Hurtownie Danych
ETL – wybrane zagadnienia

Krzysztof Jankiewicz

Politechnika Poznańska

Instytut Informatyki

Plan

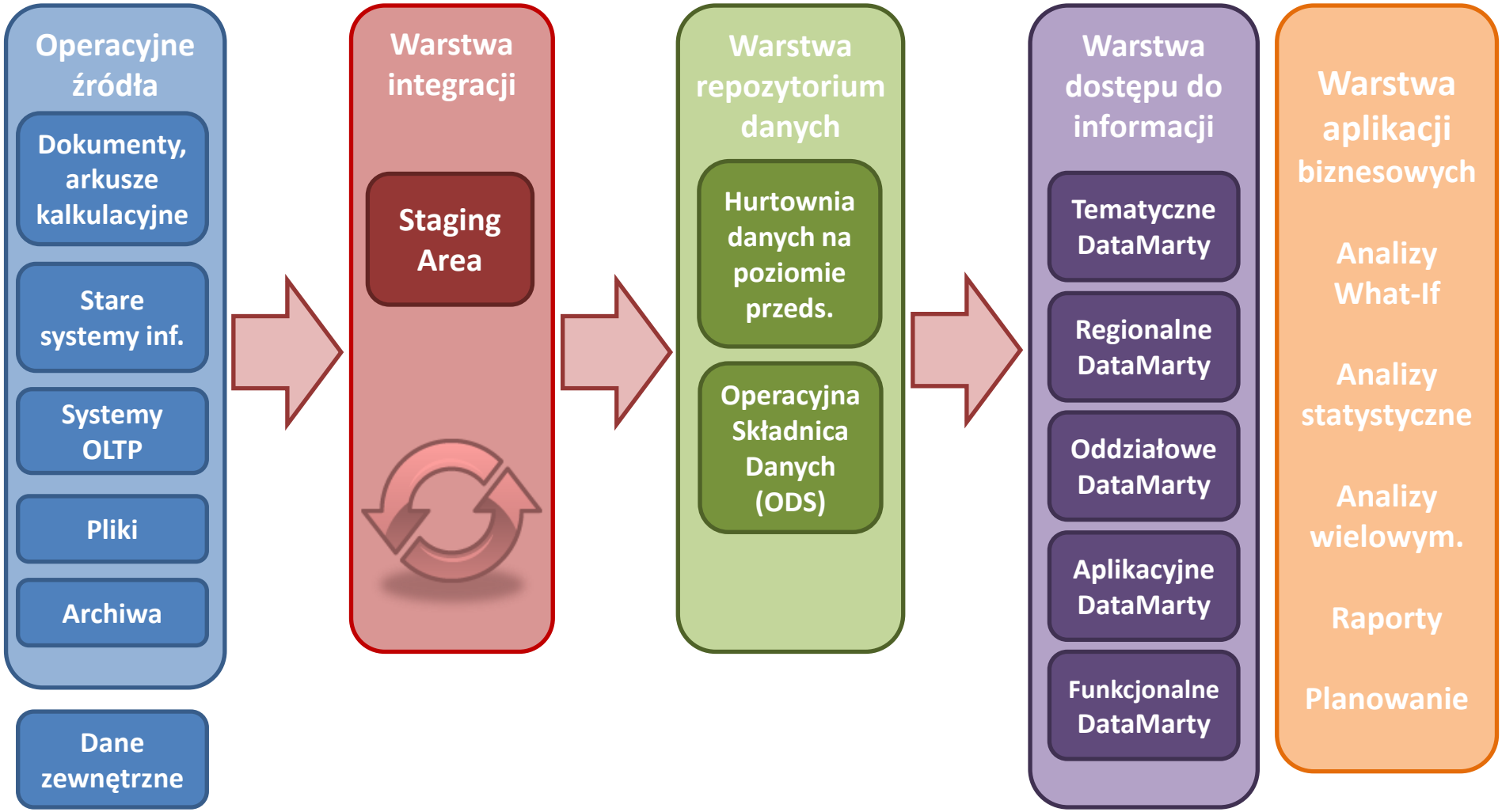
- Czym jest ETL?
- Architektura Hurtowni Danych – działania ETL
- Działania ETL
  - Ekstrakcja
  - Czyszczenie i dostosowywanie
  - Dostarczanie
  - Zarządzanie
- Architektura Hurtowni Danych – repozytoria ETL
  - Data Warehouse Staging Area
  - Operational Data Store
- Narzędzia czy własne rozwiązania?

Czym jest ETL?

