Kamila Jałocha GR. I 408053
Geoinformatyka II rok

WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

Poniższe sprawozdanie ma celu niejako odtworzenie pracy z artykułu Łukasza Jajeśnicy oraz Adama Piórkowskiego o tym samym tytule. Sprawdza jak obecne wersje programów radzą sobie z użytymi wtedy zapytaniami. W obecnej pracy zostało zmienione jedno środowisko, drugie pozostało takie samo.

W ramach sprawdzenia wydajności złączeń i zagnieżdżeń, zostały przeprowadzone testy dla dwóch systemów zarządzania bazami danych: PostgreSQL i MS SQL Server. Cały eksperyment opiera się o tabelę geochronologiczną, z którą są łączone inne tabele. Zapytania były stworzone dla tabel w postaci znormalizowanej oraz zdenormalizowanej.

W ramach przypomnienia użytych wyrażeń:

NORMALIZACJA

Normalizacja jest procesem, który ma organizować dane w tabeli, zmniejszając przy tym ilość powtarzających się danych w bazie.

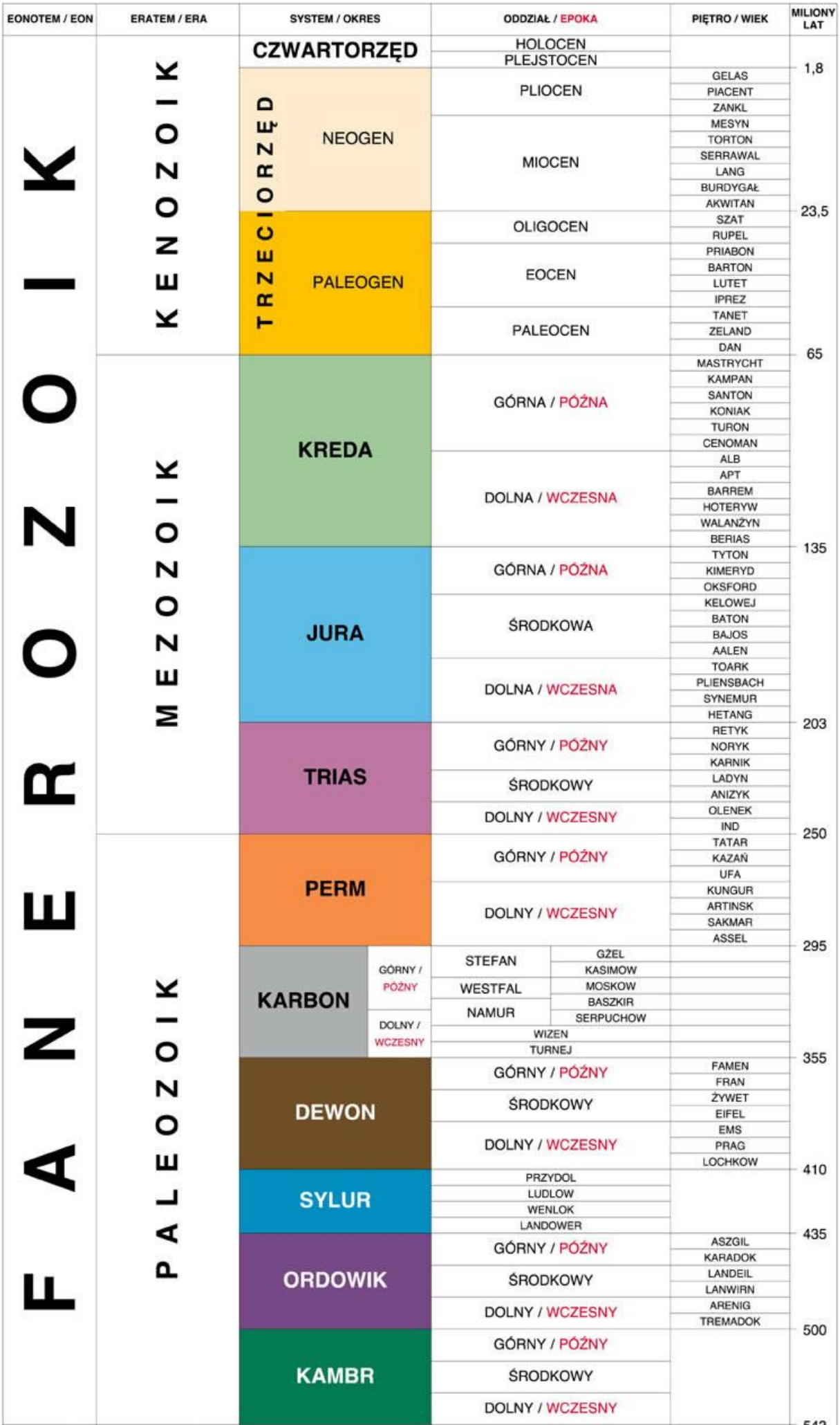
W dużym skrócie i uproszczeniu, normalizacja to podział większej tabeli na mniejsze tabele. Nie odnosi się to jednak do rekordów, a kolumn, ale też nie do ich ilości, a do relacji jakie występują między kolumnami. Normalizując dane, zmieniamy schematy tabel.

