

Politechnika Świętokrzyska

Wydział elektrotechniki automatyki i informatyki

Technologie obiektowe – Projekt

Temat: Porównanie produktów bazodanowych firmy Oracle.

Wykonali: Przemysław Postrach i Piotr Kaczmarczyk

1. Instalacja baz danych i SQL Developera

Oracle SQL database.

Plik instalacyjny został pobrany ze strony:

<https://www.oracle.com/pl/database/technologies/oracle-database-software-downloads.html>

Proces przebiegał zgodnie z instalatorem i wyświetlanymi przez niego komunikatami. Utworzona baza posiada domyślny SID pozwalający nam szybkie i łatwe korzystanie z bazy.

Oracle NoSQL database

Plik instalacyjny został pobrany z oficjalnej strony producenta:

<https://www.oracle.com/database/technologies/nosql-database-server-downloads.html>

Uruchomienie tej bazy wymaga środowiska linux oraz zainstalowanej javy. Po rozpakowaniu paczki ze strony wystarczy wejść w główny folder i użyć komendy "java -Xmx64m -Xms64m -jar <KVHOME>/lib/kvclient.jar".

Po uruchomieniu powinniśmy w konsoli widzieć następujący komunikat: "11gR2.M.N.O (....)".

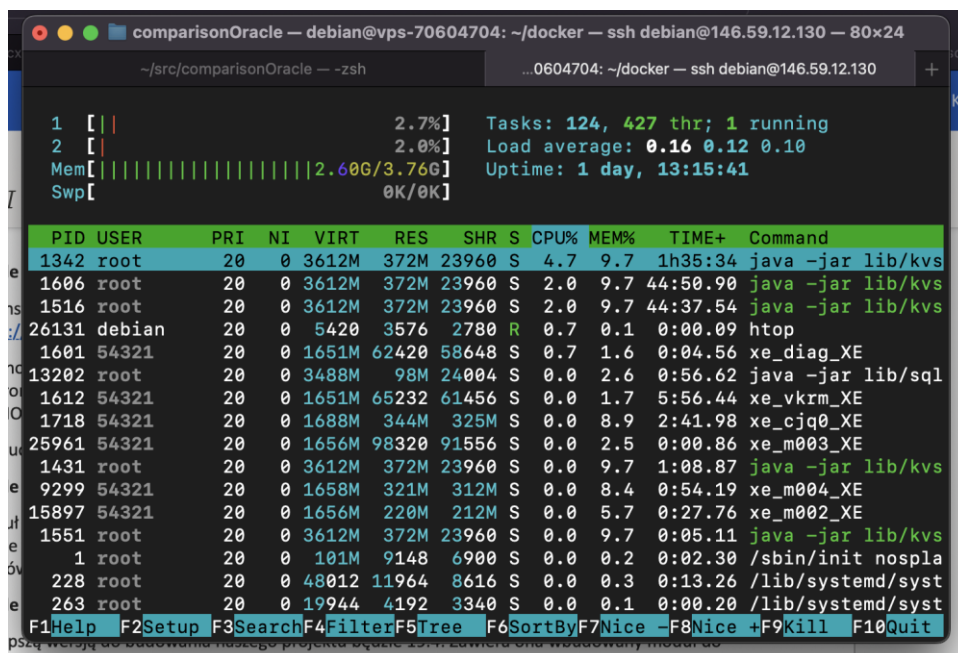
Oracle Object Database

Moduł obiektowy jest integralną częścią Oracle SQL Database od wersji 8, każde kolejne wydanie Oracle SQL database wprowadza nowe funkcjonalności zbliżające ten moduł do obiektowych języków programowania Java czy C#.

Oracle SQL Developer – 19.4

Najlepszą wersją do budowania naszego projektu będzie 19.4. Zawiera ona wbudowany moduł do baz Oracle NoSQL, który bardzo ułatwi nam tworzenie aplikacji.

2. Platforma testowa



The screenshot shows a terminal window with the title "comparisonOracle — debian@vps-70604704: ~/docker — ssh debian@146.59.12.130 — 80x24". The terminal displays system status information and a list of running processes.