- Kluczowy składnik infrastruktury Hurtowni Danych
- Czymś w rodzaju "kuchni" w restauracji?
- Skrót
  - E – ekstrakcja
  - T – transformacja
  - L – ładowanie (dostarczanie)
- A rzeczywistość? – Ralph Kimball* wyróżnia 34 działania (podsystemy) wchodzące w skład procesów ETL

* założyciel Kimball Group i autor wielu książek poświęconych tematyce Hurtowni Danych i procesów ETL

Architektura Hurtowni Danych w kontekście procesów ETL



Dlaczego aż 34 działania?

Everyone understands the three letters:

- *You get the data out of its original source location (E),*
- *you do something to it (T),*
- *and then you load it (L) into a final set of tables for the business users to query.*

*When asked about the best way to **design and build the ETL system**, many designers say, "Well, that **depends**." It depends on the **source**; it depends on **limitations of the data**; it depends on the **scripting languages** and **ETL tools** available; it depends on the **staff's skills**; and it depends on the **BI tools**.*

*But the "it depends" response is dangerous because it becomes an excuse to take an **unstructured approach** to developing an ETL system, which in the worse-case scenario results in an **undifferentiated spaghetti-mess** of tables, modules, processes, scripts, triggers, alerts, and job schedules. This "creative" design approach should not be tolerated.*

Cytat: R. Kimball - http://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/etl-architecture-34-subsystems/

Komponenty ETL

- Działania ETL zazwyczaj można pogrupować w następujące grupy uwzględniające zagadnienie, których dotyczą:
  - Ekstrakcja
  - Oczyszczenie i dostosowanie
  - Dostarczanie
  - Zarządzanie

Zakres działań 1
Ekstrakcja danych

- Działania:
  - Profilowanie danych (1) – analiza źródeł danych dostarczająca mniej lub bardziej złożonych statystyk (np. liczba krotek, wolumen danych, liczba wartości pustych, typy i formaty wykorzystywanych danych)
  - CDC (*Change Data Capture*) (2) – detekcja zmian w źródłach danych, które miały miejsce od ostatniej ekstrakcji.
  - Ekstrakcja (3) – pobranie danych z ich źródeł. Różnorodny format, funkcjonalność, dynamika i charakter źródeł determinuje złożoność tego procesu

Przykładowe działanie *Change Data Capture*

- To proces wykrywania zmian w źródłach danych
- Podział CDC:
  - inwazyjne
    - oparte na danych źródła danych (ang. *Source-Based CDC*)
    - oparte na wyzwalaczach źródła danych (ang. *Trigger-Based CDC*)
    - oparte na porównaniu poprzedniego i bieżącego obrazu źródła danych (ang. *Snapshot-Based CDC*)
  - nieinwazyjne
    - oparte na dziennikach (np. plikach dzienników powtórzeń) (ang. *Log-Based CDC*)

Source-Based CDC

- Źródło danych posiada znaczniki czasowe określające danę ostatniej modyfikacji/wstawienia
- Wady:
  - Nie każde źródło znaczniki posiada
  - Poprawne wykorzystywanie tych znaczników zależy od zewnętrznych czynników (poza ETL) np. aplikacji, wyzwalaczy
  - Brak informacji o usunięciu danych
  - Nie można wykryć wielokrotnych modyfikacji
  - Możliwość desynchronizacji zegarów (ETL, system źródłowy)
- Zalety
  - Prostota

Trigger-Based CDC

- Źródło za pomocą wyzwalaczy rejestruje fakt zmiany danych
- Rejestracja może odbywać się w różnych repozytoriach
  - we wnętrzu systemu źródłowego w przeznaczonych do tego tabelach
  - na zewnątrz systemu źródłowego np.:
    - poprzez rejestrowanie zmienionych danych w zewnętrznym repozytorium (np. Staging Area)
    - poprzez uruchamianie procedur ETL z parametrami zawierającymi informacje o zmienionych danych
- Wady:
  - Nie każde źródło danych posiada funkcjonalność umożliwiającą implementację tego rozwiązania
  - Poprawne funkcjonowanie zależy od zewnętrznych czynników (poza ETL)
- Zalety
  - możliwość rejestracji wszelkich zmian (również usunięcia)
  - możliwość rejestracji wielokrotnych modyfikacji
  - możliwość implementacji dowolnie złożonych mechanizmów CDC

Snapshot-Based CDC

- Porównanie dwóch wersji danych źródłowych – poprzedniej (przechowywanej zazwyczaj w OSA) oraz bieżącej
- Idea porównania polega na wyznaczeniu zmian w źródłowych danych
  - wstawionych – (`SELECT * FROM DANE_BIEZACE WHERE ID NOT IN (SELECT ID FROM DANE_POPRZEDNIE)`)
  - usuniętych – (`SELECT * FROM DANE_POPRZEDNIE WHERE ID NOT IN (SELECT ID FROM DANE_BIEZACE)`)
  - zmienionych – (`SELECT * FROM DANE_BIEZACE B JOIN DANE_POPRZEDNIE P ON (B.ID=P.ID) WHERE B.C1<>P.C1 OR B.C2<>P.C2 OR …`)

Jakie wady a jakie zalety ma to rozwiązanie?