PODZAPYTANIA I ZAGNIEŻDŻENIA

Podzapytania służą do zagnieżdżania zapytań. Można je zastosować, gdy jedno zapytanie ma bazować na wyniku drugiego zapytania. Podzapytanie polega na umieszczeniu instrukcji SELECT wewnątrz innej instrukcji SELECT. Podzapytanie może być nieskorelowane – wewnętrzne zapytanie nie jest powiązane z zewnętrznym i może być wykonane samodzielnie, lub skorelowane – gdzie zapytanie wewnętrzne jest powiązane z tym nadrzędnym.

TABELA STRATYGRAFICZNA

Tabela stratygraficzna to schematyczne przedstawienie historii Ziemi na podstawie następujących po sobie procesów geologicznych. W testach zostały wykorzystane tylko kolumny: Eon, Era, Okres, Epoka i Piętro.


EONOTEM / EON
ERATEM / ERA
SYSTEM / OKRES
ODDZIAŁ / EPOKA
PIĘTRO / WIEK
MILIONY LAT
FANEROZOIK
KENOZOIK
MEZOZOIK
PALEOZOIK
CZWARTORZĘD
HOLOCEN
PLEJSTOCEN
TRZECIORZĘD
NEOGEN
PLIOCEN
GELAS
PIACENT
ZANKL
MIOCEN
MESYN
TORTON
SERRAWAL
LANG
BURDYGAŁ
AKWITAN
PALEOGEN
OLIGOCEN
SZAT
RUPEL
EOCEN
PRIABON
BARTON
LUTET
IPREZ
PALEOCEN
TANET
ZELAND
DAN
KREDA
GÓRNA / PÓŹNA
MASTRYCHT
KAMPAN
SANTON
KONIAK
TURON
CENOMAN
DOLNA / WCZESNA
ALB
APT
BARREM
HOTERYW
WALANŻYN
BERIAS
JURA
GÓRNA / PÓŹNA
TYTON
KIMERYD
OKSFORD
ŚRODKOWA
KELOWEJ
BATON
BAJOS
AALEN
DOLNA / WCZESNA
TOARK
PLIENSBACH
SYNEMUR
HETANG
TRIAS
GÓRNY / PÓŹNY
RETYK
NORYK
KARNIK
ŚRODKOWY
LADYN
ANIZYK
DOLNY / WCZESNY
OLENEK
IND
PERM
GÓRNY / PÓŹNY
TATAR
KAZAŃ
UFA
DOLNY / WCZESNY
KUNGUR
ARTINSK
SAKMAR
ASSEL
KARBON
GÓRNY / PÓŹNY
STEFAN
GŻEL
KASIMOW
WESTFAL
MOSKOW
NAMUR
BASZKIR
SERPUCHOW
DOLNY / WCZESNY
WIZEN
TURNEJ
DEWON
GÓRNY / PÓŹNY
FAMEN
FRAN
ŚRODKOWY
ŻYWET
EIFEL
DOLNY / WCZESNY
EMS
PRAG
LOCHKOW
SYLUR
PRZYDOL
LUDLOW
WENLOK
LANDOWER
ORDOWIK
GÓRNY / PÓŹNY
ASZGIL
KARADOK
ŚRODKOWY
LANDEIL
LANWIRN
DOLNY / WCZESNY
ARENIG
TREMADOK
KAMBR
GÓRNY / PÓŹNY
ŚRODKOWY
DOLNY / WCZESNY
1,8
23,5
65
135
203
250
295
355
410
435
500

