Systemy baz danych – sprawozdanie

Wykonali : Jarosław Dąbrowski, Artur Disiński

Grupa : I8E1S4

Temat : Labolatorium 2 – Obiektowa baza danych

1. Opis tematu

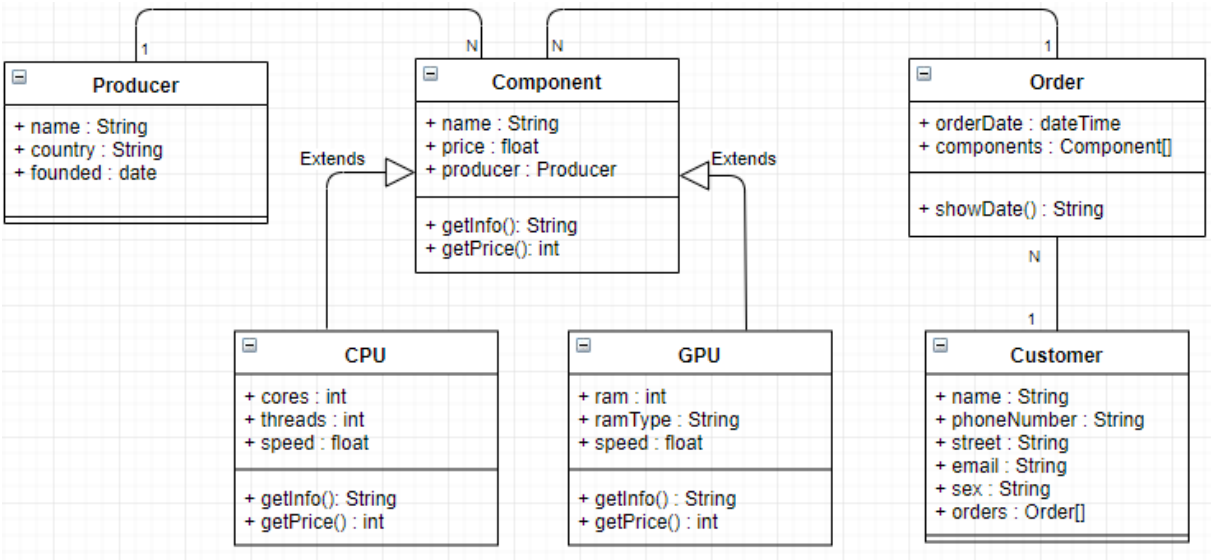
Obiektowa baza danych przechowuje informacje o firmie zajmującej się sprzedażą podzespołów komputerowych za pośrednictwem Internetu. Podstawowe założenia:

- Klient (Customer) na przestrzeni czasu może składać wiele zamówień (Order), gdzie każde zamówienie może dotyczyć wielu podzespołów (Component)
- Podzespoły dzielą się na procesory (CPU) oraz karty graficzne (GPU)
- Każdy komponent można przydzielić do konkretnego producenta. W rozpatrywanym przypadku podzespół może mieć tylko jednego producenta

2. Model bazy danych

Poniżej przedstawiono model bazy danych wraz z opisem wszystkich klas, metod i atrybutów:

Rys 1. Model bazy danych



Producer – klasa przechowująca informacje o producentach podzespołów.

Pole	Typ	Opis
name	String	Nazwa producenta. Wartość unikalna
country	String	Kraj założenie producenta
founded	date	Data założenia firmy

Component – klasa abstrakcyjna reprezentująca podzespół komputerowy.

Pole	Typ	Opis
name	String	Nazwa podzespołu komputerowego
price	float	Cena podzespołu komputerowego
producer	Producer	Jaki producent wyprodukował podzespół. Jest to relacja do klasy „Producer”
getInfo()	Metoda - String	Metoda abstrakcyjna. Odpowiada za zwracanie informacji o podzespole w postaci String
getPrice()	Metoda - String	Metoda abstrakcyjna. Odpowiada za zwrócenie ceny podzespołu

CPU – procesor, typ podzespołu komputerowego, dziedziczy z klasy „Component”

Pole	Typ	Opis
cores	int	Ilość rdzeni procesora
threads	int	Ilość wątków procesora
speed	float	Częstotliwość taktowania procesora [GHz]
getInfo()	Metoda - String	Metoda zwracająca podstawowe informacje na temat procesora w postaci String
getPrice()	Metoda - String	Metoda zwracająca cenę procesora

GPU – karta graficzna, typ podzespołu komputerowego, dziedziczy z klasy „Component”

Pole	Typ	Opis
ram	int	Ilość pamięci RAM
ramType	String	Rodzaj pamięci RAM. W bazie podano listę możliwych wartości : GDDR5, GDDR4, GDDR3, GDDR2, SDRAM, FPMRAM
speed	float	Taktowanie rdzenia GPU [GHz]
getInfo()	Metoda - String	Metoda zwracająca podstawowe informacje na temat karty graficznej w postaci String
getPrice()	Metoda - String	Metoda zwracająca cenę karty graficznej

Order – klasa przechowująca informacje o zamówieniach. Każde zamówienie może zawierać w sobie wiele podzespołów, oraz każde zamówienie zawsze dotyczy jednego klienta.

