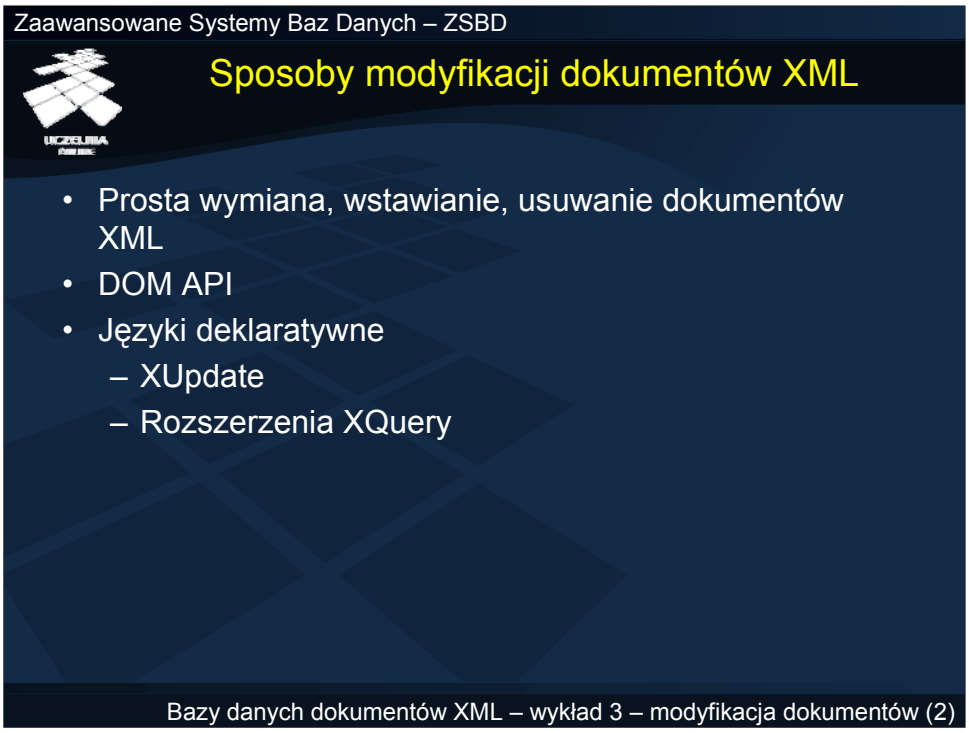
Trzeci wykład dotyczący baz danych dokumentów XML zostanie poświęcony sposobom modyfikacji dokumentów przechowywanych w bazach danych dokumentów XML. Jak do tej pory brak standardu takiego języka, mimo iż od kilku lat wykorzystywanych jest kilka bardziej lub mniej powszechnych rozwiązań.



Jak do tej pory brak standardu języka zapytań przeznaczonego do modyfikacji dokumentów XML w bazach danych dokumentów XML. Cześć z baz danych tego typu udostępnia jedynie mechanizmy pozwalające na wstawianie, usuwanie i wymianę całych dokumentów.

Mimo braku standardu udostępniane są różnego rodzaju interfejsy pozwalające użytkownikom na zmianę przechowywanych w bazach danych dokumentów XML.

Bazy danych, które umożliwią modyfikację fragmentów dokumentów XML stosują następujące podejścia:

- umożliwiają wykonywanie operacji DOM na dokumentach w nich zawartych;

- umożliwiają wykorzystanie wyrażeń XPath, które wskazują węzły, na których można przeprowadzić jedną lub wiele operacji takich jak:

  * wstawienie węzła przed lub po wskazywanych przez wyrażenia XPath fragmentach,

  * modyfikacja wskazywanego węzła,

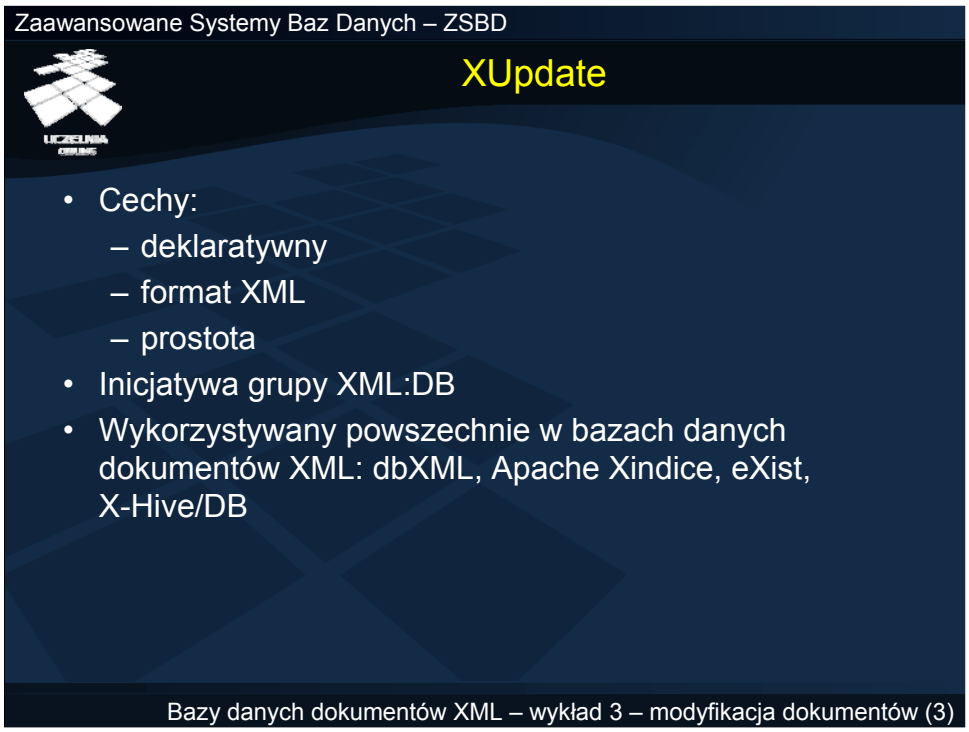
  * usunięcie wskazywanego węzła,

  * utworzenie zmiennej, której zawartość będzie identyczna ze wskazywanym węzłem,

  * zmiana nazwy znacznika wskazywanego elementu,

- wykorzystanie rozszerzeń języka XQuery;

Język XUpdate należy do drugiej grupy propozycji i on zostanie przestawiony w ramach tego wykładu jako pierwszy. Druga część wykładu skupi się na propozycji rozszerzeń języka XQuery.



Język XUpdate jest językiem stosunkowo szeroko rozpowszechnionym. Jego podstawowe cechy to:

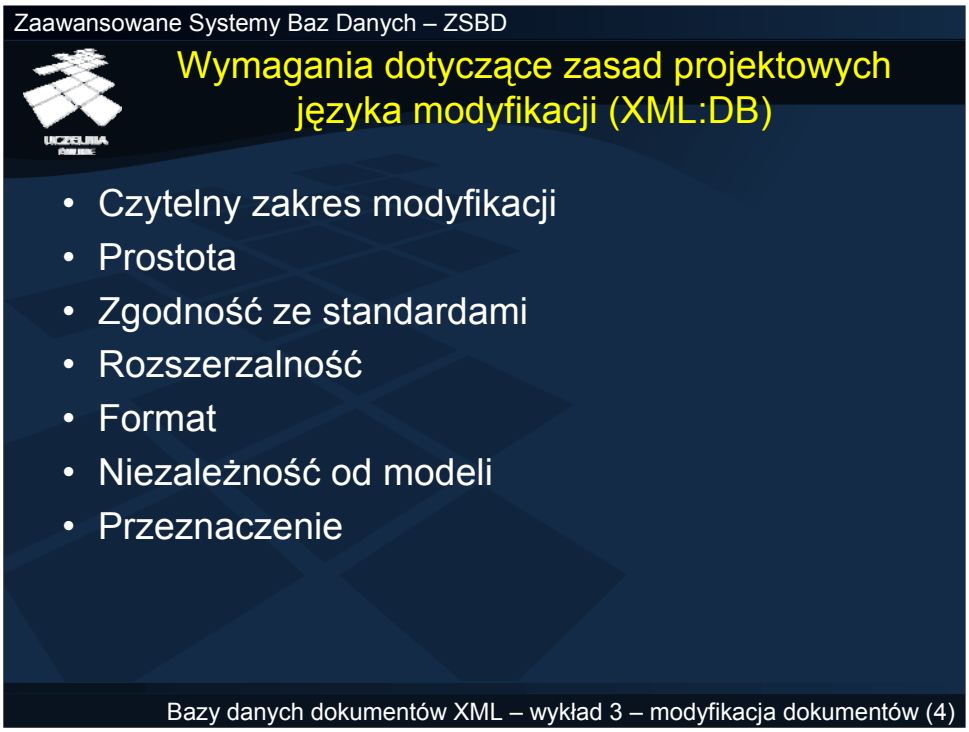
- deklaratywny charakter języka,

- format dokumentu XML,

- prostota.

Język XUpdate powstał jako inicjatywa grupy XML:DB. Ta sama grupa stworzyła XML:DB API.

XUpdate jest przykładowo wykorzystywany w bazach danych dokumentów XML takich jak: dbXML, Apache, Xindice, eXist, X-Hive/DB.



Analogicznie do wymagań narzuconych językom zapytań przeznaczonych dla baz danych dokumentów XML przez organizację W3C, XML:DB zdefiniowała własne wymagania dotyczące języka modyfikacji dokumentów XML. Wymagania te zostały podzielone na dwa zakresy: zasady projektowe oraz funkcjonalność.