STWORZENIE TABEL Z DANYMI STRATYGRAFICZNYMI

Tabele w wersji znormalizowanej zostały utworzone w obydwu środowiskach standardowymi poleceniami *CREATE TABLE* oraz *INSERT INTO.*
Sam schemat tabel jest identyczny jak w pracy, na której opiera się to sprawozdanie, tj.:

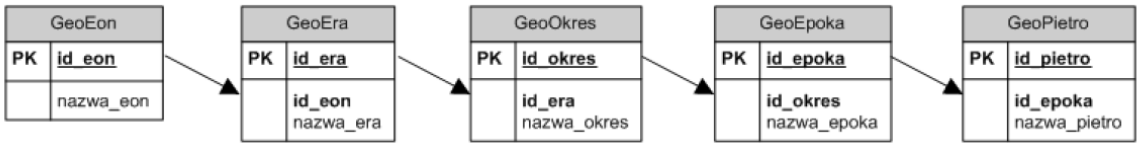


Tabela w wersji zdenormalizowanej również została utworzona w ten sam sposób:

```
CREATE TABLE GeoTabela AS(SELECT * FROM GeoPietro NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon);
```

Taki zapis wystąpił jednak tylko w systemie PostgreSQL. W SQL Server ze względu na różnice składniowe, polecenie *NATURAL JOIN* zostało zastąpione poleceniem *INNER JOIN….ON.*
Samo zapytanie wykorzystało również *SELECT,* którym wybrano odpowiednie kolumny.

STWORZENIE TABEL UZUPEŁNIAJĄCYCH

Testy opierają na sprawdzeniu złączeń i zagnieżdżeń na tabelach zawierających dużą ilość danych.

W tym celu zostały stworzone dwie dodatkowe tabele: Dziesiec i Milion. Tabela Dziesiec wypełniona jest cyframi od 0 do 9.

Tabela Milion, wypełniona danymi od 0 do 999 999, została na utworzona na jej podstawie:

```
CREATE TABLE Milion (liczba INT, cyfra INT, bit INT);

INSERT INTO Milion SELECT a1.cyfra + 10*a2.cyfra + 100*a3.cyfra +
1000*a4.cyfra + 10000*a5.cyfra + 100000*a6.cyfra AS liczba, a1.cyfra AS
cyfra, a1.bit AS bit FROM Dziesiec AS a1, Dziesiec AS a2, Dziesiec AS a3,
Dziesiec AS a4, Dziesiec AS a5, Dziesiec AS a6;
```

W wykonywanych zapytaniach, dane z tabeli geochronologicznej są łączone z danymi z tabeli Milion.

KONFIGURACJA SPRZĘTOWA I PROGRAMOWA

W przeciwieństwie do wzorcowego doświadczenia, omówione testy zostały wykonane tylko na jednym komputerze o następujących parametrach:

- CPU: AMD Ryzen 5 25000U with Radeon Vega Mobile Gfx 2,0 GHz
- RAM: Pamięć DDR4 8,0 GB
- HDD: HFS256G39TNF-N3A0A, dysk SSD
- S.O.: Microsoft Windows 10 Home

Wybrane systemy zarządzania bazami danych:

- Microsoft SQL Server Management Studio 18
  Microsoft SQL Server 2019, wersja 2019.15.0.2080.9
- PostgreSQL wersja 14.2
  pgAdmin4 wersja 6.4

W obu środowiskach, każde zapytanie zostało wykonane pięciokrotnie.
Obydwa systemy były zainstalowane w tym samym czasie.

