|  | CCC | 150 | 150 | 1 |  |
| 2) Filtr |  |  |  | >1 |  |
| Wynik | BBB | 100 | 50 | 2 | 1 |
|  | AAA | 300 | 100 | 3 | 3 |

2. Kolejność: najpierw filtr, potem podsumowania

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Towar | Cena | Cena | Towar | Towar\_prim |
|  | AAA | 100 | 100 | AAA | AAA |
|  | BBB | 50 | 50 | BBB | BBB |
|  | CCC | 150 | 150 | CCC |  |
|  | DDD | 200 | 200 | DDD | DDD |
|  | EEE | 80 | 80 | EEE |  |
|  | FFF | 75 | 75 | FFF | FFF |
| 1) Filtr (WHERE) |  |  | <100 |  |  |
|  | BBB | 50 | 50 | BBB | BBB |
|  | EEE | 80 | 80 | EEE |  |
|  | FFF | 75 | 75 | FFF | FFF |
| 2) Agregaty->> | Min | Suma | Policz | Policz | Policz |
| Wynik | BBB | 205 | 3 | 3 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Towar | Cena | Cena | Upust: iif([cena]>100; 20%; 10%) | PoUpuscie: [Cena]-iif([cena]>100; 15%; 10%) \* [Cena] |
| AAA | 100 | 100 | 0,1 | 90 |
| BBB | 50 | 50 | 0,1 | 45 |
| CCC | 150 | 150 | 0,2 | 120 |
| DDD | 200 | 200 | 0,2 | 160 |
| EEE | 80 | 80 | 0,1 | 72 |
| FFF | 75 | 75 | 0,1 | 67,5 |
| **Policz** | **Suma** | **Policz** | **Maksimum** | **Minimum** |
| **6** | **655** | **6** | **0,2** | **45** |

Działanie wartości *Null*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Towar | Cena | Cena | Upust: iif([cena]>100; 20%; 10%) | PoUpuscie: [Cena]-iif([cena]>100; 15%; 10%) \* [Cena] |
| AAA | 100 | 100 | 0,1 | 90 |
| BBB | 100 | 100 | 0,1 | 45 |
| CCC | 150 | 150 | 0,2 | 120 |
|  | 200 | 200 | 0,2 | 160 |
| EEE | 80 | 80 | 0,1 | 72 |
| FFF | 75 | 75 | 0,1 | 67,5 |
| **Policz** | **Suma** | **Policz** | **Maksimum** | **Minimum** |
| **5** | **705** | **6** | **0,2** | **45** |

*Null* to nie zero (0), ani nie pusty tekst (””).

Zadanie dla podsumowania danych w bibliotece

Wyznaczyć dla książek w bibliotece zestawienie zawierające: liczbę książek, wartość księgozbioru, najniższą i najwyższą cenę książki. Nazwa kwerendy: *Zestawienie dla biblioteki*.

Różnica między *pierwszy* i *minimum* oraz *ostatni* i *maksimum*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **nr rekordu** | Data | Data | Data | Data |
| **1** | **2019-03-15** | 2019-03-15 | 2019-03-15 | 2019-03-15 |
| **2** | 2018-12-15 | 2018-12-15 | 2018-12-15 | 2018-12-15 |
| **3** | 2015-10-21 | 2015-10-21 | 2015-10-21 | 2015-10-21 |
| **4** | 2020-06-15 | 2020-06-15 | 2020-06-15 | **2020-06-15** |
| **5** | 2010-04-04 | **2010-04-04** | 2010-04-04 | 2010-04-04 |
| **6** | 2020-05-01 | 2020-05-01 | **2020-05-01** | 2020-05-01 |
|  | **Pierwszy** | **Minimum** | **Ostatni** | **Maksimum** |
|  | 2019-03-15 | 2010-04-04 | 2020-05-01 | 2020-06-15 |

**Dwie części merytoryczne kursu nt. Baz danych (dotyczy MS Access)**

* bazodanowa; tworzenia bazy danych (tabele z określeniem pól i typów, klucze główne, powiązania między tabelami), tworzenie zestawień (kwerendy wybierające i podsumowujące, w tym z grupowaniem), manipulowanie danymi (kwerendy funkcjonalne),
* aplikacyjna; formularze, raporty, makra, programy w VBA, strony WWW

**Agregaty domeny**

Wszystkie agregaty domeny mają dwa pierwsze parametry obowiązkowe, trzeci parametr jest opcjonalny (nieobowiązkowy). Wszystkie parametry są wyrażeniami tekstowymi. Parametr *domena* jest zbiorem rekordów przeglądanych przez agregat domeny; może to być tabela lub wynik kwerendy wybierającej.

Jeżeli zbiór rekordów (domena) jest pusty (np. wskutek działania kryterium), to funkcja (dowolna z niżej wymienionych) zwraca wartość *Null*. W przeciwnym przypadku zwraca wartość opisaną poniżej.

* **DLookup(*wyrażenie; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca wartość *wyrażenie* obliczoną na podstawie danych w pierwszym rekordzie domeny.

DLookup("[nazwisko\_a] & ': ' & [tytuł]"; "książki"; "[Syg] = '0001'")

IIf(IsNull(DLookUp("[Syg]"; "[Książki niewypożyczone]"; "[Syg] = Formularze![KSIĄŻKI]![Syg]"));

" Wypożyczona "; " W bibliotece ")

Jeżeli w kwerendzie *Książki niewypożyczone* brak książki o sygnaturze podanej w formularzu, funkcja *DLookup* zwróci wartość *Null*, co oznacza, że książka jest wypożyczona. W przeciwnym przypadku, książka jest niewypożyczona – znajduje się w bibliotece do dyspozycji czytelnika.

* **DSum(*wyrażenie; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca wartość sumy *wyrażenie* obliczoną na podstawie danych ze wszystkich rekordów domeny.

DSum**(**"[Cena]" ; "książki" ; "[Nazwisko\_a] = Forms![KSIĄŻKI]![Nazwisko\_a]"**)**

* **DMin(*wyrażenie; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca wartość minimalną *wyrażenie* obliczoną na podstawie danych ze wszystkich rekordów domeny.

DMin("[Data\_zap]"; "Czytelnicy")

* **DMax(*wyrażenie; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca wartość maksymalną *wyrażenie* obliczoną na podstawie danych ze wszystkich rekordów domeny.

DMax("[Data\_wyp]"; "wypożyczenia")

DMax("([Data\_zw]-[Data\_wyp])\7","[Zwroty]")

* **DCount(*wyrażenie; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca liczbę niepustych wartości *wyrażenie* ze zbioru rekordów domeny.

DCount("[Syg]"; "Zwroty")

* **DAvg(wyrażenie*; domena*** [***; kryteria***] **)**

Jeżeli zbiór rekordów jest niepusty, funkcja zwraca średnią wartości *wyrażenie* ze zbioru rekordów domeny.

DAvg("([Data\_zw]-[Data\_wyp])\7","[Zwroty]")