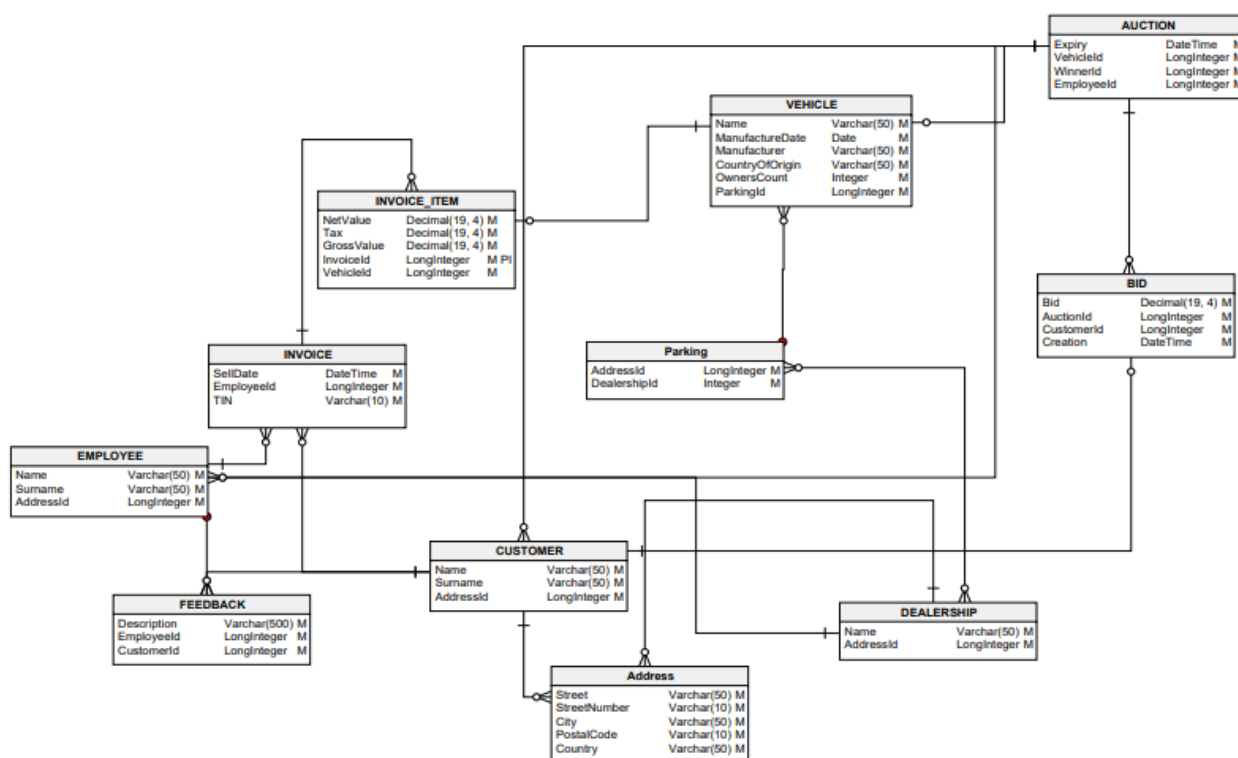
```
1 [||| 2.7%] Tasks: 124, 427 thr; 1 running
2 [||| 2.0%] Load average: 0.16 0.12 0.10
Mem[|||||2.60G/3.76G] Uptime: 1 day, 13:15:41
Swp[|||||0K/0K]
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1342	root	20	0	3612M	372M	23960	S	4.7	9.7	1h35:34	java -jar lib/kvs
1606	root	20	0	3612M	372M	23960	S	2.0	9.7	44:50.90	java -jar lib/kvs
1516	root	20	0	3612M	372M	23960	S	2.0	9.7	44:37.54	java -jar lib/kvs
26131	debian	20	0	5420	3576	2780	R	0.7	0.1	0:00.09	htop
1601	54321	20	0	1651M	62420	58648	S	0.7	1.6	0:04.56	xe_diag_XE
13202	root	20	0	3488M	98M	24004	S	0.0	2.6	0:56.62	java -jar lib/sql
1612	54321	20	0	1651M	65232	61456	S	0.0	1.7	5:56.44	xe_vkrm_XE
1718	54321	20	0	1688M	344M	325M	S	0.0	8.9	2:41.98	xe_cjq0_XE
25961	54321	20	0	1656M	98320	91556	S	0.0	2.5	0:00.86	xe_m003_XE
1431	root	20	0	3612M	372M	23960	S	0.0	9.7	1:08.87	java -jar lib/kvs
9299	54321	20	0	1658M	321M	312M	S	0.0	8.4	0:54.19	xe_m004_XE
15897	54321	20	0	1656M	220M	212M	S	0.0	5.7	0:27.76	xe_m002_XE
1551	root	20	0	3612M	372M	23960	S	0.0	9.7	0:05.11	java -jar lib/kvs
1	root	20	0	101M	9148	6900	S	0.0	0.2	0:02.30	/sbin/init nospla
228	root	20	0	48012	11964	8616	S	0.0	0.3	0:13.26	/lib/systemd/syst
263	root	20	0	19944	4192	3340	S	0.0	0.1	0:00.20	/lib/systemd/syst

At the bottom of the terminal, there are keyboard shortcuts for navigating the htop interface: F1Help, F2Setup, F3Search, F4Filter, F5Tree, F6SortBy, F7Nice, F8Nice, F9Kill, F10Quit.

Test został przeprowadzony na maszynie wirtualnej z zainstalowanym systemem linux debian 11. Maszyna jest wyposażona w dysk SSD, 2 rdzenie procesora oraz 4 GB pamięci RAM.

3. Testowa baza danych



Przedstawiona baza danych została zrobiona dla komisji samochodowej z możliwością licytacji aut.