Wymagania dotyczące zasad projektowych są następujące:

- zakres – język modyfikacji musi opisywać to w jaki sposób mają być wykonywane zapytania i modyfikacja zawartości XML;

- prostota – język modyfikacji powinien być prosty, o dużej sile wyrazu i czytelny dla autora;

- zgodność ze standardami – język modyfikacji powinien być zgodny ze standardami dotyczącymi: przestrzeni nazw, wyrażeń XPath, standardu XPointer;

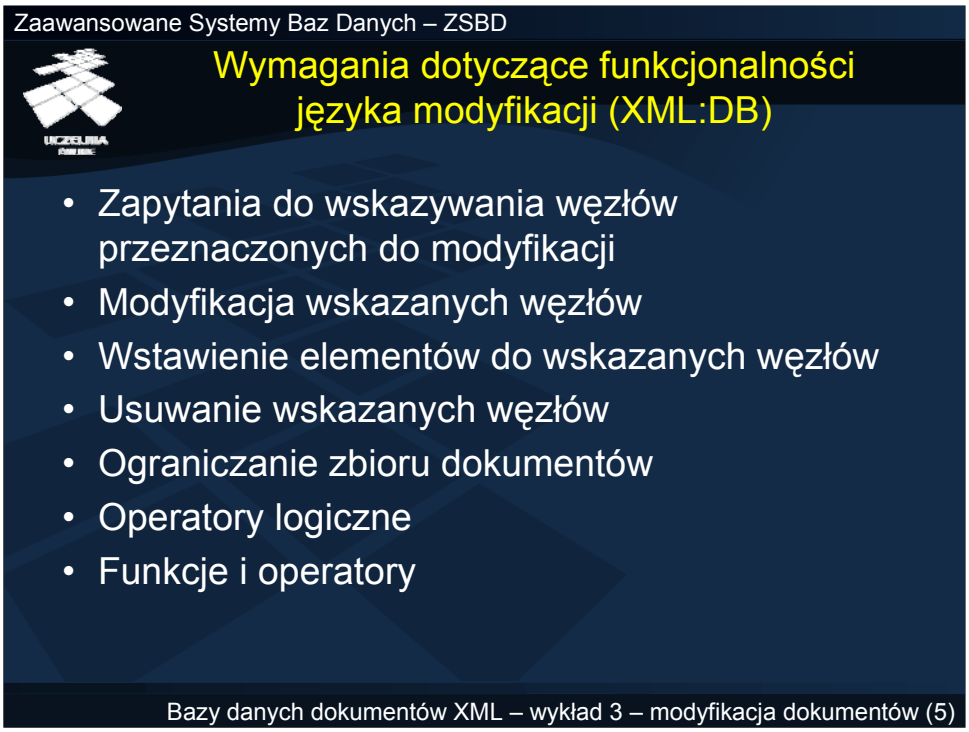
- rozszerzalność – musi być otwarty na przyszłe rozszerzenia;

- format – specyfikacja języka musi być elementem XML;

- niezależność od modeli – specyfikacja języka nie może być zaprojektowana w sposób uzależniający język od obiektów XML lub modeli parserów takich jak DOM czy SAX;

- przeznaczenie – język musi być zdolny do wykonywania działań na rzecz części lub całości jednego lub wielu dokumentów XML;

- integracja – rozwój specyfikacji języka modyfikacji wymaga interakcji z innymi grupami roboczymi XML:DB. W szczególności z grupą roboczą XML Database API Working Group.



Omówimy teraz wymagania dotyczące funkcjonalności języka. Są one następujące:

- zapytania – język musi udostępniać prostą funkcjonalność dotyczącą zapytań. Zapytanie jest wstępnym etapem polegającym na wybraniu zbioru węzłów przeznaczonych do modyfikacji;

- modyfikacja – język musi definiować mechanizm modyfikacji. Modyfikacja jest wyrażeniem sposobu modyfikacji zawartości XML wskazanej w zapytaniu;

- wstawianie – język musi umożliwiać wstawianie nowych elementów XML do węzłów uzyskanych w wyniku zapytania;

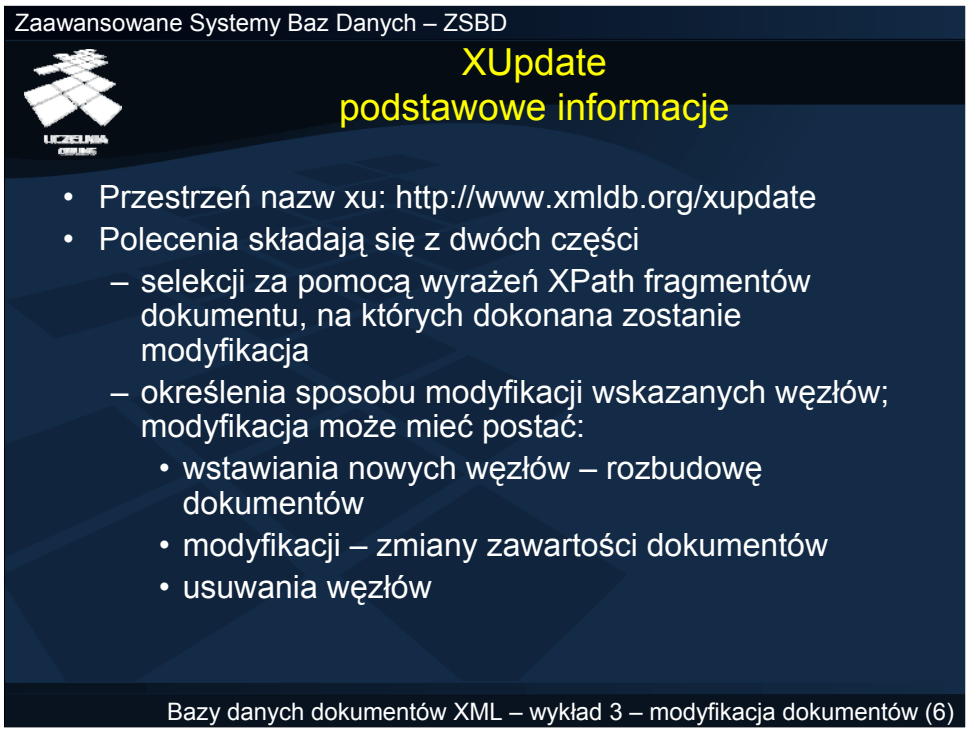
- usuwanie – język musi umożliwiać usuwanie elementów XML z węzłów uzyskanych w wyniku zapytania;

- ograniczanie – język musi posiadać mechanizmy ograniczające zbiór przepytywanych dokumentów;

- operatory logiczne – język modyfikacji musi zawierać zbiór operatorów przeznaczonych do operacji logicznych. Zbiór ten powinien zawierać takie operatory jako 'and', 'not', 'or';

- logika – język powinien udostępniać minimalny zbiór operatorów i funkcji pozwalających operować na danych liczbowych, datach itp.

Język XUpdate oczywiście spełnia wszystkie te wymagania.



Wszystkie polecenia XUpdate zostały umieszczone w przestrzeni nazw: http://www.xmldb.org/xupdate.

Polecenia dotyczące modyfikacji dokumentu XML przeważnie składają się z dwóch części:

- zadaniem pierwszej części jest wskazanie za pomocą wyrażeń XPath tych fragmentów dokumentu XML, które mają podlegać modyfikacji lub innej operacji;

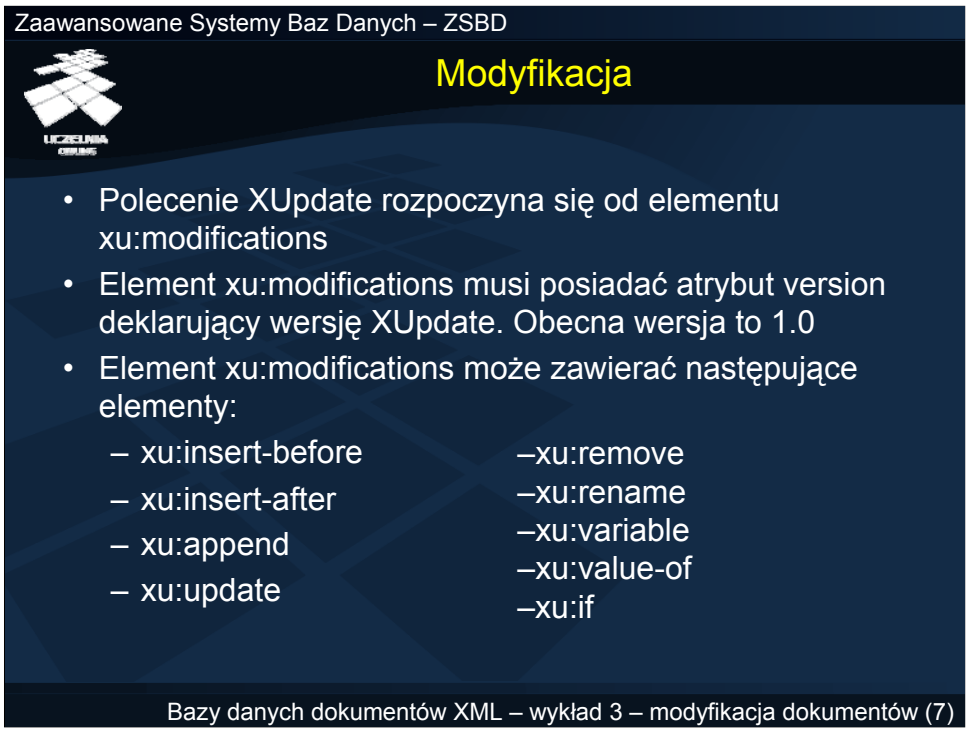
- zadaniem drugiej części polecenia XUpdate jest przeważnie określenie sposobu modyfikacji wskazanych węzłów.

Modyfikacja dokumentów XML za pomocą wyrażeń XUpdate może, między innymi, mieć postać:

- wstawiania nowych węzłów – rozbudowę dokumentów,

- modyfikacji – zmiany zawartości wskazywanych węzłów,

- usuwania węzłów.



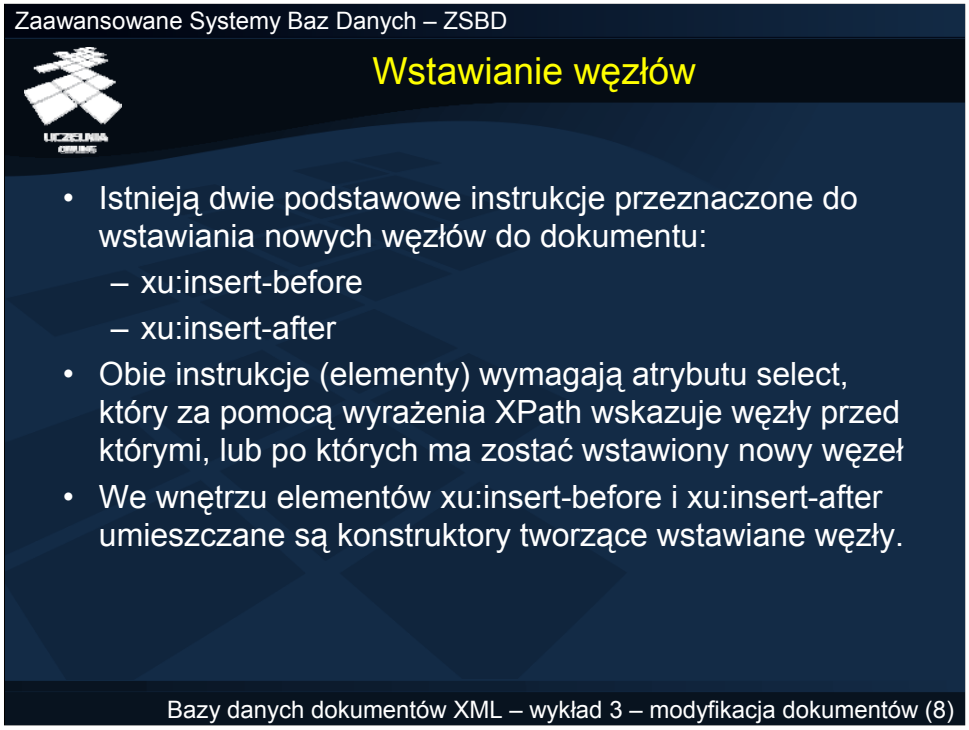
Polecenie modyfikacji w języku XUpdate rozpoczyna się elementem xu:modifications, który otacza szczegółowe dyrektywy dotyczące konkretnych zmian.

