Etap 3B

Grupa B6 - Loty

✈️ Hanna Grodzicka

🇧🇷 Mateusz Najda

Rozszerzenia

1. Tabela `Flight`
   - rejestracja schematu

```
DBMS_XMLSCHEMA.registerSchema('flightSchema.xsd',
'<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Flight">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="FlightNumber"/>
        <xs:element type="xs:string" name="DestinationCountry"/>
        <xs:element type="xs:string" name="DestinationCity"/>
        <xs:element type="xs:dateTime" name="Departure"/>
        <xs:element type="xs:dateTime" name="Arrival"/>
        <xs:element type="xs:integer" name="Passengers"/>
        <xs:element type="xs:string" name="Status"/>
        <xs:element type="xs:decimal" name="Delay"/>
        <xs:element type="xs:boolean" name="IsNational"/>
        <xs:element type="xs:integer" name="PlaneId"/>
        <xs:element type="xs:integer" name="AirportId"/>
      </xs:sequence>
      <xs:attribute type="xs:positiveInteger" name="id" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>'
);
end;
```

   - utworzenie tabeli

```
CREATE TABLE Flight_xml OF XMLType
XMLSCHEMA "flightSchema.xsd" ELEMENT "Flight";

alter table Flight_xml add constraint flight_pk primary key (XMLDATA."id");

alter table Flight_xml add foreign key (XMLDATA."PlaneId") references Plane_xml (XMLDATA."id") on delete cascade;
alter table Flight_xml add foreign key (XMLDATA."AirportId") references Airport_xml (ID) on delete cascade;
```

   - przekopiowanie danych z tabeli `Flight` do `Flight_xml`

```
INSERT INTO Flight_xml
SELECT XMLElement("Flight",
                    XMLAttributes(f.ID as "id"),
                    XMLElement("FlightNumber",f.FLIGHT_NUMBER),
                    XMLElement("DestinationCountry",f.DESTINATION_COUNTRY),
                    XMLElement("DestinationCity",f.DESTINATION_CITY),
                    XMLElement("Departure",f.DEPARTURE),
                    XMLElement("Arrival",f.ARRIVAL),
                    XMLElement("Passengers",f.PASSENGERS),
                    XMLElement("Status",f.STATUS),
                    XMLElement("Delay",f.DELAY),
                    XMLElement("IsNational",f.IS_NATIONAL),
                    XMLElement("PlaneId",f.PLANE_ID),
                    XMLElement("AirportId",f.AIRPORT_ID)
                   ) AS "result"
FROM Flight f;
```

2. Tabela `Plane`
   - rejestracja schematu

```
begin
DBMS_XMLSCHEMA.registerSchema('planeSchema.xsd',
'<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Plane">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="Manufacturer"/>
        <xs:element type="xs:string" name="Model"/>
        <xs:element type="xs:date" name="ProductionDate"/>
        <xs:element type="xs:integer" name="SeatCount"/>
        <xs:element type="xs:decimal" name="FuelCapacity"/>
        <xs:element type="xs:integer" name="CrusingRange"/>
        <xs:element type="xs:integer" name="AirlineId"/>
      </xs:sequence>
      <xs:attribute type="xs:positiveInteger" name="id" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>'
);
end;
```

   - utworzenie tabeli

```
CREATE TABLE Plane_xml OF XMLType
XMLSCHEMA "planeSchema.xsd" ELEMENT "Plane";

alter table Plane_xml add constraint plane_pk primary key (XMLDATA."id");

alter table Plane_xml add foreign key (XMLDATA."AirlineId") references Airline (ID) on delete cascade;
```

   - przekopiowanie danych z tabeli `Plane` do `Plane_xml`

```
INSERT INTO Plane_xml
SELECT XMLElement("Plane",
                  XMLAttributes(p.ID as "id"),
                  XMLElement("Manufacturer",p.MANUFACTURER),
                  XMLElement("Model",p.MODEL),
                  XMLElement("ProductionDate",p.PRODUCTION_DATE),
                  XMLElement("SeatCount",p.SEAT_COUNT),
                  XMLElement("FuelCapacity",p.FUEL_CAPACITY),
                  XMLElement("CrusingRange",p.CRUISING_RANGE),
                  XMLElement("AirlineId",p.AIRLINE_ID)
                 ) AS "result"
FROM Plane p;
```

3. Tabela `Airport` (kolumna `Location`)
   - rejestracja schematu

```
begin
DBMS_XMLSCHEMA.registerSchema('locationSchema.xsd',
'<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Location">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Coordinates">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:float" name="Longitude"/>
              <xs:element type="xs:float" name="Latitude"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element type="xs:string" name="Country"/>
        <xs:element type="xs:string" name="City"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>'
```

```
);
end;
```