Log-Based CDC

- Analiza dzienników generowanych podczas standardowej pracy bazy danych
- Najmniej inwazyjny sposób wykrywania zmian
- Często używany w rozwiązaniach komercyjnych, ale także niekomercyjnych
  - Oracle GoldenGate
  - Attunity Stream
  - Wisdomforce
  - mysqlbinlog

Cechy różnych typów CDC

Cechy	Source-Based	Trigger-Based	Snapshot-Based	Log-Based
Rozróżnienie pomiędzy wstawieniem a modyfikacją	Nie	Tak	Tak	Tak
Detekcja wielu (sekwencji) modyfikacji	Nie	Tak	Nie	Tak
Detekcja usunięcia	Nie	Tak	Tak	Tak
Bezinwazyjne	Nie	Nie	Nie	Tak
Wsparcie dla systemów czasu rzeczywistego	Nie	Tak	Nie	Tak
Wymagane zaangażowanie administratora bazy danych	Nie	Tak	Nie	Tak
Niezależne od bazy danych	Tak	Nie	Tak	Nie

Zakres działań 2
Oczyszczenie i dostosowanie (1/2)

- Z reguły najbardziej złożony komponent (zagadnienie) w procesach ETL
  - praktycznie każda instytucja posiada problemy z jakością danych
  - bardzo rzadko istotne dane źródłowe przechowywane są w pojedynczym źródle …
- Działania
  - Oczyszczanie danych (4) – naprawa (poprawa jakości) danych wchodzących do procesów ETL, monitorowanie i raportowanie błędów w danych
    Detekcja danych wymagających naprawy wynika z:
    - profilowania danych
    - stosowanych reguł biznesowych
  - Obsługa błędów (5) – wykrywanie i rejestracja błędów zachodzących podczas procesów ETL. Pozwala to na monitorowanie i analizę błędów i ich przyczyn

Oczyszczenie i dostosowanie (2/2)

- Utrzymanie wymiaru kontrolnego (*audit dimension*) (6) – wymiar kontrolny to wewnętrzna składowa hurtowni danych, tabela powiązana z tabelą faktów zawierająca metadane dotyczące zmian w tabeli faktów:
  - czas rozpoczęcia/zakończenia ładowania danych do tabeli faktów
  - liczbę załadowanych/błędnych rekordów
  - informacje o błędnych rekordach
  - itp.
- Wykrywanie duplikatów (7) – jeden z bardziej złożonych problemów
  - duplikaty dokładne – problem trywialny
  - duplikaty przybliżone
- Potwierdzanie zgodności danych (8)
  - weryfikacja czy poszczególne wymiarów pochodzące z wielu źródeł są:
    - strukturalne identyczne,
    - pozbawione: duplikatów, błędnych danych
    - ustandaryzowane pod względem zawartości
  - weryfikacja czy dane faktów odnoszą się do określonych danych z tabel wymiarów

Przykładowe działanie Wykrywanie duplikatów

- Wykrywanie duplikatów dokładnych z reguły jest zadaniem trywialnym
- Wykrywanie duplikatów przybliżonych nie jest niestety już takie proste
  - czy Jankiewicz i Jaszkiewicz to ta sama osoba?
  - czy 79 Stapleton Road i Stapleton Rd, 79 to ten sam adres?
- W wielu przypadkach pomocne bywają techniki Fuzzy Matching.
  Przykłady często wykorzystywanych algorytmów:
  - Levenshtein i Damerau-Levenshtein
  - Needleman Wunsch
  - Jaro i Jaro Winkler
  - Pair letters similarity
  - Algorytmy fonetyczne

Levenshtein i Damerau-Levenshtein

- Wyznaczają odległość miedzy ciągami znaków wyrażoną krokami edycji wymaganymi do tego aby jeden ciąg przekształcić w drugi
- Algorytm Levenshtein uwzględnia następujące operacje
  - dodanie znaku – `Insert (i)`
  - usunięcie znaku – `Delete (d)`
  - zmiana znaku – `Substitute (s)`
- Algotrym Damerau-Levenshtein uzupełnia powyższą listę o
  - zamianę znaków – `Transpose (t)`