Element xu:modifications musi posiadać atrybut version deklarujący wersję XUpdate. Obecna wersja to 1.0. Dodatkowo powinien posiadać deklarację przestrzeni nazw XUpdate. Z reguły deklaracja przestrzeni nazw tworzy prefiks o nazwie xupdate, choć może przyjmować również inne wartości, dla przykładu xu. W dalszej części wykładu będzie wykorzystywany prefiks xu.

Element xu:modifications może zawierać następujące elementy:

- xu:insert-before,
- xu:insert-after,
- xu:append,
- xu:update,
- xu:remove,
- xu:rename,
- xu:variable,
- xu:value-of,
- xu:if.

Znaczenie wymienionych elementów zostanie szczegółowo omówione na następnych slajdach.



Istnieją dwie podstawowe instrukcje przeznaczone do wstawiania nowych węzłów do dokumentu:

- xu:insert-before – wstawia nowy węzeł przed wskazanym istniejącym już węzłem

- xu:insert-after – wstawia nowy węzeł bezpośrednio po wskazanym istniejącym już węźle

Obie instrukcje (elementy) wymagają atrybutu select, który za pomocą wyrażenia XPath wskazuje węzły przed którymi, lub po których mają zostać wstawione nowe węzły.

Elementy xu:insert-before i xu:insert-after mogą zawierać elementy (konstruktory) tworzące węzeł, który ma zostać wstawiony. Typ nowego węzła jest zależny od użytego konstruktora. Możliwe konstruktory to:

- xu:element – tworzy nowy węzeł będący elementem,

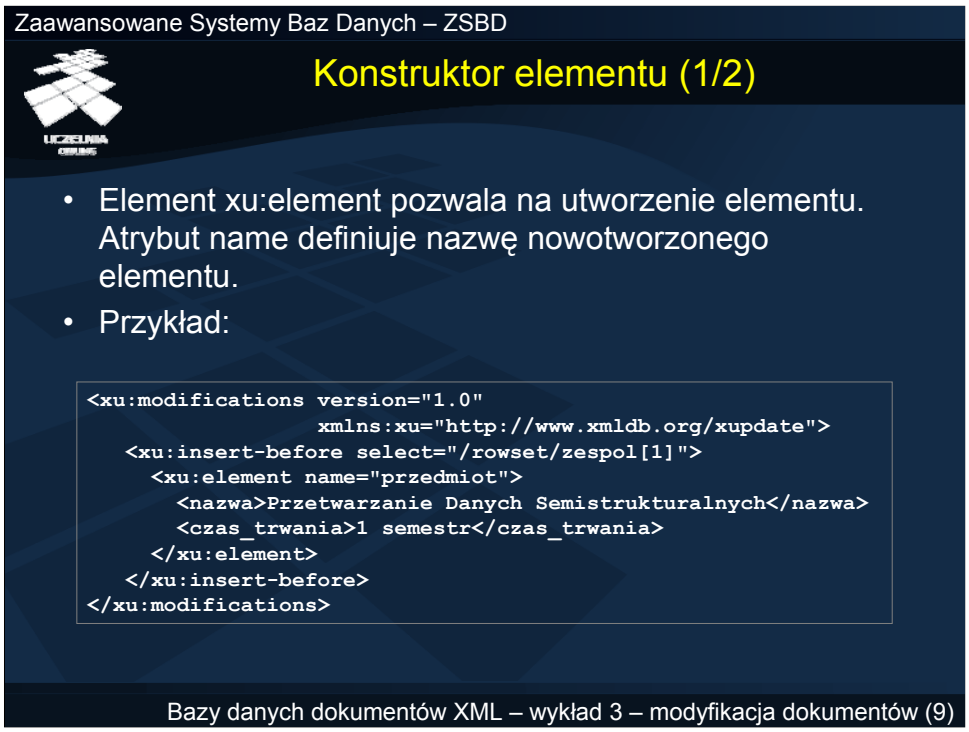
- xu:attribute – tworzy nowy węzeł będący atrybutem,

- xu:text – tworzy nowy węzeł tekstowy,

- xu:processing-instruction – tworzy węzeł będący instrukcją przetwarzania,

- xu:comment – tworzy nowy węzeł będący komentarzem.

Na następnych slajdach przedstawione zostaną przykłady wykorzystania powyższych konstruktorów.

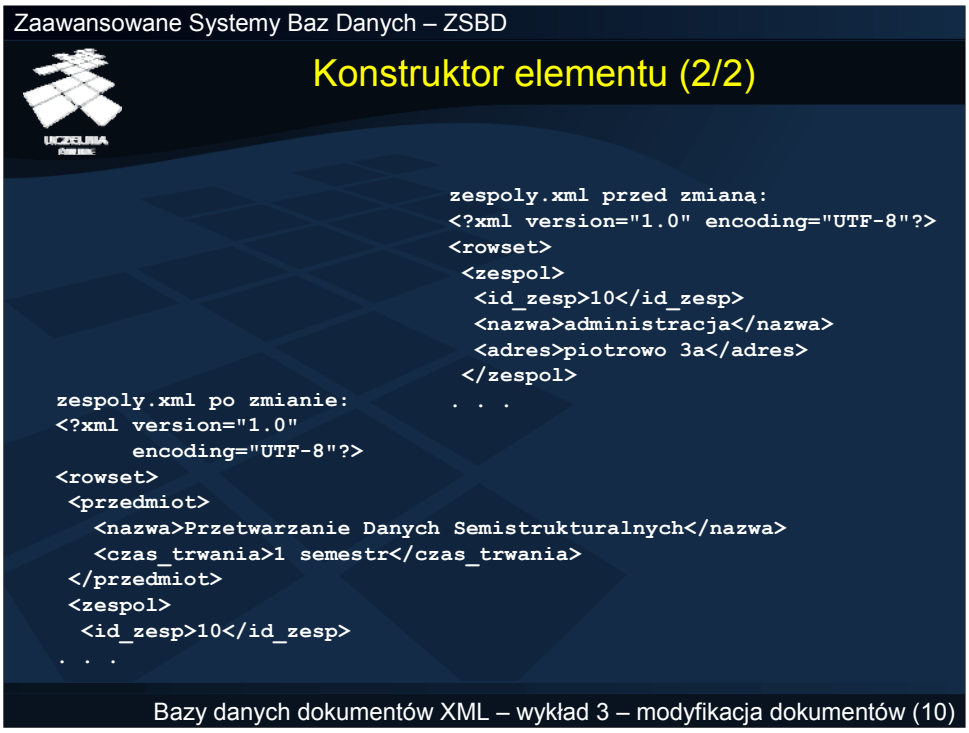


Element xu:element pozwala na utworzenie elementu. Atrybut name definiuje nazwę nowotworzonego elementu.

Dla przykładu, polecenie umieszczone na slajdzie zawiera polecenie modyfikacji bieżącego dokumentu lub dokumentów znajdujących się w bieżącej kolekcji.

Atrybuty version oraz deklaracja przestrzeni nazw została umieszczona w elemencie xu:modifications.

Modyfikacja polega na dodaniu węzła przed węzły wskazywane przez atrybut select w elemencie xu:insert-before. W naszym przypadku dodawany węzeł zostanie umieszczony przed pierwszym elementem zespol znajdującym się we wnętrzu elementu rowset. Dodawanym węzłem będzie element o nazwie przedmiot>. Wynika to z typu elementu konstruktora, w tym przypadku xu:element, oraz wartości jego atrybutu name. Zawartością dodawanego elementu przedmiot będzie para elementów nazwa i czas_trwania. Zawartość dodawanego elementu została utworzona bez wykorzystania elementów będących konstruktorami.



Na slajdzie przedstawiono przykładowy dokument zespoly.xml przed zmianą za pomocą omawianego polecenia i postać tego samego dokumentu po zastosowaniu modyfikacji XUpdate.

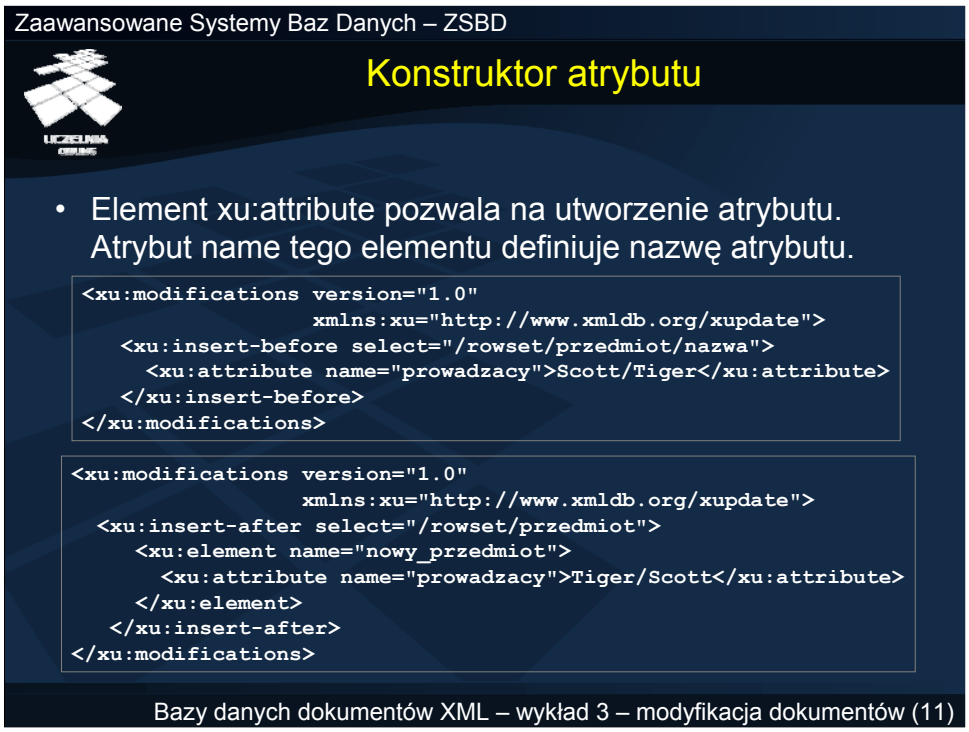
Zwróćmy uwagę na dodany nowy element przedmiot i jego zawartość.

Wśród wymagań narzuconych na język modyfikacji dokumentów XML jest takie, które mówi, że język musi posiadać mechanizmy ograniczające zbiór przepytywanych i modyfikowanych dokumentów. Zwróćmy uwagę na to, że składnia XUpdate nie pozwala na wskazywanie dokumentów XML lub ich zbiorów. Jak zatem to wymaganie jest spełnione?

Polecenie modyfikacji języka XUpdate dotyczy mianowicie tzw. bieżących dokumentów lub bieżącej kolekcji. W poleceniu XUpdate nie ma możliwości bezpośredniego wskazania na dokument będący odbiorcą modyfikacji (można to zrobić jedynie pośrednio odpowiednio definiując wyrażenia ścieżkowe zapytania). A zatem, użytkownik przed wykonaniem modyfikacji musi w bazie danych dokumentów XML wskazać, który z dokumentów lub, która z kolekcji będzie uznana za bieżącą.