- utworzenie tabeli

```
create table Airport_xml (
  ID        number(10) not null,
  NAME      varchar2(255) not null,
  CODE      varchar2(255) not null unique,
  LOCATION  XMLTYPE not null,
  primary key (ID))
XMLTYPE COLUMN LOCATION
XMLSCHEMA "locationSchema.xsd" ELEMENT "Location";
```

- przekopiowanie danych z tabeli Airport do Airport_xml

```
INSERT INTO Airport_xml
SELECT a.ID, a.NAME, a.CODE, XMLElement("Location",
                             XMLElement("Coordinates",XMLElement("Longitude", a.LONGITUDE), XMLElement("Latitude", a.LATITU
                             XMLElement("Country",a.COUNTRY),
                             XMLElement("City",a.CITY)
                             )
FROM Airport a;
```

Operacje

1. **Procent lotów opóźnionych**

   Opis: Stosunek liczby lotów opóźnionych (o dowolną dodatnią wartość) do liczby wszystkich lotów w historii danej linii lotniczej.

   Rozszerzenie schematu: rozszerzenie 1 i 2

   Operacja SQL: SELECT (operacja 1 z zestawu A)

   ```
   SELECT SUM(CASE WHEN f.delay > 0 THEN 1 ELSE 0 END) / COUNT(*)
   FROM flight f
     JOIN plane p ON p.id = f.plane_id
     JOIN airline a ON a.id = p.airline_id
     INNER JOIN flight_employee fe ON f.id = fe.flight_id
     INNER JOIN employee e ON e.id = fe.employee_id
   GROUP BY a.id;
   ```

   Implementacja XML: PL/SQL z wyrażeniem `extractValue`

   ```
   SELECT SUM(CASE WHEN extractValue(value(f), '//Delay') > 0 THEN 1 ELSE 0 END) / COUNT(*)
   FROM Flight_xml f
     JOIN Plane_xml p ON extractValue(value(p), '//@id') = extractValue(value(f), '//PlaneId')
     JOIN airline a ON a.id = extractValue(value(p), '//AirlineId')
     INNER JOIN flight_employee fe ON extractValue(value(f), '//@id') = fe.flight_id
     INNER JOIN employee e ON e.id = fe.employee_id
   GROUP BY a.id;
   ```

2. **Liczba lotów do Brazylii**

   Opis: Liczba lotów na wszystkich lotniskach, których lokalizacja zawiera się pomiędzy długościami geograficznymi skrajnych punktów Brazylii (73°58′58.19″ W i 34°47′35.33″ W) oraz których wylot nastąpił w pierwszej połowie roku.

   Rozszerzenie schematu: rozszerzenie 1 i 3

   Operacja SQL: SELECT (operacja 1 z zestawu B)

   ```
   SELECT COUNT(*)
   FROM flight f
     JOIN airport a ON a.id = f.airport_id
   ```

```
WHERE EXTRACT(MONTH FROM TRUNC(f.departure)) IN (1, 6)
      AND a.longitude BETWEEN -73.98283055555555 AND -34.79314722222222;
```

Implementacja XML: PL/SQL z wyrażeniem `extractValue`

```
SELECT COUNT(*)
FROM Flight_xml f
  JOIN Airport_xml a ON a.id = extractValue(value(f), '//AirportId')
WHERE EXTRACT(MONTH FROM TRUNC(extractValue(value(f), '//Departure'))) IN (1, 6)
      AND extractValue(a.location, '//Coordinates/Longitude') BETWEEN -73.98283055555555 AND -34.79314722222222;
```

3. **Wybrane typy samolotów**

   Opis: Wszystkie unikalne typy samolotów, którymi lecieli pracownicy pracujący dla linii lotniczej (zatrudniającej ich), których pierwsza litera imienia należy do pierwszej połowy alfabetu (A-M).

   Rozszerzenie schematu: rozszerzenie 2

   Operacja SQL: SELECT (operacja 2 z zestawu B)

```
SELECT DISTINCT model
FROM plane p
  JOIN airline a ON a.id = p.airline_id
  JOIN employee e ON a.id = e.airline_id
WHERE SUBSTR(e.name, 1, 1) BETWEEN 'A' AND 'M';
```

   Implementacja XML: PL/SQL z wyrażeniem `extractValue`

```
SELECT DISTINCT extractValue(value(p), '//Model')
FROM Plane_xml p
  JOIN airline a ON a.id = extractValue(value(p), '//AirlineId')
  JOIN employee e ON a.id = e.airline_id
WHERE SUBSTR(e.name, 1, 1) BETWEEN 'A' AND 'M';
```

4. **Obliczenie przelatanych godzin**

   Opis: Obliczenie przelatanych godzin dla wszystkich pracowników na podstawie długości lotów, w których brali udział.