WYKONANE TESTY

W tym kroku zostały odtworzone zapytania, które były wykonywane w pracy wzorcowej. Zapytania te miały na celu sprawdzić wydajność złączeń i zagnieżdżeń z utworzoną tabelą geochronologiczną w wersji znormalizowanej i zdenormalizowanej. Sposób postępowania obejmował dwa etapy:

1. Wykonanie zapytań bez nałożonych indeksów (obecne były jedynie indeksy tworzące się automatycznie w momencie dodania klucza głównego do tabeli)
2. Wykonanie zapytań z nałożonymi indeksami na kolumny biorące udział w złączeniu.

Cztery zaproponowane zapytania:

- 1 Zapytanie (1Z) – złączenie tabeli Milion z tabelą geochronologiczną w postaci zdenormalizowanej, z dołączonym warunkiem modulo.
- 2 Zapytanie (2Z) – złączenie tabeli Milion z tabelą geochronologiczną w postaci znormalizowanej (złączenie pięciu tabel).
- 3 Zapytanie (3Z) – złączenie tabeli Milion z tabelą geochronologiczną w postaci zdenormalizowanej poprzez zagnieżdżenie skorelowane.
- 4 Zapytanie (4Z) – złączenie tabeli Milion z tabelą geochronologiczną w postaci znormalizowanej poprzez zagnieżdżenie skorelowane, a w zapytaniu wewnętrznym następuje złączenie poszczególnych pięciu tabel.

Kod SQL w PostgreSQL (praktycznie identyczny):

1Z:
```
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON (mod(Milion.liczba,68) = (Geotabela.id_pietro));
```