Taki sposób rozwiązania problemu ograniczania zbioru modyfikowanych dokumentów ma jedną wadę. Nie ma takim przypadku możliwości modyfikacji zbioru dokumentów w oparciu o informacje zawarte w innym zbiorze dokumentów.

A zatem, dla przykładu, nie będzie możliwości modyfikacji dokumentów dotyczących zespołów na podstawie dokumentów dotyczących pracowników, jeżeli dokumenty te nie współdzielą jednej kolekcji.

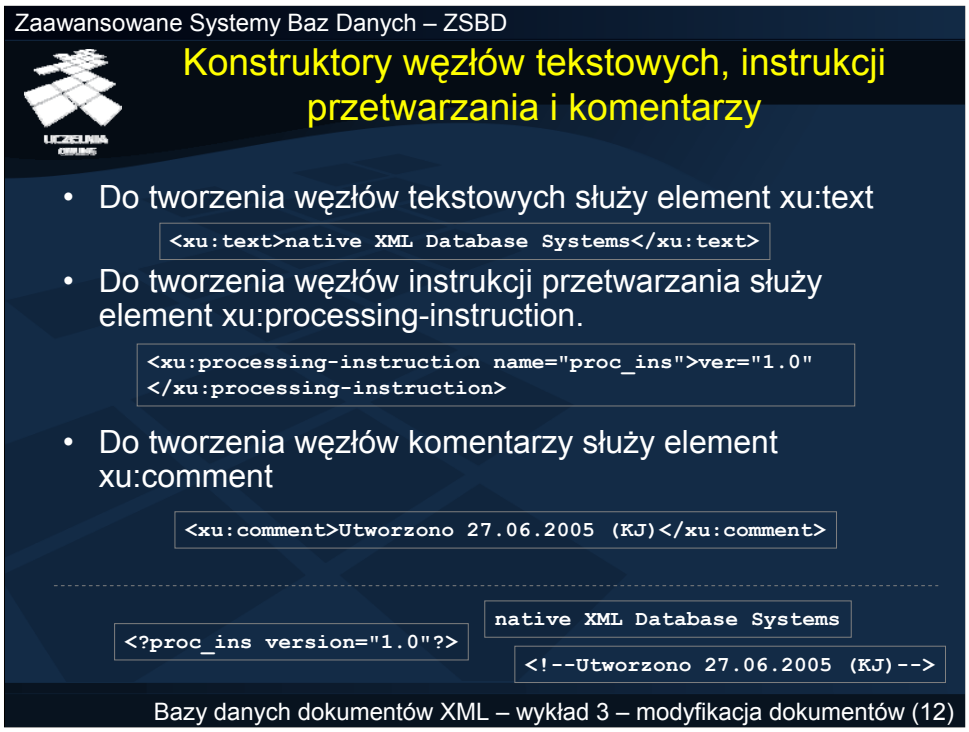


Element xu:attribute pozwala na utworzenie atrybutu. Atrybut name tego elementu definiuje nazwę atrybutu.

Element xu:attribute może zostać użyty jako samodzielny obiekt wstawiany do dokumentu lub też jako uzupełnienie definicji wstawianego elementu. W pierwszym przypadku należy pamiętać, że atrybut umieszczany będzie zawsze we wnętrzu znacznika rozpoczynającego określony element.

Przykład użycia elementu xu:attribute, jako samodzielnego elementu, został zamieszczony w górnej części slajdu. Polecenie to utworzy atrybut jako węzeł przed elementem nazwa w elemencie przedmiot. Oznacza to, że atrybut ten w rzeczywistości stanie się elementem wewnętrznym elementu przedmiot – będzie jego nowym atrybutem.

Przykład użycia elementu xu:attribute, jako fragmentu definicji nowowstawianego elementu, został przedstawiony poniżej i nie wymaga szerszego komentarza.



Oprócz węzłów elementów i atrybutów, w dokumencie XML-owym, mogą znajdować się także węzły innego typu, takie jak: węzły tekstowe, węzły instrukcji przetwarzania oraz węzły komentarzy. Dla każdego z nich istnieje indywidualny konstruktor w polecenia XUpdate.

I tak:

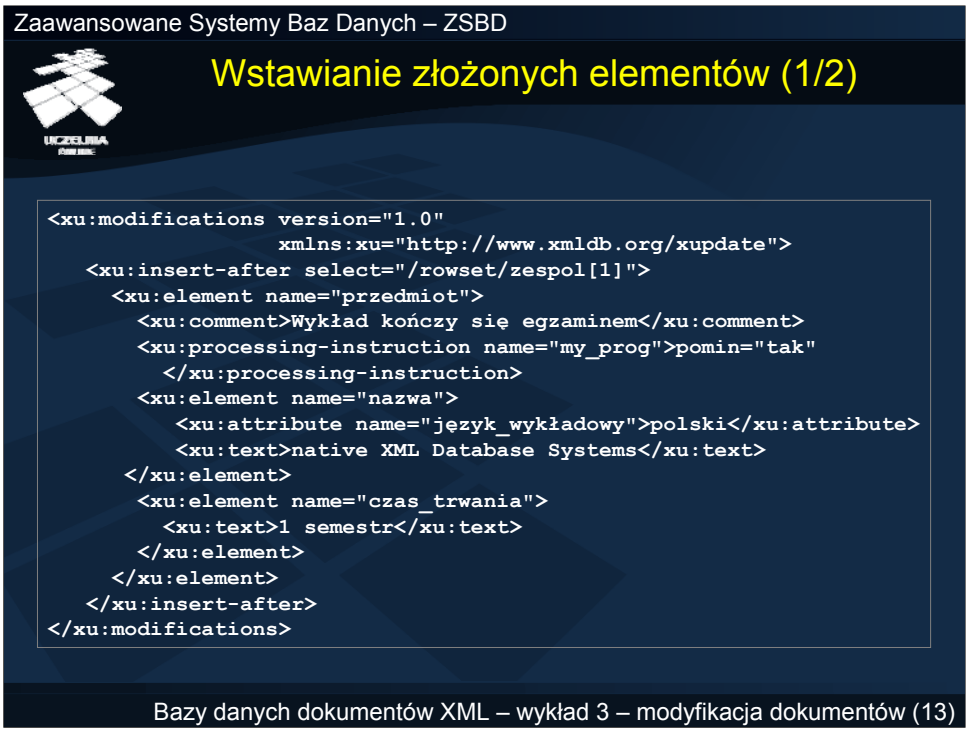
- do tworzenia węzłów tekstowych służy element xu:text,

- do tworzenia węzłów instrukcji przetwarzania służy element xu:processing-instruction. Atrybut name tego elementu definiuje nazwę węzła instrukcji przetwarzania,

- do tworzenia węzłów komentarzy służy element xu:comment.

Na slajdzie umieszczono po jednym przykładzie dla każdego z trzech omawianych konstruktorów. Ich użycie spowoduje utworzenie odpowiednio węzła tekstowego, instrukcji przetwarzania, oraz komentarza z informacją dotyczącą czasu utworzenia aktualnego slajdu. Obiekty utworzone przez omawiane konstruktory zostały przedstawione w dolnej części slajdu.

Wszystkie te elementy mogą być wykorzystane w dwojaki sposób, albo jako konstruktory indywidualnych nowotworzonych węzłów, albo też jako fragmenty większych konstrukcji.

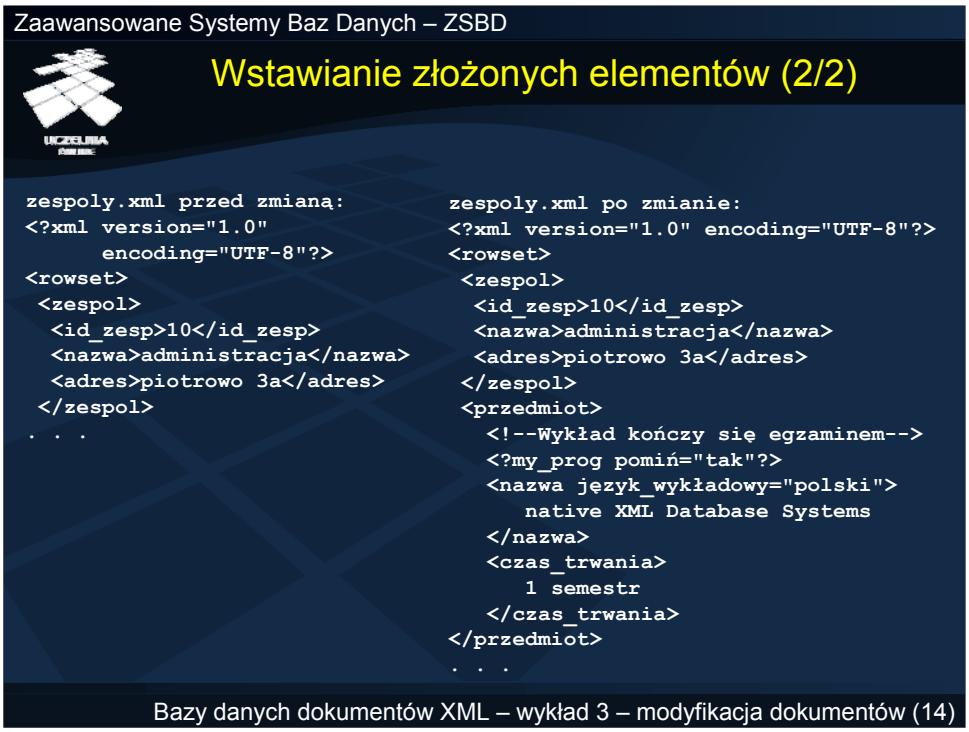


Tak jak wspomniano wcześniej, poszczególne elementy można wykorzystywać jako samodzielne wstawiane węzły, lub jako konstrukcje większej całości. Jako przykład niech posłuży nam polecenie zamieszczone na slajdzie.

Pierwszy element xu:modifications rozpoczyna polecenie XUpdate, atrybuty w nim zawarte definiują przestrzeń nazw XUpdate oraz wskazują na wersję języka XUpdate.

Kolejny element xu:insert-after jest pierwszym i ostatnim poleceniem zawartym w analizowanym wyrażeniu XUpdate. Polecenie to będzie wstawiało nowy węzeł po węźle wskazanym za pomocą atrybutu select. Wartość atrybutu select decyduje o tym, że nowo-wstawiany węzeł będzie umieszczony po pierwszym elemencie zespol znajdującym się we wnętrzu elementu rowset. Wewnątrz elementu xu:insert-after rozpoczyna się definicja nowowstawianego węzła.