- Wynikiem porównania ciągów znaków dla obu algorytmów jest minimalna liczba wymaganych zmian
- Aby uniezależnić się od długości ciągów (odległość 2 ma inne znaczenie dla ciągu 2-literowego i 20-literowego) wyznacza się odległość procentowo dzieląc ją przez długość dłuższego ciągu

Levenshtein i Damerau-Levenshtein algorytm

```
int Damerau-Levenshtein-Distance(char str1[1..lenStr1], char str2[1..lenStr2])

  declare int d[0..lenStr1, 0..lenStr2]
  declare int i, j, cost

  for i from 0 to lenStr1  d[i, 0] := i
  for j from 1 to lenStr2  d[0, j] := j

  for i from 1 to lenStr1
    for j from 1 to lenStr2
        if str1[i] = str2[j] then cost := 0  else cost := 1
        d[i, j] := minimum(  d[i-1, j  ] + 1,     // usunięcie (d)
                             d[i  , j-1] + 1,     // wstawienie (i)
                             d[i-1, j-1] + cost   // zmiana (s)
                          )
        // poniższy fragment nie występuje w algorytmie Levenshteina
        if(i > 1 and j > 1 and str1[i] = str2[j-1] and str1[i-1] = str2[j]) then
            d[i, j] := minimum( d[i, j],
                                d[i-2, j-2] + cost   // zamiana (t)
                              )
return d[lenStr1, lenStr2]
```

Damerau, Fred J. (March 1964), "A technique for computer detection and correction of spelling errors", Communications of the ACM (ACM) 7 (3): 171–176

http://en.wikipedia.org/wiki/Damerau–Levenshtein_distance

Levenshtein i Damerau-Levenshtein przykłady

str1→		**n**	**a**	**j**
↓str2		1	2	3
j	1	**1 s**	2 s	2 e
a	2	2 s	**1 e**	2 i
n	3	2 e	2 d	**2 s**

str2→		**f**	**o**	**o**	**t**
↓str1		1	2	3	4
f	1	**0 e**	1 i	2 i	3 i
o	2	1 d	**0 e**	1 e	2 i
t	3	2 d	1 d	**1 t**	1 e
o	4	3 d	2 e	1 e	**1 t**

str2→		**a**	**m**	**p**	**h**	**i**	**b**	**i**	**a**
↓str1		1	2	3	4	5	6	7	8
a	1	**0 e**	1 i	2 i	3 i	4 i	5 i	6 i	7 e
m	2	1 d	**0 e**	**1 i**	2 i	3 i	4 i	5 i	6 i
f	3	2 d	1 d	1 s	**2 s**	3 s	4 s	5 s	6 s
i	4	3 d	2 d	2 s	2 s	**2 e**	3 i	4 e	5 i
b	5	4 d	3 d	3 s	3 s	3 s	**2 e**	3 i	4 i
i	6	5 d	4 d	4 s	4 s	3 e	3 d	**2 e**	3 i
a	7	6 e	5 d	5 s	5 s	4 d	4 s	3 d	**2 e**

str2→		**h**	**u**	**t**
↓str1		1	2	3
c	1	**1 d**	2 s	3 s
h	2	**1 e**	2 s	3 s
ł	3	**2 d**	2 s	3 s
ó	4	3 d	**3 s**	3 s
d	5	4 d	4 s	**4 s**

http://fuzzy-string.com/

Needleman Wunsch

- Określa podobieństwo pomiędzy ciągami znaków znajdując dodatkowo ich "najlepsze dopasowanie".
- Przez najlepsze dopasowanie rozumiemy dopasowanie wymagające najmniejszej liczby mutacji.
- Często wykorzystywane w bioinformatyce.
- Zaproponowane przez C. Wunscha i S. Needelmana w 1970.
- Algorytm wykorzystuje:
  - macierz kosztów zamiany mutacji (liter) – *scorring function* σ
  - koszt usunięcia lub dodania mutacji (litery) – $gap\_penalty$
  - rekurencyjną funkcję wykorzystywaną podczas wypełniania tabeli T

$$T(i,j) = \max \begin{cases} T(i-1, j-1) + \sigma(i,j) \\ T(i-1, j) + gap\_penalty \\ T(i, j-1) + gap\_penalty \end{cases}$$

Needleman Wunsch przykład

σ	n	a	j
n	1	-1	-1
a	-1	1	-1
j	-1	-1	1

gap_penalty = -2

str1→		n	a	j
↓str2	0	-2	-4	-6
j	-2	-1	-3	-3
a	-4	-3	0	-1
n	-6	-3	-2	-1

$$T(i,j) = \max \begin{cases} T(i-1, j-1) + \sigma(i,j) \\ T(i-1, j) + gap\_penalty \\ T(i, j-1) + gap\_penalty \end{cases}$$