Pole	Typ	Opis
orderDate	dateTime	Data oraz dokładny czas złożenia zamówienia
components	Component[]	Lista podzespołów wchodzących w skład zamówienia. Jest to relacja 1-wiele do klasy „Component”
showDate()	Metoda - String	Metoda wyświetlająca informację o dacie i godzinie

Customer – klasa przechowująca informacje o klientach sklepu internetowego.

Pole	Typ	Opis
name	String	Nazwa użytkownika
phoneNumber	String	Numer telefonu klienta
street	String	Adres zamieszkania klienta
email	String	Adres email klienta
orders	Order[]	Lista zamówień złożonych przez klienta. Jest to relacja 1-wiele do klasy „Order”

3. Implementacji bazy danych

Implementacja obiektowej bazy danych została zrealizowana przy pomocy oprogramowania Cache Intersystems. Dokładna implementacja bazy danych jest dostępna w dołączonym pliku „SBD_LAB2.xml” oraz w repozytorium Github : https://github.com/jdabrowski11926/SBD-LAB-2

4. Wypełnienie bazy danych danymi testowymi

Baza danych została wypełniona przy pomocy metody „Populate” z klasy „Populate”. Poniżej przedstawiono zestaw komend odpowiedzialnych za wypełnienie bazy danymi oraz późniejsze wyświetlenie:

```
Do $SYSTEM.SQL.Shell()
DELETE FROM Sbd.GPU
DELETE FROM Sbd.CPU
DELETE FROM Sbd.Order
DELETE FROM Sbd.Producer
DELETE FROM Sbd.Customer
q
Do ##class(Sbd.Customer).Populate(10)
Do ##class(Sbd.Producer).Populate(10)
Do ##class(Sbd.Order).Populate(10)
Do ##class(Sbd.GPU).Populate(10)
Do ##class(Sbd.CPU).Populate(10)
Do $SYSTEM.SQL.Shell()
SELECT * FROM Sbd.GPU
SELECT * FROM Sbd.CPU
SELECT * FROM Sbd.Order
SELECT * FROM Sbd.Producer
SELECT * FROM Sbd.Customer
```

Rys 2. Wyświetlenie zawartości tabeli GPU

```
USER>>SELECT * FROM Sbd.GPU
6.      SELECT * FROM Sbd.GPU

ID      name    order   price   producer        ram     ramType speed
2118    RTX TURBO       346     5573    353     6       FPMRAM  1385
2119    AXD     348     6587    353     10      SDRAM   1115
2120    RTX 999999 Ti   345     7190    356     4       SDRAM   925
2121    GTX 1080        347     475     360     7       GDDR4   851
2122    GTX 1080        343     5803    354     15      SDRAM   2404
2123    RTXXX   343     6302    352     5       FPMRAM  2959
2124    GTX 0   344     2302    358     9       GDDR5   1439
2125    GTX 0   350     24      353     10      GDDR5   696
2126    RTX 999999 Ti   345     8971    357     9       GDDR4   2270
2127    RTX TURBO       342     3526    355     9       FPMRAM  1011
```

5. Przykładowe zastosowania – terminal

Poniżej przedstawiono zestaw komend odpowiedzialnych za dodawanie nowego klienta. Na początku dodano niezgodne dane (customer.sex = „M”, gdzie dostępne wartości to „Male” i „Female”) co spowodowało wyświetlenie błędu. Po zmianie pola na dozwoloną wartość rekord klienta został dodany do tabeli „Customer”.

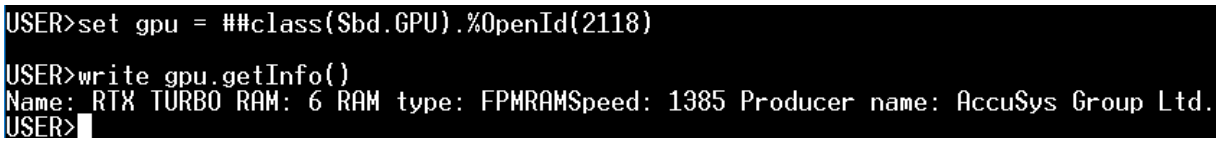
```
q
Set customer = ##class(Sbd.Customer).%New()
set customer.emailAddress = "abcie@mail.pl"
set customer.name = "Robert Clown"
set customer.phoneNumber = "123123123"
set customer.sex = "M"
set customer.street = "Abcde"
Set sc = customer.%Save()
zwrite sc
set customer.sex = "Male"
Set sc = customer.%Save()
zwrite sc
```