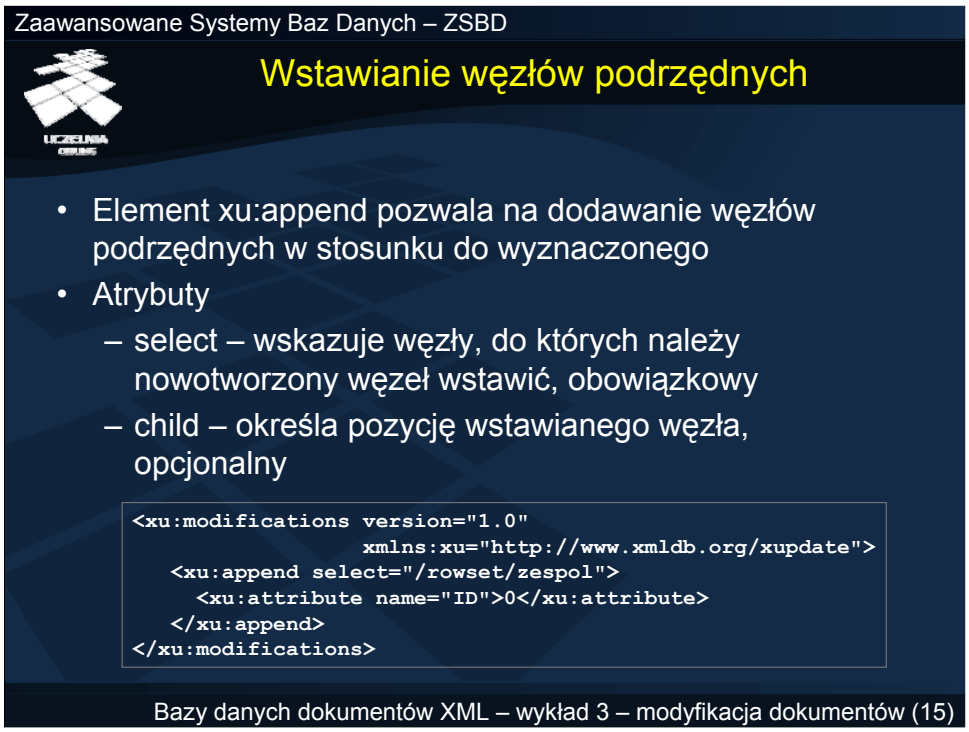
Element xu:element i jego atrybut name świadczy o tym, że nowowstawiany węzeł będzie elementem przedmiot. Jego zawartość rozpoczyna się od komentarza i instrukcji przetwarzania. Następnie rozpoczyna się podelement nazwa. Zawartość podelementu nazwa została zdefiniowana za pomocą dwóch konstruktorów xu:attribute i xu:text. Definicja elementu nazwa na tym się kończy. Bezpośrednio za nim rozpoczyna się definicja kolejnej składowej elementu przedmiot. Jest nią element czas_trwania, którego zawartością jest węzeł tekstowy.



Na slajdzie został przedstawiony przykładowy dokument zespoly.xml przed i po zmianie za pomocą omawianego polecenia.

Zakładamy że dokument zespoly.xml był dokumentem roboczym (bieżącym) podczas wykonywania polecenia XUpdate.

Zwróćmy uwagę na nowy element przedmiot wstawiony bezpośrednio za pierwszym elementem zespol i jego złożoną zawartość utworzoną przy wykorzystaniu omówionych wcześniej konstruktorów.



Elementy xu:insert-before i xu:insert-after dodają węzły przed lub po węzłach wskazywanych przez atrybut select. Nie jest to czasami wygodne rozwiązanie. Za pomocą elementu xu:append możemy zdefiniować polecenie polegające na dodaniu węzła podrzędnego w stosunku do wyznaczonego węzła. Instrukcja xu:append posiada dwa atrybuty:

- select – za pomocą którego możemy wskazać węzły, do których należy nowotworzony węzeł wstawić. Jest to atrybut obowiązkowy;

- child – który definiuje pozycję dla wstawianego węzła. Typ wyrażenia występującego jako wartość atrybutu child to liczba całkowita. Jest to atrybut opcjonalny, jeśli zostanie on pominięty, wówczas nowotworzony węzeł zostanie umieszczony za ostatnim potomkiem wskazywanego węzła.

Podobnie jak w przypadku poleceń xu:insert-before i xu:insert-after, element xu:append może zawierać następujące elementy:

- xu:element,

- xu:attribute,

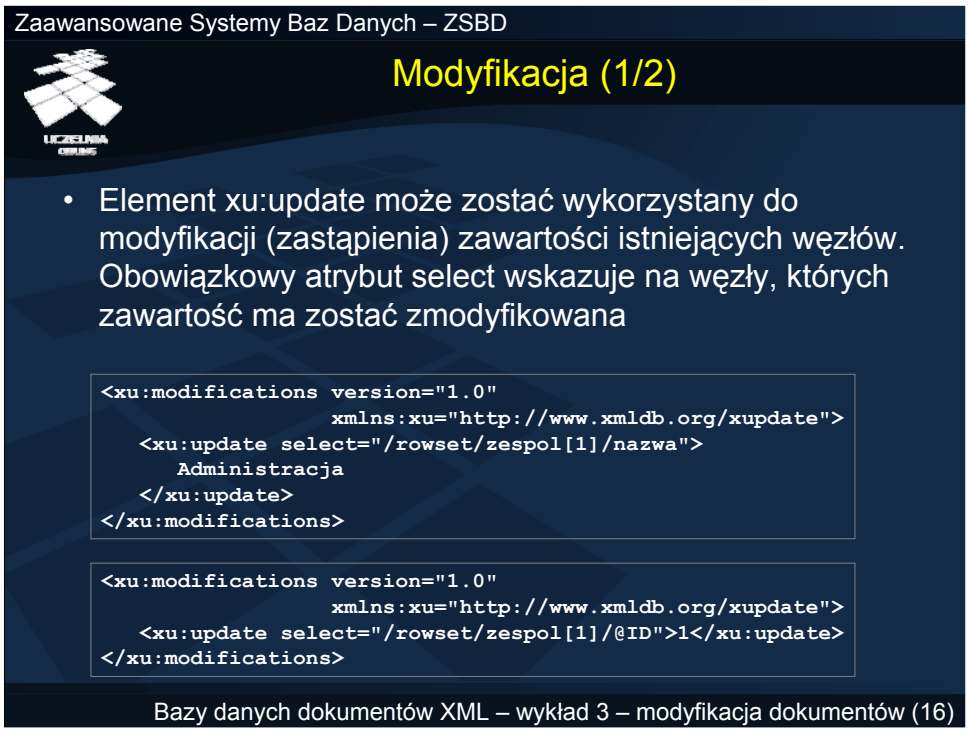
- xu:text,

- xu:processing-instruction,

- xu:comment.

Na slajdzie przedstawiony został przykład wstawiający do każdego elementu zespół atrybut ID o wartości równej 0.

W przypadku wstawiania atrybutu pominięcie atrybutu child było zamierzone. Kolejność atrybutów we wnętrzu elementów XML nie ma bowiem znaczenia.



Za pomocą instrukcji XUpdate możemy także modyfikować zawartość węzłów istniejących w dokumencie. W tym celu może zostać wykorzystany element xu:update. Jego zadaniem jest zastąpienie zawartości określonych węzłów inną zawartością zdefiniowaną we wnętrzu elementu xu:update. Obowiązkowy atrybut select wskazuje na węzły, których zawartość ma zostać wymieniona.

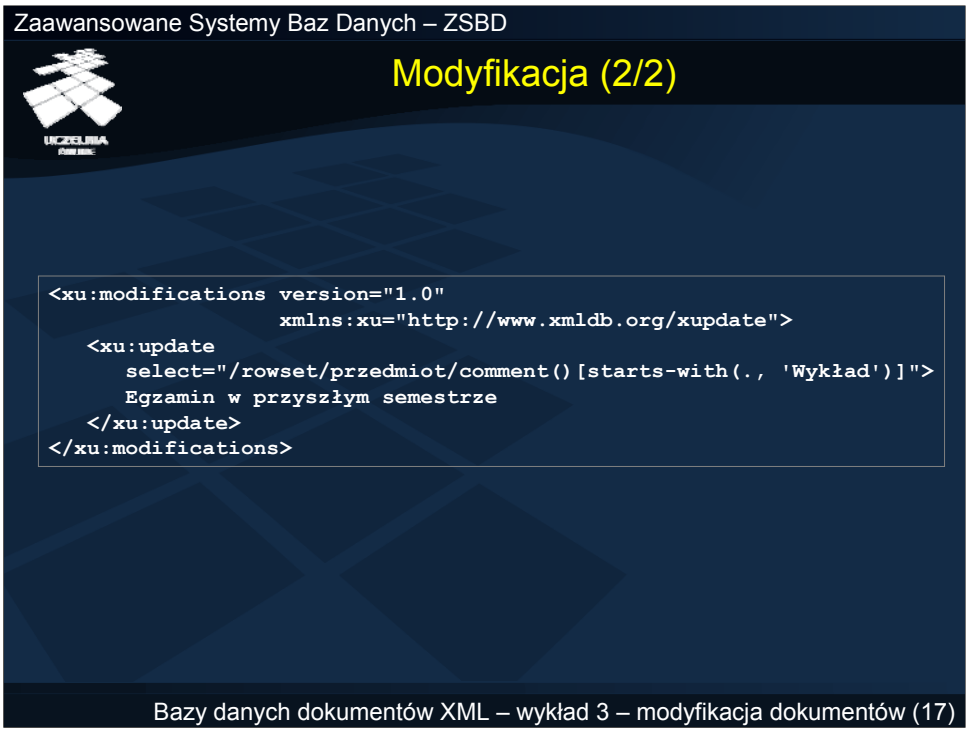
Wymiana zawartości może dotyczyć węzłów prostych takich jak: komentarze, atrybuty, węzły tekstowe i węzły elementów prostych. Wartość, na którą zawartość węzła będzie wymieniana musi być prosta. Oznacza to, że nie możemy, dla przykładu, zamienić elementu złożonego na zawartość złożoną.

Przykład modyfikacji zawartości elementu na zawartości prostą został umieszczony na slajdzie jako pierwszy.

Przykład modyfikacji zawartości atrybutu został umieszczony poniżej.

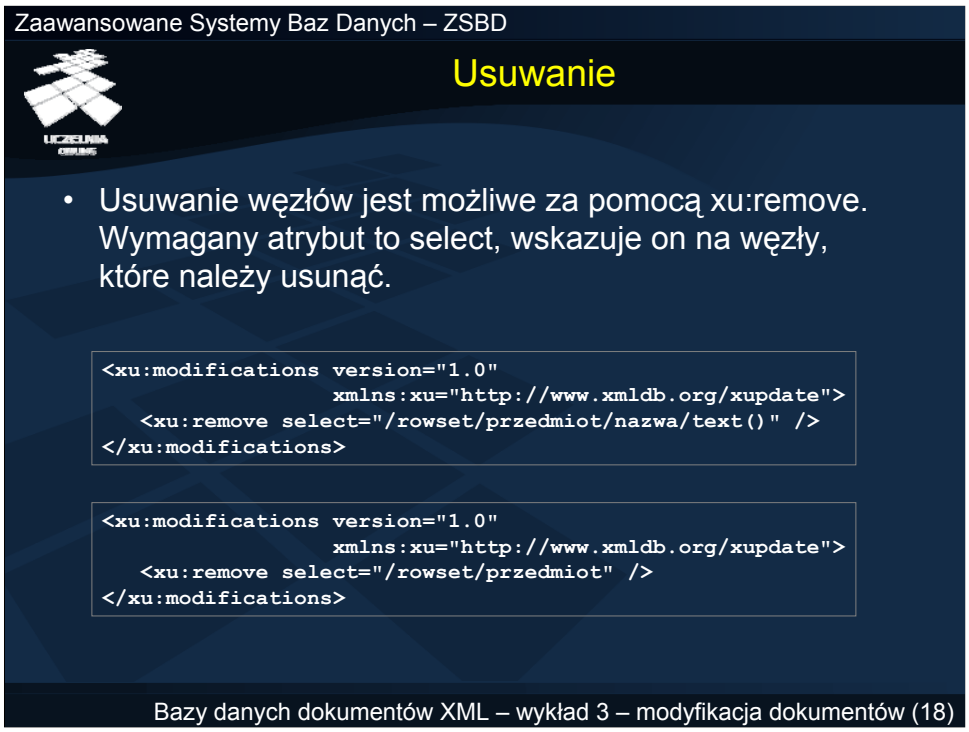
W przykładzie pierwszym modyfikacji podlega zawartość elementu nazwa pierwszego z zespołów. Zostaje ona w wyniku polecenia wymieniona na ciąg znaków Administracja.