Przykładowe dane zostały wygenerowane przy użyciu strony: <https://www.mockaroo.com/>

4. Porównanie baz danych

	Tworzenie schematu
SQL	<pre>CREATE TABLE VEHICLE (Id integer NOT NULL, Name varchar2(50) NOT NULL, ManufactureDate date NOT NULL, Manufacturer varchar2(50) NOT NULL, CountryOfOrigin varchar2(50) NOT NULL, OwnersCount integer NOT NULL, ParkingId integer NOT NULL, CONSTRAINT VEHICLE_pk PRIMARY KEY (Id) CONSTRAINT VEHICLE_Parking FOREIGN KEY (ParkingId) REFERENCES Parking (Id));</pre>
NoSQL	<pre>CREATE TABLE VEHICLE (Id integer, Name string, Manufacturedate string,</pre>

	<p>Manufacturer string, CountryOfOrigin string, OwnersCount integer, ParkingId integer, PRIMARY KEY (Id));</p>
Object	<pre> CREATE OR REPLACE TYPE faktura AS OBJECT (DATA DATE, KWOTANETTO DECIMAL, CONSTRUCTOR FUNCTION faktura(DATA DATE) RETURN SELF AS RESULT, CONSTRUCTOR FUNCTION faktura(DATA DATE, KWOTANETTO DECIMAL) RETURN SELF AS RESULT, MEMBER FUNCTION STAWKAVAT RETURN INT, MEMBER FUNCTION WYLICZBRUTTO RETURN DECIMAL, MEMBER PROCEDURE DRUKUJ) not final; CREATE OR REPLACE TYPE BODY faktura AS CONSTRUCTOR FUNCTION faktura(DATA DATE) RETURN SELF AS RESULT IS BEGIN SELF.DATA := DATA; RETURN; END; CONSTRUCTOR FUNCTION faktura(DATA DATE, KWOTANETTO DECIMAL) RETURN SELF AS RESULT IS BEGIN SELF.DATA := DATA; SELF.KWOTANETTO := KWOTANETTO; RETURN; END; MEMBER FUNCTION STAWKAVAT RETURN INT IS BEGIN RETURN 1; END; MEMBER FUNCTION WYLICZBRUTTO RETURN DECIMAL IS BEGIN RETURN KWOTANETTO * ((STAWKAVAT / 100) + 1); END; MEMBER PROCEDURE DRUKUJ IS BEGIN DBMS_OUTPUT.PUT_LINE(''); END; END; CREATE OR REPLACE TYPE polskafaktura UNDER faktura (OVERRIDING MEMBER FUNCTION STAWKAVAT RETURN INT, OVERRIDING MEMBER PROCEDURE DRUKUJ) instantiable final; </pre>