- Wyznaczenie macierzy T umożliwia określenie najlepszego dopasowania pomiędzy sekwencjami
  - startujemy od dolnej prawej komórki
  - przechodzimy kolejno do "poprzedzających" komórek wybierając te o najlepszym wyniku

```
n a j
  |
j a n
```

Needleman Wunsch przykłady

$$T(i,j) = \max \begin{cases} T(i-1, j-1) + \sigma(i,j) \\ T(i-1, j) + gap\_penalty \\ T(i, j-1) + gap\_penalty \end{cases}$$

- Budowa macierzy kosztów umożliwia uwzględnianie różnorodnych aspektów związanych ze zmianą liter np.
  - odległości klawiszy na klawiaturze,
  - trudności podmiany nukleotydów

σ	F	O	T
F	1	-5	-1
O	-5	1	-4
T	-1	-4	1

gap_penalty = -1

```
F O O T _
| |   |
F O _ T O
```

```
G A T T G A C A
  | |     | | |
_ A T _ _ A C A
```

gap_penalty = -5

σ	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

str2→		F	O	O	T
↓str1	0	-1	-2	-3	-4
F	-1	1	0	-1	-2
O	-2	0	2	1	0
T	-3	-1	1	0	2
O	-4	-2	0	2	1

str2→		G	A	T	T	G	A	C	A
↓str1	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-5	-1	5	0	-5	-10	-15	-20	-25
T	-10	-6	-5	13	8	3	-2	-7	-12
A	-15	-11	4	8	9	7	13	8	3
C	-20	-16	-1	4	8	4	8	22	17
A	-25	-21	-6	-1	3	7	14	17	32

Jaro i Jaro Winkler algorytm (1/4)

- Wyznacza podobieństwo pomiędzy dwoma ciągami znaków.
- Wynik mieści się w granicach 0..1, gdzie 0 oznacza brak podobieństwa, a 1 identyczność.
- Wyznaczanie podobieństwa odbywa się w kilku krokach:
  - Krok 1 – wyznaczanie pasujących znaków
    - Na początku wyliczana jest *matchRange* – maksymalna odległość pomiędzy znakami pozwalająca na uznanie ich pasującymi $matchRange = \frac{\max(|s1|,|s2|)}{2} - 1$
    - Następnie wyznaczane są pasujące do siebie znaki oraz ich liczba *numCommon*. Pasujące znaki są takie same, a ich odległość nie jest większa niż *matchRange*

M A R T H A
M A R H T A
matchRange = max(6,6)/2 - 1 = 3 - 1 = 2
numCommons = 6

J O N E **S**
J O H **N S** O N
matchRange = max(5,7)/2 - 1 = 3 - 1 = 2
numCommons = 4

Jaro i Jaro Winkler algorytm (2/4)

- Krok 2 – wyznaczanie liczby transpozycji *numTransposed*
  - Po dopasowaniu liter podsekwencje dopasowanych znaków w obu porównywanych ciągach są ekstrahowane
  - Obie te podsekwencje posiadają tę samą długość.
  - Liczba znaków w jednej z podsekwencji, która nie pokrywa się (pod względem pozycji) ze znakami z drugiej podsekwencji jest liczbą pół-transpozycji (ang. *half transpositions*).
  - Całkowita liczba transpozycji podzielona przez dwa i zaokrąglona w dół jest liczbą transpozycji

M A R T H A
M A R H T A
numTransposed = 2/2 = 1

J O N E **S** => **J O N S**
J O H **N S** O N => **J O N S**
numTransposed = 0

Jaro
algorytm (3/4)

- Podobieństwo ciągów wg algorytmu **Jaro** wyrażone jest wzorem

$$d_j = \begin{cases} 0 & if\ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & otherwise \end{cases}$$

gdzie:
  - m – *numCommons*
  - t – *numTransposed*
- Innymi słowy, podobieństwo wg alg. Jaro jest średnią z trzech wartości:
  - procentowo wyrażonej części pasującego pierwszego ciągu
  - procentowo wyrażonej części pasującego drugiego ciągu
  - procentowo wyrażonej części pasujących znaków, które nie wymagają transpozycji.