W przykładzie drugim modyfikacji polega atrybut ID tego samego zespołu. Po wykonaniu polecenia będzie on posiadał wartość 1.



Przykład modyfikacji węzła komentarza umieszczonego we wnętrzu elementu przedmiot i rozpoczynającego się od słowa ‘Wykład’ został zamieszczony na bieżącym slajdzie. Jak widać modyfikacji mogą podlegać wszystkie elementy, które jesteśmy w stanie wskazać za pomocą wyrażeń XPath.

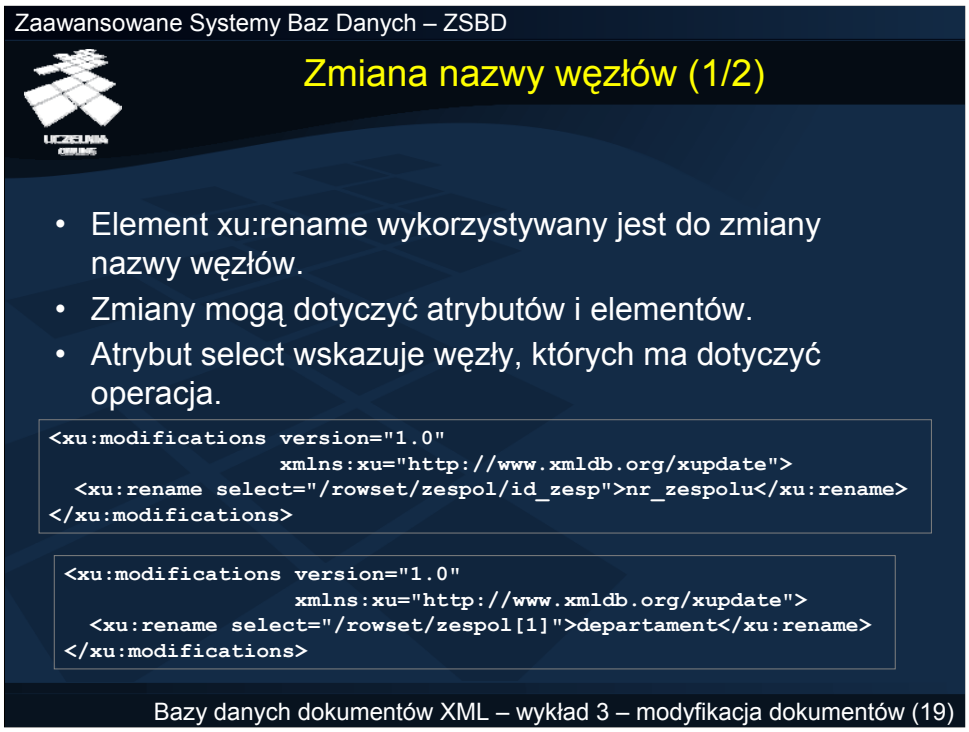
Należy zatem pamiętać, że podczas konstruowania poleceń XUpdate jest dostępna pełna funkcjonalność wyrażeń XPath, włącznie z predykatami, funkcjami itp..



Za pomocą polecenia xu:remove możemy usuwać niepotrzebne węzły istniejące w dokumencie. Usuwane mogą być węzły zarówno proste jak i złożone. Atrybut select elementu xu:remove wskazuje węzły mające podlegać usunięciu. Element xu:remove jest elementem pustym, czyli nie posiada żadnej zawartości.

Na slajdzie, jako pierwszy, zamieszczono przykład, który usunie zawartość tekstową elementów wskazywanych przez wyrażenie XPath "/rowset/przedmiot/nazwa/text()"

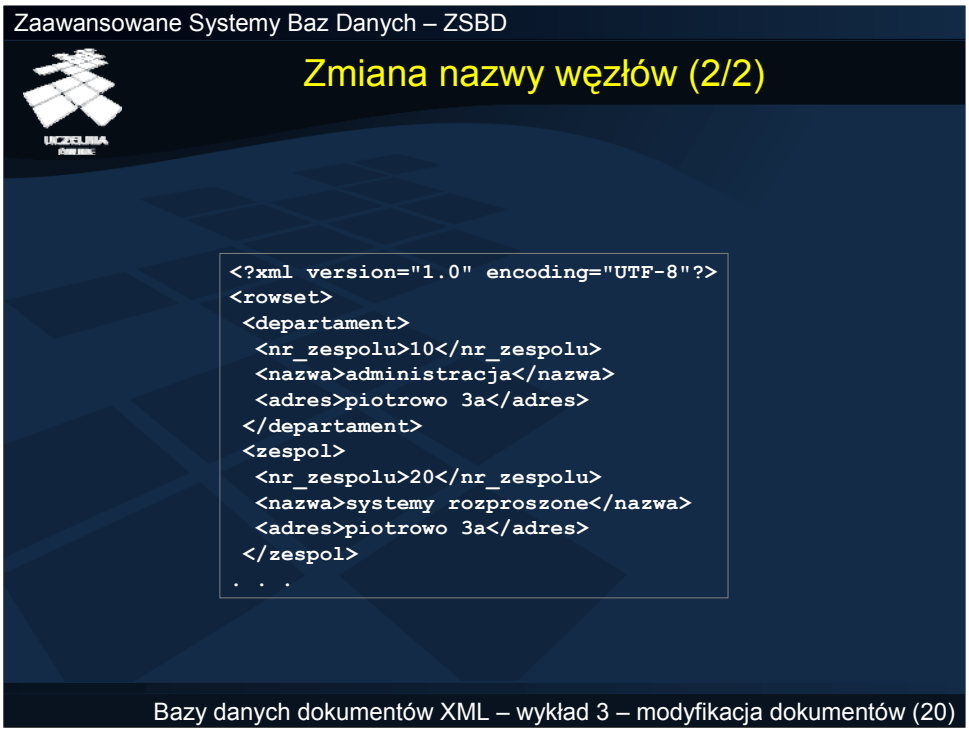
Poniżej zamieszczono przykład, który usunie elementy przedmiot umieszczone we wnętrzu elementów rowset.



Za pomocą elementu xu:rename użytkownik może zmienić nazwy wybranych węzłów. Zmiany nazw mogą dotyczyć atrybutów i elementów. Próba zmiany innego rodzaju węzłów kończy się błędem. Atrybut select elementu xu:rename wskazuje węzły, których ma dotyczyć operacja.

Przykładowa operacja przedstawiona na slajdzie jako pierwsza, pozostawi w efekcie dokument, w którym wszystkie elementy id_zesp będą zamienione na elementy nr_zespolu z zachowaniem ich zawartości.

Drugi przykład dokona zmiany nazwy pierwszego elementu zespol, po modyfikacji nowa nazwa tego elementu to departament.



Przykładowa postać dokumentu po modyfikacji została przedstawiona na slajdzie.

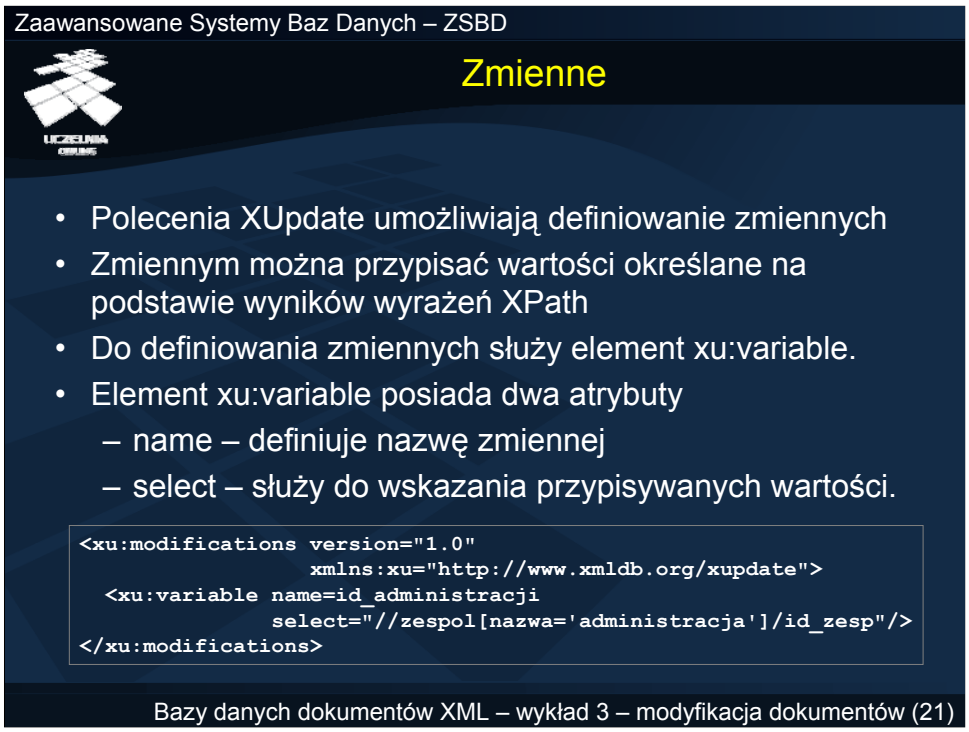
Zwróćmy uwagę na zmienione elementy.

Jak do tej chwili wszelkie modyfikacje oraz wstawianie nowych elementów powodowały, zmiany oparte na wartościach konstruowanych niezależnie od aktualnej zawartości dokumentu. Dla przykładu wstawialiśmy nowy element przedmiot, w którym wszystkie elementy definiowaliśmy od nowa w oparciu o konstruktory. Modyfikacja zawartości nazw zespołów czy atrybutów ID również była oparta o wartości stałe, niezależne od wartości oryginalnych.

Wyobraźmy sobie jednak bardziej złożony przypadek. Chcemy zmodyfikować dokument pracownicy.xml i zawarty w nim element pracownik dotyczący pracownika o nazwisku IKSIŃSKI. Chcemy aby wartość elementu placa_pod pracownika IKSIŃSKI posiadała wartość uzyskaną z elementu placa_pod pracownika YGREKOWSKI.