```

CREATE OR REPLACE TYPE BODY polskafaktura AS
    OVERRIDING MEMBER FUNCTION STAWKAVAT RETURN INT IS
    BEGIN
        RETURN 23;
    END;
    OVERRIDING MEMBER PROCEDURE DRUKUJ IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE(SELF.DATA ||
                                ' Faktura PL ' ||
                                SELF.KWOTANETTO ||
                                ' Stawka VAT: ' ||
                                SELF.STAWKAVAT ||
                                ' Kwota Brutto: ' ||
                                SELF.WYLICZBRUTTO);
    END;
END;

CREATE OR REPLACE TYPE niemieckafaktura UNDER faktura
(
    OVERRIDING MEMBER FUNCTION STAWKAVAT RETURN INT,
    OVERRIDING MEMBER PROCEDURE DRUKUJ
) instantiable final;

CREATE OR REPLACE TYPE BODY niemieckafaktura AS
    OVERRIDING MEMBER FUNCTION STAWKAVAT RETURN INT IS
    BEGIN
        RETURN 19;
    END;
    OVERRIDING MEMBER PROCEDURE DRUKUJ IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE(SELF.DATA ||
                                ' Faktura DE ' ||
                                SELF.KWOTANETTO ||
                                ' Stawka VAT: ' ||
                                SELF.STAWKAVAT ||
                                ' Kwota Brutto: ' ||
                                SELF.WYLICZBRUTTO);
    END;
END;

CREATE TABLE tabela_faktura OF faktura;

```

	Create
SQL	INSERT INTO VEHICLE (ID, NAME, MANUFACTUREDATE, MANUFACTURER, COUNTRYOFORIGIN, OWNERSCOUNT, PARKINGID) VALUES (1, 'I', to_date('2/4/1960', 'MM/DD/RRRR'), 'Infiniti', 'Ukraine', 8, 56);
NoSQL	INSERT INTO VEHICLE (ID, NAME, MANUFACTUREDATE, MANUFACTURER, COUNTRYOFORIGIN, OWNERSCOUNT, PARKINGID) VALUES (1, 'I', '2/4/1960', 'Infiniti', 'Ukraine', 8, 56);

Object	INSERT INTO tabela_faktura VALUES (polskafaktura(SYSDATE, 100)); INSERT INTO tabela_faktura VALUES (niemieckafaktura(SYSDATE, 100));
---------------	---

	Read
SQL	SELECT * FROM AUCTION; SELECT COUNT(*) FROM AUCTION;
NoSQL	SELECT * FROM AUCTION; SELECT COUNT(*) FROM AUCTION;
Object	SELECT f.*, f.STAWKAVAT(), f.WYLICZBRUTTO() FROM tabela_faktura f; SELECT COUNT(*) FROM TABELA_FAKTURA;

	Update
SQL	UPDATE customer SET Surname = 'Anderson' WHERE id = 1000;
NoSQL	UPDATE customer SET Surname = 'Anderson' WHERE id = 1000;
Object	UPDATE tabela_faktura SET KWOTANETTO = 200 WHERE KWOTANETTO = 100;

	DELETE
SQL	DELETE FROM VEHICLE;
NoSQL	DELETE FROM VEHICLE;
Object	DELETE FROM TABELA_FAKTURA;

	Drop
SQL	DROP TABLE PARKING cascade constraints;
NoSQL	DROP TABLE PARKING;
Object	DROP TABLE TABELA_FAKTURA; DROP TYPE FAKTURA; DROP TYPE BODY FAKTURA;

5. Mierzenie wydajności zapytań

SELECT 1 - select * from customer where lower(name) like '%a%';

Zwraca wszystkich klientów których imię zawiera literę "a"

SELECT 2 - select avg(bid), customerid from bid group by customerid;

SELECT 3 -

SELECT *
FROM CUSTOMER C