M A R T H A
M A R H T A
numCommons = 6
numTransposed = 1
jaroProximity = 1/3(6/6+6/6+5/6) =* ***0,944***

J O N E ***S***
J O H ***N S*** O N
numCommons = 4
numTransposed = 0
jaroProximity = 1/3(4/5+4/7+4/4) =* ***0,79***

Jaro Winkler algorytm (4/4)

- Modyfikacja Winklera pozwala na **zwiększenie** wartość podobieństwa dla ciągów, których podobieństwo wg algorytmu Jaro przekracza zadany poziom *weightThreshold* (w źródłowych opracowaniach 0,7)
- Zwiększenie wartości podobieństwa opiera się na analizie podobieństwa pierwszych *numChars* znaków (w źr. opr. 4).
- Ostatecznie zatem podobieństwo ciągów wg algorytmu **Jaro Winkler** wyrażone jest wzorem:

$$d_w = \begin{cases} d_j & if\ d_j \leq weightThreshold \\ d_j + 0.1 * prefixMatch(s1, s2, numChars) * \left(1 - d_j\right) & otherwise \end{cases}$$

- gdzie:
  - d_j – podobieństwo wg alg. Jaro
  - *prefixMatch* – długość wspólnego prefiksu *s1* i *s2* ograniczona do *numChars*

M A R T H A
M A R H T A
jaroProximity = ***0,944***
jaroWinklerProx= ***0,944+0,1*3*(1-0,944)=0,961***

J O N E ***S***
J O H ***N S*** O N
jaroProximity = ***0,79***
jaroWinklerProx=***0,79+0,1*2*(1-0,79)=0,832***

Pair letters similarity

- Bardzo prosta metoda wyznaczania podobieństwa, która oparta jest na porównywaniu par znaków w obu ciągach
- Algorytm
  - każdy porównywany ciąg znaków dzielony jest na pary znaków
  - wyznaczana jest liczba par znaków istniejących w obu porównywanych ciągach
  - wyznaczona powyżej liczba dzielona jest przez liczbę wszystkich par

```
M A R T H A  => (MA,AR,RT,TH,HA)
M A R H T A  => (MA,AR,RH,HT,TA)
```

*commonPairs = 4 (2*2)*
allPairs = 10
pairLettersSimilarity = 0,4

```
J O N E S      => (JO,ON,NE,ES)
J O H N S O N => (JO,OH,HN,NS,SO,ON)
```

*commonPairs = 4 (2*2)*
allPairs = 10
pairLettersSimilarity = 0,4

Algorytmy fonetyczne

- Algorytmy fonetyczne dopasowują ciągi znaków na podstawie ich "brzmienia".
- Algorytmy te użyteczne są w przypadku korzystania z określonego języka (angielski) i nie przydatne są w przypadku innych języków
- Przykładowe algorytmy fonetyczne to:
  - Metaphone,
  - Double Metaphone,
  - SoundEx,
  - Refined SoundEx

Zakres działań 3
Dostarczanie danych (1/2)

- Dostarczanie danych do systemu źródłowego musi uwzględniać wiele różnych aspektów np.:
  - Różne sposoby aktualizacji tabel wymiarów uwzględniające stosowane techniki SCD
  - Generowanie sztucznych kluczy
  - Zapewnienie, że wszystkie dane wymiarów zostaną załadowane zanim będą ładowane tabele faktów
  - Ładowanie tabel faktów
    - z reguły posiadają znaczące rozmiary
    - uwzględniają lub nie modyfikację
  - Różne modele hurtowni danych MOLAP, ROLAP
- Działania:
  - Obsługa SCD (9)
  - Generowanie sztucznych kluczy (10)
  - Budowanie i kontrola hierarchii wymiarów (11)
  - Obsługa wymiarów specjalnych (12)

Dostarczanie danych (2/2)

- Ładowanie tabel faktów (13)
- Pobieranie kluczy wymiarów podczas ładowania faktów (14)
- Ładowanie tabel wymiarów o różnej wysokości hierarchii lub korzystających z powiązań N-M (15)
- Obsługa faktów, które dotyczą wartości nie znajdujących się w tabelach wymiarów (*ang. Late-Arriving Data*) (16)
- Dostarczanie wymiarów do Data Martów (17)
- Dostarczanie faktów do Data Martów (18)
- Obliczanie danych zagregowanych (19)
- Obsługa wielowymiarowych kostek danych (MOLAP) (20)
- Dostarczanie (integracja) danych hurtowni do zewnętrznych repozytoriów (21)

Przykładowe działanie SCD

- Slowly Changing Dimension

Przykładowe działanie
Obsługa wymiarów specjalnych

- W każdej hurtowni istnieje co najmniej jeden specjalny wymiar (wymiar czasu), którego utrzymanie różni się od utrzymania wymiarów konwencjonalnych
- Innymi przykładami wymiarów specjalnych mogą być:
  - wymiary "**śmieciowe**" (*junk*, *garbage*) – zawierające atrybuty, które są wymagane podczas analiz jednak nie pasują do innych wymiarów
  - **mini-wymiary** – pozwalają wyodrębnić atrybuty wymiarów o różnej częstotliwości zmian – patrz SCD 4
  - skurczone lub **zwinięte** wymiary – tworzone z wymiarów podstawowych wówczas, gdy zagregowane dane tworzone są na niższym poziomie ziarnistości (sklep->miejscowość)
  - **statyczne** wymiary – zwykle małe tabele, nie pochodzące ze źródeł danych, zawierające np. tłumaczenia, dane typu lookup (listy wartości, statusy, warianty, rozwinięte ciągi znaków)
  - wymiary **utrzymywane przez użytkowników** – sposoby grupowania, hierarchie, opisy, nie pochodzące ze źródeł danych ale wymagane podczas raportowania.