   Rozszerzenie schematu: rozszerzenie 1

   Operacja SQL: UPDATE (operacja 4 z zestawu A)

```
UPDATE employee eu
SET eu.flown_hours = (SELECT SUM(24 * (f.arrival - f.departure))
                      FROM flight f
                      JOIN flight_employee fe ON f.id = fe.flight_id
                      JOIN employee e ON e.id = fe.employee_id);
```

   Implementacja XML: PL/SQL z wyrażeniem `extractValue`

```
UPDATE employee eu
SET eu.flown_hours = (SELECT SUM(24 * (sysdate + extractValue(value(f), '//Arrival') - extractValue(value(f), '//Departure'
                       FROM Flight_xml f
                       JOIN flight_employee fe ON extractValue(value(f), '//@id') = fe.flight_id
                       JOIN employee e ON e.id = fe.employee_id);
```

5. **Usunięcie nieprawidłowych lotów**

   Opis: Usunięcie wszystkich lotów, dla których liczba pasażerów przekracza liczbę siedzeń w samolocie.

   Rozszerzenie schematu: rozszerzenie 1 i 2

   Operacja SQL: DELETE (operacja 5 z zestawu B)

```
DELETE (SELECT *
        FROM flight f
          INNER JOIN plane p ON p.id = f.plane_id
        WHERE f.passengers > p.seat_count);
```

Implementacja XML: XQuery (`XMLTable`) oraz PL/SQL z wyrażeniem `extractValue`

```
DELETE (SELECT *
        FROM Flight_xml, XMLTable('for $f in /Flight return $f' passing object_value) xf
          INNER JOIN Plane_xml p ON extractValue(value(p), '//@id') = extractValue(value(xf), '//PlaneId')
        WHERE extractValue(value(xf), '//Passengers') > extractValue(value(p
), '//SeatCount');
```

Pomiary

Zestaw A

Aa Numer próby	# Próbka obciążenia [s]	# Dane XML [s]
1	5.31	3.6
2	5.7	4.04
3	5.65	3.88
4	5.51	3.72
5	5.61	3.53

$$\sigma_{probek} = 0.13792751719653, \quad \sigma^2_{probek} = 0.019024$$

$$\sigma_{xml} = 0.18586016248782, \quad \sigma^2_{xml} = 0.034544$$

Zestaw B

Aa Numer próby	# Próbka obciążenia [s]	# Dane XML [s]
1	3.5	0.18
2	3.03	0.25
3	3.2	0.31
4	3.16	0.16
5	2.99	0.18

$$\sigma_{probek} = 0.17984437717093, \quad \sigma^2_{probek} = 0.032344$$

$$\sigma_{xml} = 0.056071383075505, \quad \sigma^2_{xml} = 0.003144$$

Zestaw C

Aa Numer próby	# Próbka obciążenia [s]	# Dane XML [s]
1	27.3	17.63
2	29.38	16.31
3	26.81	16.8
4	26.52	16.4
5	26.52	16.71

$$\sigma_{probek} = 1.0754645507872, \quad \sigma^2_{probek} = 1.156624$$

$$\sigma_{xml} = 0.46746122833878, \quad \sigma^2_{xml} = 0.21852$$

Średnie pomiary czasu

Zestaw	Próbka obciążenia [s]	próbka	Dane XML [s]	xml	AVG(xml) / AVG(próbka) * 100%
A	5.556 ±0.121	5.556	3.754 ±0.163	3.754	67.566594672426
B	3.176 ±0.158	3.176	0.216 ±0.0491	0.216	6.801007556675
C	27.306 ±0.943	27.306	16.77 ±0.41	16.77	61.415073610196

Ostatnia kolumna determinuje jaki procent pierwotnego czasu próbek obciążenia stanowią czasy z wykorzystaniem danych XML.

Wnioski

- Stosowanie rozszerzeń w postaci XML przyspiesza czas zwykłych operacji SQL.
  W przypadku zestawu C, który nie miał żadnych zmienionych operacji, zwykłe zapytania SQL wykonały się w prawie 40% krótszym czasie.
- Stosowanie zapytań PL/SQL z wyrażeniami XPath oraz XQuery prawdopodobnie też wpływa na szybsze czasy wykonania.
  Czas wykonania zestawu B, w którym zostały zmienione aż 3 zapytania w ten sposób, skrócił się ponad 14 razy.