```

LEFT JOIN AUCTION A2 on C.ID = A2.WINNERID
LEFT JOIN BID B on C.ID = B.CUSTOMERID and A2.ID = B.AUCTIONID
LEFT JOIN VEHICLE V on A2.VEHICLEID = V.ID
LEFT JOIN PARKING P on V.PARKINGID = P.ID
LEFT JOIN ADDRESS A3 on C.ADDRESSID = A3.ID
WHERE lower(A3.CITY) LIKE '%a%' AND lower(V.NAME) LIKE '%inf%';

```

Zwraca średnią ofertę licytacji dla każdego klienta

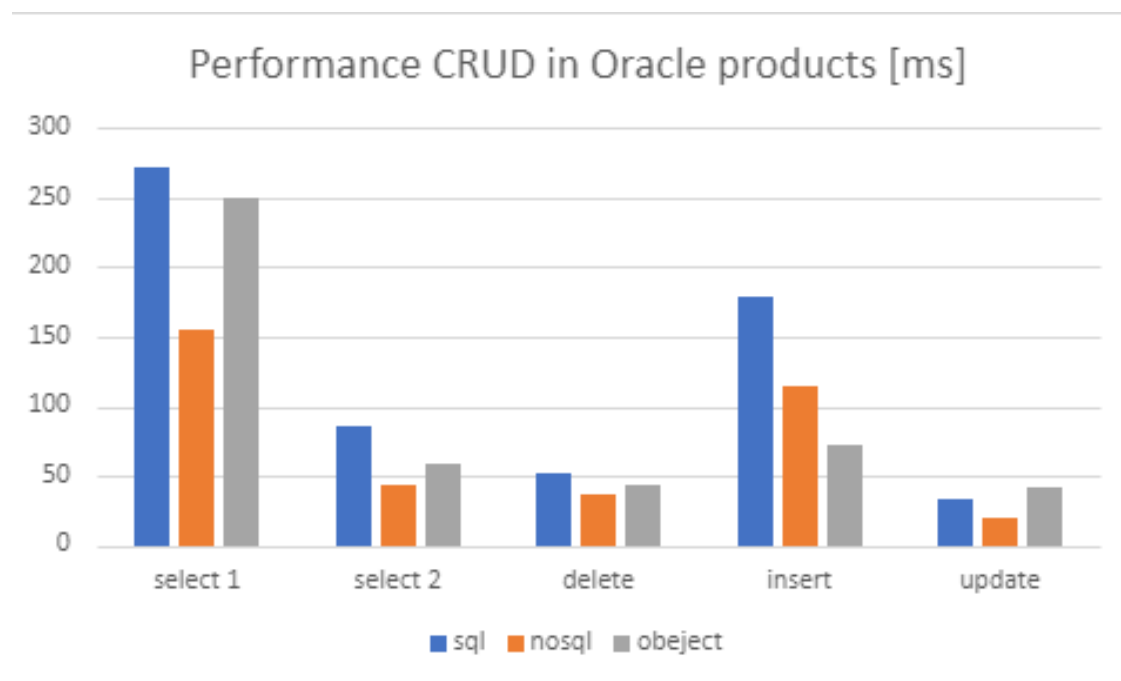
DELETE - Usunięcie 1000 rekordów

INSERT – Dodanie 1000 rekordów

UPDATE – Modyfikacja 1 pola 1000 rekordów

Zaprezentowane dane przedstawiają średnią z 5 pomiarów jednostkowych.

ms	sql	nosql	object
select 1	272 ms	155 ms	250 ms
select 2	86 ms	44 ms	59 ms
delete	52 ms	37 ms	43 ms
insert	179 ms	114 ms	71 ms
update	34 ms	20 ms	42 ms



6. Unikalne funkcjonalności dla danego typu baz danych

Oracle SQL

Największym atutem bazy danych SQL jest jej nakierowanie na operacje na danych tabelarycznych. Dzięki rozbudowanemu silnikowi bazy możemy w szybki sposób wyciągać dane. Społeczność zgromadzona wokół tego typu bazy pomaga znaleźć optymalne rozwiązania naszych problemów.

Oracle NoSQL

Baza danych NoSql charakteryzuje się przede wszystkim innymi typami niż SQL. Są one zbliżone do tych znanych nam z Javy czy C#. Dzięki temu osobą programującym łatwiej stworzyć taki system. Sama forma danych jakie otrzymujemy na formę JSON. Pomaga to potem pracować na danych, ponieważ taką samą formę możemy otrzymać budując interfejsy API. Baza Oracle NoSql jest podobna charakterystyką do innych baz tego typu. Wyróżnia ją przede wszystkim składnia bardzo podobna do swojego SQLowego odpowiednika.

Oracle Object Database

Obiektowa baza danych cechuje się przechowywaniem obiektów jako podstawowa jednostka danych. Zaletą tego rozwiązania jest możliwość zawarcia logiki biznesowej bezpośrednio w obiektach przy użyciu funkcji, procedur oraz właściwości, w projekcie został stworzony abstrakcyjny typ faktury, oraz typy polskiej faktury i niemieckiej faktury. Dwie implementacje nadpisywały logikę z faktury abstrakcyjnej powodując, że logika