W takim przypadku konieczna jest modyfikacja zawartości dokumentu z uwzględnieniem jego zawartości początkowej.

Aby coś takiego było możliwe musimy we wnętrzu takich elementów jak xu:insert czy xu:append odwołać się do wartości uzyskanych uprzednio w wyniku zapytania. Możliwości takie dają zmienne, które zostaną omówione na kilku kolejnych slajdach.



Wewnątrz poleceń XUpdate użytkownik może definiować zmienne i przypisywać im wartości. Zmienne identyfikowane są przez nazwę, a wartością zmiennej może być obiekt dowolnego typu zwróconego przez wyrażenie XPath.

Do definiowania zmiennej służy element xu:variable. Posiada on dwa atrybuty:

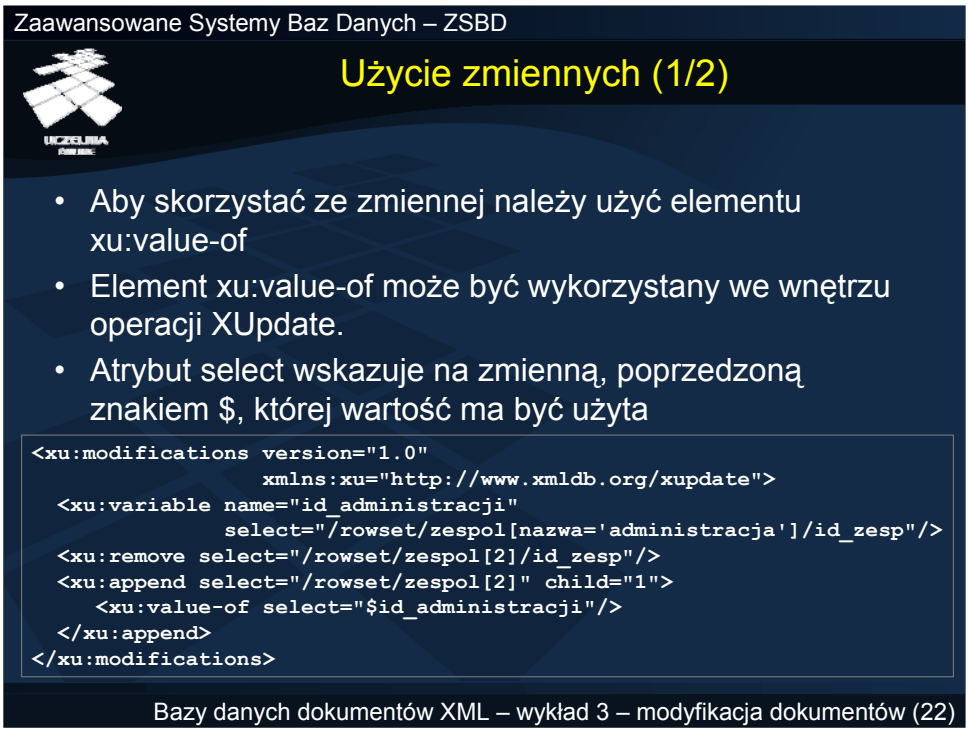
- name – służy do nadania nazwy zmiennej

- select – za pomocą wyrażenia XPath pozwala na przypisanie zmiennej określonej wartości

Przykładowo, polecenie przedstawione na slajdzie definiuje zmienną o nazwie id_administracji i przypisuje jej element id_zesp znajdujący się w elemencie zespol z podelementem nazwa, którego zawartość tekstowa to 'administracja'.

Wartością zmiennej może być zarówno pojedynczy węzeł, jak i las węzłów.

Zmienne można wykorzystać na wiele sposobów. Można za ich pomocą, dla przykładu, zmienić kolejność węzłów, przenieść węzeł w inne miejsce dokumentu czy też skopiować węzeł.

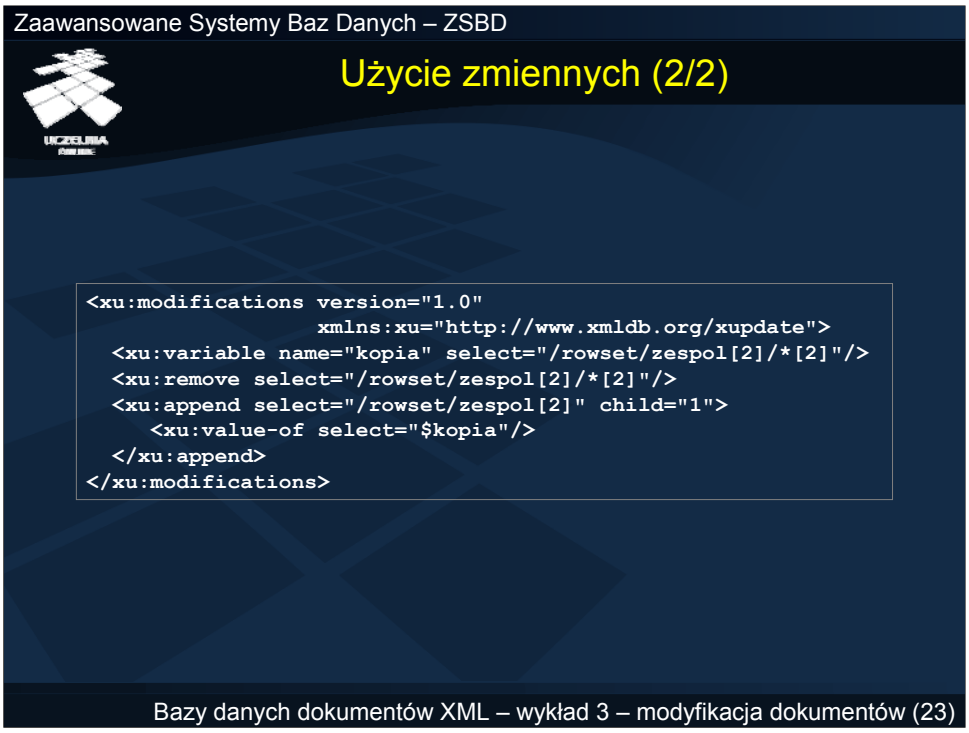


Aby skorzystać z wcześniej zdefiniowanej zmiennej należy użyć elementu xu:value-of.

Element xu:value-of pozwala na odwołanie się do wartości zmiennej wykorzystując ją w ramach operacji XUpdate. Atrybut select elementu xu:value-of służy do wskazania nazwy zmiennej, której wartość ma zostać wykorzystana. Nazwa zmiennej w atrybucie select musi być poprzedzona znakiem $.

Przykładowe użycie zmiennej zostało przedstawione na slajdzie.

W przykładzie do zmiennej id_administracji zostaje przypisany identyfikator zespołu administracja. Następnie w tym samym poleceniu zostanie usunięty element id_zesp w drugim z kolei zespole. W jego miejsce za pomocą instrukcji xu:append zostanie wstawiona wartość wcześniej utworzonej zmiennej.



W drugim przykładzie drugi element umieszczony w drugim elemencie zespol przenosimy na miejsce pierwsze. Innymi słowy zamieniamy pozycjami podelement pierwszy i drugi drugiego zespołu.

Tak jak wspomniano wcześniej przy użyciu zmiennych można: zmienić kolejność węzłów, przenieść węzeł w inne miejsce dokumentu czy też skopiować węzeł.

Nie można niestety ingerować w wartość przypisaną zmiennej co oznacza, że, dla przykładu, nie podniesiemy płacy IKSIŃSKIEMU o 10 zł, nie dodamy do istniejącej nazwy zespołu kolejnego członu np. "i Zarządzania" itp.

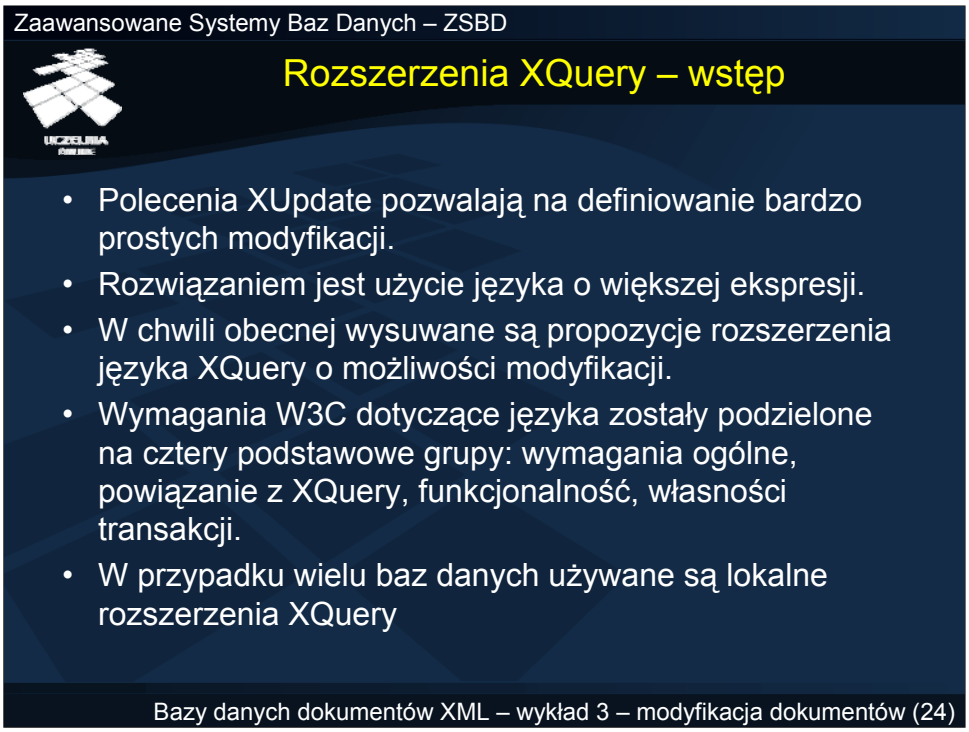
Kolejną wadą języka XUpdate jest brak możliwości stosowania zaawansowanych konstrukcji takich jak wyrażenia ilościowe lub możliwość zdefiniowania sekwencji zmian.

Dla przykładu, w języku XUpdate możemy zamienić miejscami dwa elementy np. placa_pod i nazwisko. Robimy to wykorzystując zmienne. Niestety operację taką możemy wykonać na rzecz jednego elementu pracownik. Nie ma możliwości zamiany tych elementów w wielu elementach pracownik za pomocą jednego polecenia. Musielibyśmy to wykonywać wielokrotnie, indywidualnie dla każdego pracownika!

Z analogicznych powodów nie możemy zdefiniować modyfikacji, która umieści atrybut ID w kolejnych elementach zespol, a wartość wstawianego atrybutu oprze o zawartość elementu id_zesp będącego elementem podrzędnym dla każdego z elementów zespol.

Innymi słowy, to do czego przyzwyczaił nas, dla przykładu, język SQL w relacyjnych systemach baz danych, po prostu nie jest osiągalne za pomocą języka XUpdate.