Rys 3. Efekt działania kodu odpowiedzialnego za dodanie klienta do bazy danych

```
USER>q

USER>Set customer = ##class(Sbd.Customer).%New()

USER>set customer.emailAddress = "abcie@mail.pl"

USER>set customer.name = "Robert Clown"

USER>set customer.phoneNumber = "123123123"

USER>set customer.sex = "M"

USER>set customer.street = "Abcde"

USER>Set sc = customer.%Save()

USER>zwrite sc
sc="0 "_$lb($lb(7205,"M",",Male,Female",,,,,,,$lb("zsexIsValid+1^Sbd.Customer.1","USER",$lb("e^zsexIsValid+1^Sbd.Customer.1^1","e^%V
alidateObject+7^Sbd.Customer.1^4","e^%SerializeObject+3^Sbd.Customer.1^1","e^%Save+8^Sbd.Customer.1^5","e^^0")),"0 "_$lb($lb(5802,"
Sbd.Customer:sex","M",,,,,,,$lb("EmbedErr+1^%occSystem","USER",$lb("e^EmbedErr+1^%occSystem^1"))))))/* ERROR #7205: Datatype value '
M' not in VALUELIST ',Male,Female'-   > ERROR #5802: Datatype validation failed on property 'Sbd.Customer:sex', with value equal to
"M" */

USER>set customer.sex = "Male"

USER>Set sc = customer.%Save()

USER>zwrite sc
sc=1
```

Poniżej przedstawiono przykład wywołania metody w klasie GPU odpowiedzialnej za wyświetlanie informacji:

Rys 4. Przykład wywołania metody z obiektu GPU



6. Podsumowanie i wnioski

Napotkane problemy i ich rozwiązania

Głównym problemem przy korzystaniu z Cache Intersystems była nietypowa składnia języka Cache. Przykładowo łączenie ciągu znaków odbywa się przy pomocy znaku ”_” a indeksowanie elementów w tablicy lub liście rozpoczyna się od elementu ”1” (a nie ”0” jak to jest przyjęte w większości języków programowania).

Ocena środowiska

Praca w środowisku Cache Intersystem jest trudna ze względu na jego małą popularność co wiąże się z trudnościami w wyszukiwaniu informacji na jego temat.

Jak środowisko realizuje zagadnienia związane z:	Odpowiedź
Obiektowością - Metodami	Dobrze, nie napotkano się na żadne problemy
Obiektowością – Dziedziczeniem, typami abstrakcyjnymi	W definicji klasy wystarczy dodać słowo kluczowe [Abstract]. Nie można stworzyć instancji klasy abstrakcyjnej.
Obiektowością – Związkami między klasami (asocjacja, kompozycja)	Można realizować ją za pomocą relacji (Relationship) Relationship orders As Sbd.Order
Obiektowością – Typy danych – proste, złożone	Można tworzyć pola w metodach, które przyjmują określone wartości. Można to uznać za szczególny typ danych. W naszym projekcie przykładem na to może być pole ”sex” w klasie Customer: Property sex As %String(VALUELIST = ",Male,Female");
Obiektowością - Polimorfizmem	Klasy dziedziczące mogą mieć metody oraz parametry o tych samych nazwach co klasa, z której dziedziczą. W naszym projekcie przykładem na to może być pole „name” albo metoda „getInfo()”
Obiektowością – Tożsamością danych	Każde pole z każdej tabeli otrzymuje specjalny identyfikator
Obiektowością – Enkapsulacją	Istnieje możliwość tworzenia metod prywatnych dodając słowo klucz [Private]
Obiektowością – Trwałością danych	Klasy dziedziczące z %Persistent mogą być zapisywane w bazie danych
Administracja – Zarządzanie środowiskiem	Środowisko Cache Intersystem może być zarządzane przez aplikację Management Portal

Interfejs – Czy narzędzie posiada API? Dla jakich języków?	Istnieje możliwość utworzenia metod korzystając z języków programowania: Cache, Java, javascript, Basic, mvbasic, TSQL. W projekcie został wykorzystany jedynie język Cache
Środowisko – Czy narzędzie zawiera w sobie środowisko programistyczne?	Tak, środowisko programistyczne znajduje się w aplikacji Cache Studio
Skalowalność – Czy narzędzie umożliwia horyzontalne skalowanie środowiska? (rozproszone przetwarzanie, magazynowanie, replikacja)	Aby poprawić szybkość i wydajność dostępu użytkowników do danych, InterSystems IRIS może korzystać z rozproszonego buforowania. Dane użytkownika mogą być przechowywane na wielu serwerach.
Multi-model – Czy narzędzie zapewnia inne rodzaje bazy danych?	Wg oficjalnej strony Cache InterSystems, Cahce jest również w pełni funkcjonalną relacyjną bazą danych *„In addition to being a high-performance object database, Caché is also a full-featured relational database”*

7. Linki

Projekt w repozytorium Github : https://github.com/jdabrowski11926/SBD-LAB-2

Filmik z działania aplikacji : https://www.youtube.com/watch?v=B04XoW5Xvqs