biznesowa nie wychodzi poza obiekty encji. Dzięki zastosowaniu takiego podejścia możemy w łatwy sposób reprezentować obiekty z języków Java czy C# w postaci obiektów bazodanowych, co skraca czas wytwarzania oprogramowania. Ponadto obiektowa baza danych oracle pomaga nam zachować dobre reguły programowania, np. KISS, DRY, SOLID.

7. Wnioski

Porównując te trzy produkty od firmy Oracle doskonale widzimy w jakim celu zostały one stworzone i do jakiego klienta trafiają najlepiej.

Jeśli potrzebujemy mieć szybki dostęp do mocno połączonych ze sobą danych to wybór bazy SQL będzie najlepszy. Z tej trójki jest to najbardziej dojrzała technologia z wieloletnim wsparciem i rozwojem developerów. Sprawdzi się dobrze dla ustrukturyzowanych danych, które zawsze posiadają takie same kolumny.

W sytuacji, kiedy mamy dane nieustandaryzowane, które nie posiadają połączeń między sobą to najlepszym rozwiązaniem będzie wybór NoSQL. Niestety w przypadku produktu Oracle mamy bardzo małą społeczność programistów i dokumentacji od producenta. Nie jest ona tak rozbudowana jak dokumentacja MongoDB czy Firebase. Powoduje to małą jej popularność i ilość dostępnych opcji. Plusem używania Oracle NoSQL na tle konkurencji jest na podobna składnia do SQLa. Dzięki temu programista może łatwo spróbować nowej technologii.

Baza obiektowa najlepiej sprawdzi się w momencie, kiedy potrzeba w bazie przechowywać dużą ilość obiektów i zawartej w niej logiki biznesowej. Niestety technologia ta nie znalazła szerokiego zastosowania przez co nie posiada bogatej społeczności ani dobrej dokumentacji jak np. MySQL.

Jak widzimy produkty te posiadają różne zastosowania i przeznaczenie. Dlatego bezpośrednie porównanie performance tych baz jest nieistotne. Przy dużej ilości danych ze względu na dojrzałość technologii wygra SQL. Jeśli będziemy mierzyć łatwość implementacji w aplikacji webowej i dostęp do poszczególnych danych to wyjdzie na prowadzenie NoSQL. Gdy dane są obiektami złożonymi to najlepiej będzie użyć podejścia obiektowego w naszych danych.