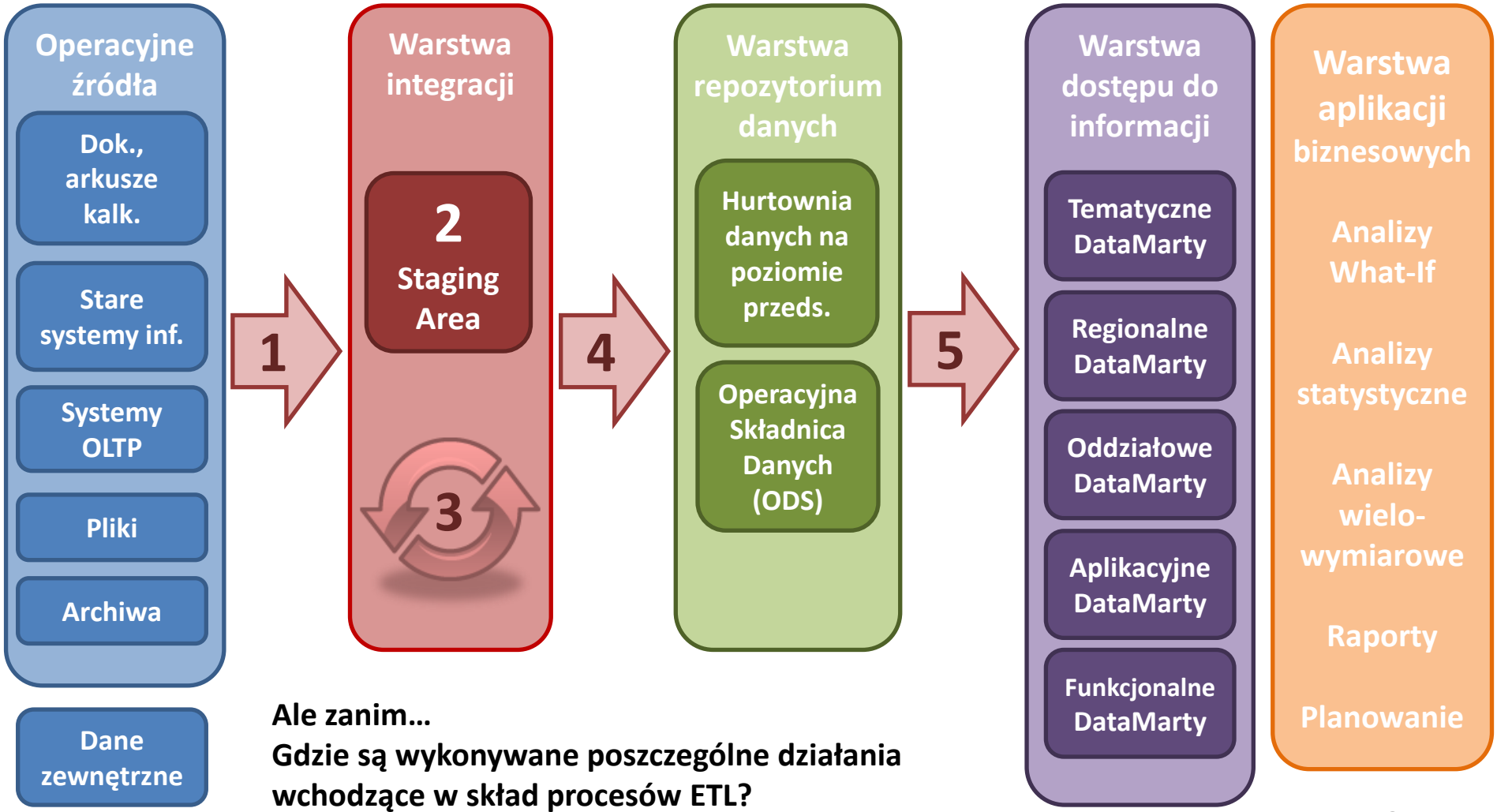
Zakres działań 4
Zarządzanie (1/2)