2Z:
```
SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro ON (mod(Milion.liczba,68) = GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

3Z:
```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68) = (SELECT id_pietro FROM GeoTabela WHERE mod(Milion.liczba,68)=(id_pietro));
```

4Z:
```
SELECT COUNT(*) FROM Milion WHERE mod(Milion.liczba,68) IN (SELECT GeoPietro.id_pietro FROM GeoPietro JOIN GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka JOIN GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres JOIN GeoEra ON GeoOkres.id_era = GeoEra.id_era JOIN GeoEon ON GeoEra.id_eon = GeoEon.id_eon);
```

Kod SQL w MS SQL Server(niewielkie różnice w stosunku do PostgreSQL):

1Z:
```
SELECT COUNT (*) FROM Milion INNER JOIN GeoTabela
ON Milion.liczba % 68 = GeoTabela.id_pietro;
```

2Z:
```
SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro
ON Milion.liczba % 68 = GeoPietro.id_pietro
INNER JOIN GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
INNER JOIN GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
INNER JOIN GeoEra ON GeoOkres.id_era = GeoEra.id_era
INNER JOIN GeoEon ON GeoEra.id_eon = GeoEon.id_eon
```

3Z:
```
SELECT COUNT(*) FROM Milion
WHERE Milion.liczba % 68 = (SELECT id_pietro FROM GeoTabela
WHERE Milion.liczba % 68 = id_pietro);
```

4Z:
```
SELECT COUNT(*) FROM Milion
WHERE Milion.liczba % 68 IN (SELECT GeoPietro.id_pietro FROM GeoPietro
JOIN GeoEpoka ON GeoPietro.id_epoka = GeoEpoka.id_epoka
JOIN GeoOkres ON GeoEpoka.id_okres = GeoOkres.id_okres
JOIN GeoEra ON GeoOkres.id_era = GeoEra.id_era
JOIN GeoEon ON GeoEra.id_eon = GeoEon.id_eon);
```

WYNIKI

Dla każdego zapytania został obliczony średni czas wykonania oraz wybrany czas minimalny, obydwa przedstawione w ***milisekundach*.**

Uzyskane wyniki zaprezentowano w tabelach poniżej:

Tabela1

MSSQL SERVER								
	1Z		**2Z**		**3Z**		**4Z**	
BEZ INDEKSÓW	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG
	101	143	54	76	3067	3149	47	74
Z INDEKSAMI	42	48	43	49	1112	1135	43	63

Tabela2

POSTGRESQL								
	1Z		**2Z**		**3Z**		**4Z**	
BEZ INDEKSÓW	MIN	AVG	MIN	AVG	MIN	AVG	MIN	AVG
	201	263	453	491	11570	12238	15764	16332
Z INDEKSAMI	211	222	417	442	12104	12394	16364	17465

WNIOSKI

Pierwszym co można powiedzieć zestawiając ze sobą obecne wyniki z wynikami z artykułu oryginalnego, to to, że systemy są o wiele wydajniejsze. Szczególnie można zwrócić uwagę na PostgreSQL, który był wykorzystany w obydwu doświadczeniach. Obecnie wykonanie tych czterech przykładowych zapytań zajmuje kilkaset milisekund bądź kilkanaście sekund w zależności od zapytania, podczas gdy wcześniej wykonanie chociażby zapytania czwartego zajmowało nawet do ponad minuty.

Porównując ze sobą SQL Server i PostgreSQL, widać, że system MSSQL Server, biorąc pod uwagę ogólne spojrzenie, jest zdecydowanie szybszy od PostgreSQL. Widać także, że nałożenie indeksów na poszczególne kolumny w tym środowisku sprawiło, że zapytania wykonywały się o wiele szybciej, czego nie można powiedzieć o PostgreSQL, gdzie momentami można było mieć wrażenie, że indeksacja wręcz wydłuża cały proces. W szczególności było to zauważalne w trzecim i czwartym zapytaniu.

W systemie SQL Server zapytania wykonywały się w podobnym czasie, zarówno dla postaci znormalizowanej jak i zdenormalizowanej. Jedynie wyraźna różnica następowała w przypadku zapytania trzeciego, gdzie czas wykonania bez indeksu to ponad 3 sekundy, a z indeksem ponad sekundę, podczas gdy reszta zapytań zajmowała do kilkudziesięciu milisekund.

Nadal jednak można powiedzieć, że dla większości przypadków postać zdenormalizowana jest wydajniejsza.
Potwierdza się również wniosek, że zagnieżdżenia są wolniejsze niż zwykłe złączenia tabel, a także fakt, że postać znormalizowana działa szybciej dla zagnieżdżenia skorelowanego.

MSSQL Server zdecydowanie przeważa pod względem wydajności w stosunku do złączeń jak i do zagnieżdżeń. W większości, to właśnie w tym systemie zapytania wykonywały się szybciej.

Ciężko jest jednak jednoznacznie porównać oba systemy. Patrząc jedynie na MSSQL Server można by dojść do wniosku, że postać znormalizowana usprawnia wydajność, ponieważ to właśnie te zapytania z reguły wykonywały się szybciej. Natomiast w systemie PostgreSQL jest dokładnie na odwrót – to zapytania z tabelami w postaci zdenormalizowanej wykonywały się szybciej.

Podsumowując, uwzględniając powyższe wyniki, normalizacja może prowadzić zarówno do spadku jak i do niewielkiego wzrostu wydajności. Uzależnione jest to środowiskiem, w jakim wykonywane są zapytania, ale także od rodzaju samego zapytania. Istotne jest także, czy na tabele zostały nałożone indeksy, ponieważ ich obecność w wielu sytuacjach może poprawić sprawność. Choć z reguły normalizacja wiąże się jednak z wysokimi kosztami wydajnościowymi, pozwala zachować porządek w bazie danych, rozwiązuje problem wystąpienia wszelkich anomalii dodawania, usuwania i modyfikacji rekordów oraz zmniejsza ogólną ilość danych przechowywanych w bazie.

Bibliografia:

1. Jaję śnica Ł., Piórkowski A.: Wydajność złączeń i zagnieżdżeń dla schematów znormalizowanych i zdenormalizowanych [pdf]
2. Dr inż. Lupa Michał: 2022 Bazy danych I – materiały z wykładu
3. Wydawnictwo Naukowe PWN S.A. 2004 http://stareaneksy.pwn.pl/historia_ziemi/przyklady/?pokaz=tabela