- Działania pozwalające zarządzać infrastrukturą ETL
- Działania
  - Harmonogramowanie operacji ETL (22)
  - Archiwizowanie (23)
  - Odtwarzanie i restartowanie (24)
  - Kontrola wersji (25)
  - Migrowanie rozwiązań ze środowiska programistycznego do produkcyjnego (26)
  - Monitorowanie przepływu zadań ETL (27)
  - Sortowanie danych (28)
  - Analizowanie pochodzenia i zależności pomiędzy danymi (29)
  - Informowanie o błędach (30)
  - Zrównoleglanie działań i potokowa ich realizacja (31)
  - Zapewnienie bezpieczeństwa (32)
  - Monitorowanie i audyt poszczególnych etapów procesów ETL – pochodzenia danych, operacji wykonywanych, wyglądu danych na poszczególnych etapach przetwarzania itp. (33)
  - Zarządzanie repozytorium metadanych (34)

Przykładowe działanie Archiwizowanie

- Archiwizowanie to podstawowy mechanizm zabezpieczający przed utratą danych
- W kontekście procesów ETL zaleca się archiwizację w trzech momentach
  - bezpośrednio po ekstrakcji danych i przed ich modyfikacją
  - po oczyszczaniu danych (zestaw działań 2)
  - po ostatecznym przygotowaniu danych do ich udostępnienia
- Archiwizacja samej Hurtowni Danych nie należy do zadań ETL, jednak warto uwzględnić/dostosować również ten aspekt przy projektowaniu procesów ETL

Architektura Hurtowni Danych repozytoria dla procesów ETL



Ale zanim…
Gdzie są wykonywane poszczególne działania wchodzące w skład procesów ETL?

Data Warehouse Staging Area
co to jest?

- To miejsce składowania danych oraz zbiór procesów wykorzystywanych podczas przetwarzania ETL mających miejsce pomiędzy:
  - źródłami danych a hurtownią danych
  - hurtownią danych a data martami

Data Warehouse Staging Area
jakie ma cechy?

- Dane przechowywane w DSA:
  - mogą mieć charakter tymczasowy – po poprawnym załadowaniu danych mogą być kasowane,
  - mogą służyć jako słowniki – które np. są wykorzystywane przez procesy ETL,
  - mogą zawierać poprzednie obrazy danych źródłowych – np. w celu porównania ich z obecną ich postacią,
  - mogą zawierać dane pośrednie uzyskiwane podczas procesów ETL

Data Warehouse Staging Area
do czego może być wykorzystywany?

- Do przechowywania danych z różnych źródeł, które następnie będą **wielokrotnie wykorzystywane** w ramach procesów ETL.
- Do przechowywania danych wydobytych z systemów źródłowych w sposób szybki i prosty, aby następujące po tym procesy ETL **nie angażowały** systemów **źródłowych**.
- Do **wyszukania zmian** pomiędzy systemem źródłowym w obecnej i przeszłej postaci.
- Do wyszukania zmian w bieżącej postaci hurtowni danych i data martów.
- Do **transformacji** danych realizowanych **wieloetapowo** ze składowaniem pośrednich efektów przetwarzania.
- Do wyznaczania **agregatów**.
- Do wytworzenia i składowania danych w postaci docelowej dla hurtowni danych (w szczególności tabel faktów) w celu ich wydajnego ładowania.

Operacyjna Składnica Danych (Operational Data Store) – charakterystyka

- **Tematyczna** (ang. *Subject Oriented*) – dane w ODS dotyczą określonego zagadnienia.
- **Zintegrowana** – dane w ODS pochodzą zazwyczaj z wielu źródłowych aplikacji. Dane są pobierane i przetwarzane w ramach procesów ETL.
- **Aktualna** – dane w ODS są bieżące. Nie przechowujemy danych historycznych.
- **Szczegółowa** – dane w ODS przechowywane są na poziomie szczegółów (bardzo często na poziomie analogicznym do systemów źródłowych).

Narzędzia ETL czy własne rozwiązania?

- Oba podejścia mają swoje wady i zalety
- Generalnie zaleca się wykorzystywanie narzędzi ETL, jednak w niektórych indywidualnych podejściach własne rozwiązania w dalszym ciągu są uzasadnione

- Narzędzia ETL – zalety
  - Wizualizują procesy ETL – automatyczna dokumentacja
  - Strukturalne podejście do projektowania ETL
  - Stabilne mechanizmy uruchamiania i monitorowania procesów ETL
  - Mechanizmy analizy pochodzenia i zależności danych
  - Zaawansowane komponenty czyszczenia danych
  - Wydajność
- Narzędzia ETL – wady
  - Koszty licencji
  - Konieczna dobra znajomość narzędzia – ich możliwości i ograniczeń
  - Ograniczona elastyczność