Kończąc opis języka XUpdate, należy zaznaczyć, że istnieje wiele lokalnych jego rozszerzeń, eliminujących powyższe wady. Przykładem może być tu wersja języka XUpdate wykorzystywana w ramach projektu Orbeon Presentation Serwer (http://www.orbeon.com/ops/doc/processors-xupdate).



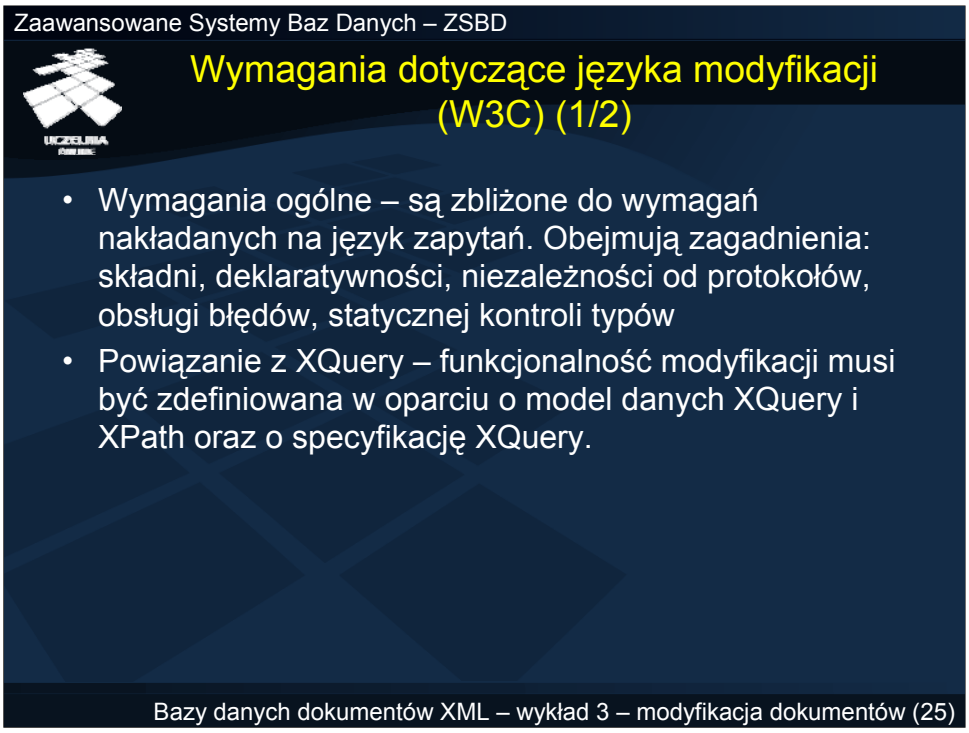
Polecenia XUpdate pozwalają na definiowanie bardzo prostych modyfikacji, co w wielu przypadkach może być dalece niewystarczające.

Rozwiązaniem jest użycie języka o większej ekspresji.

Obecnie przez W3C XQuery working group wysuwane są propozycje rozszerzenia języka XQuery o możliwości modyfikacji.

Analogicznie jak w przypadku XML:DB XUpdate Working Group, tak samo W3C XQuery Working Group zdefiniowało wymagania dotyczące języka modyfikacji dla baz danych dokumentów XML. Zostały one podzielone na cztery podstawowe grupy: wymagania ogólne, powiązanie z XQuery, funkcjonalność, własności transakcji.

W przypadku wielu bazach danych dokumentów XML już w chwili obecnej używane są lokalne rozszerzenia XQuery, przykładem może być tutaj Tamino firmy Software AG.



Tak jak wspomnieliśmy wcześniej wymagania dotyczące języka modyfikacji zostały podzielone na cztery podstawowe grupy: wymagania ogólne, powiązanie z XQuery, funkcjonalność, własności transakcji.

Wymagania ogólne – są zbliżone do wymagań nakładanych na język zapytań. Szczegółowe wymagania są następujące:

- wymagania dotyczące składni: może istnieć wiele postaci składni. Jedna ze składni musi być czytelna dla człowieka, jedna musi mieć format XML;

- język modyfikacji musi być deklaratywny;

- język musi być niezależny od wykorzystywanych protokołów;

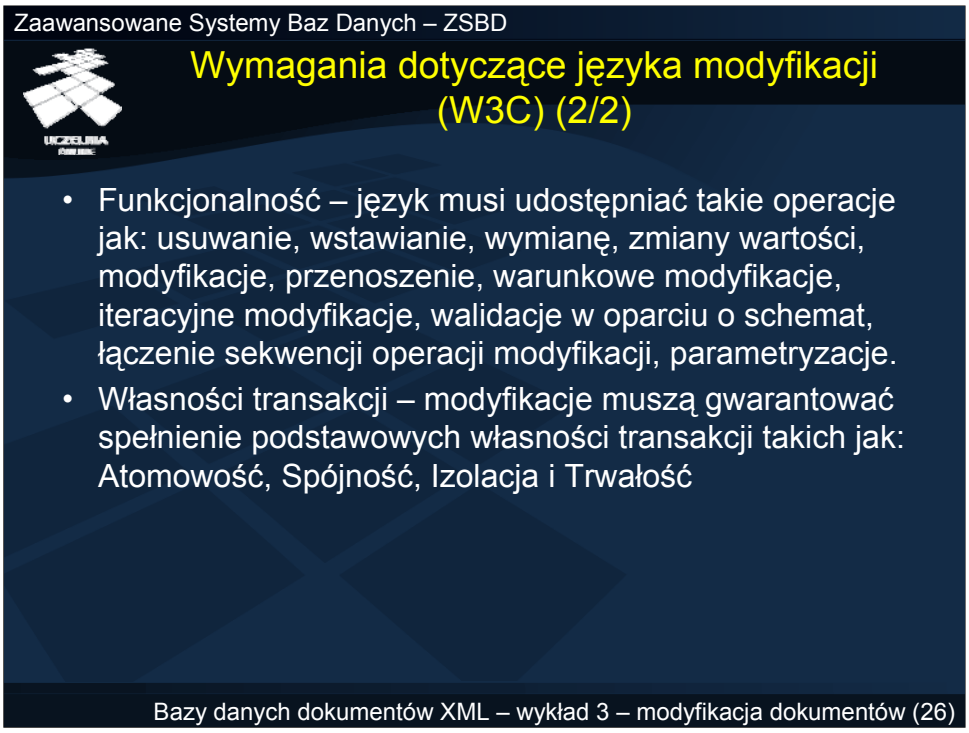
- musi mieć zdefiniowane standardowe warunki wystąpienia błędów podczas ewaluacji polecenia;

- język modyfikacji powinien udostępniać opcjonalną statyczną kontrolę typów.

Wymagania dotyczące powiązania z XQuery – dotyczą dwóch zagadnień: modelu danych oraz funkcjonalności XQuery. Szczegółowe wymagania są następujące:

- funkcjonalność języka modyfikacji musi być zdefiniowana w oparciu o model danych XQuery i XPath;

- własności języka modyfikacji muszą być oparte o specyfikację XQuery. Język modyfikacji musi wykorzystywać XQuery do wskazywania węzłów przeznaczonych do modyfikacji. Język modyfikacji musi wykorzystywać XQuery do wskazywania węzłów używanych przy modyfikacji.

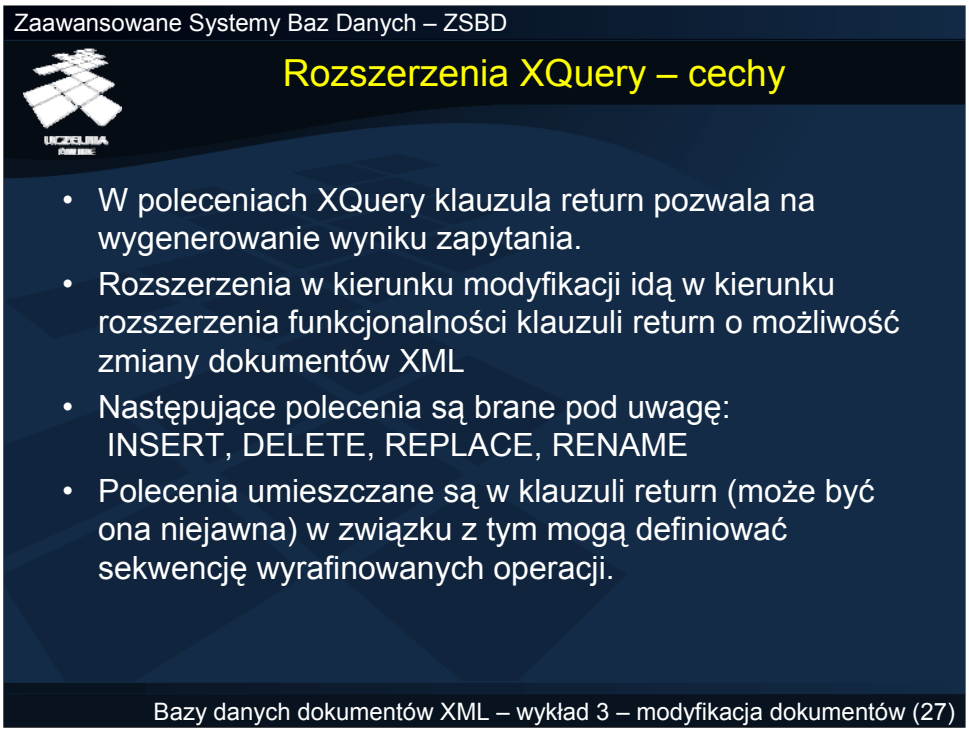


Wybrane wymagania dotyczące funkcjonalności języka modyfikacji są następujące:

- język modyfikacji musi dawać możliwość zmiany własności istniejących węzłów z zachowaniem ich tożsamości. Jednocześnie musi umożliwiać wykonanie nowej kopii węzła i wykonanie na nich odpowiednich zmian;
- język musi umożliwiać usuwanie węzłów;
- musi umożliwiać wstawianie nowych węzłów w ściśle określone miejsca;
- język musi dawać możliwość wymiany węzłów;
- powinien dawać możliwość zmiany takich własności węzła jak: nazwa, typ, zawartość itp.;
- może umożliwiać przenoszenie węzłów z jednej lokalizacji do innej;
- musi umożliwiać wykonywanie warunkowych modyfikacji;
- musi umożliwiać iterację po zbiorze węzłów w celu ich indywidualnej modyfikacji;
- może wspierać jawną walidację w oparciu o Schemat XML z zachowaniem tożsamości węzłów;
- język musi dawać możliwość łączenia operatorów modyfikacji z innymi operatorami modyfikacji;
- język modyfikacji powinien umożliwiać parametryzację operacji modyfikacji.

Ponadto W3C definiuje założenia dotyczące zachowywania podczas operacji wykonywanych za pomocą języka modyfikacji podstawowych własności transakcji takich jak Atomowość, Spójność, Izolacja i Trwałość.

Należy w tym momencie zaznaczyć, że obecna postać propozycji rozszerzeń języka XQuery o własności modyfikacji nie spełnia jeszcze wszystkich z wymienionych założeń.



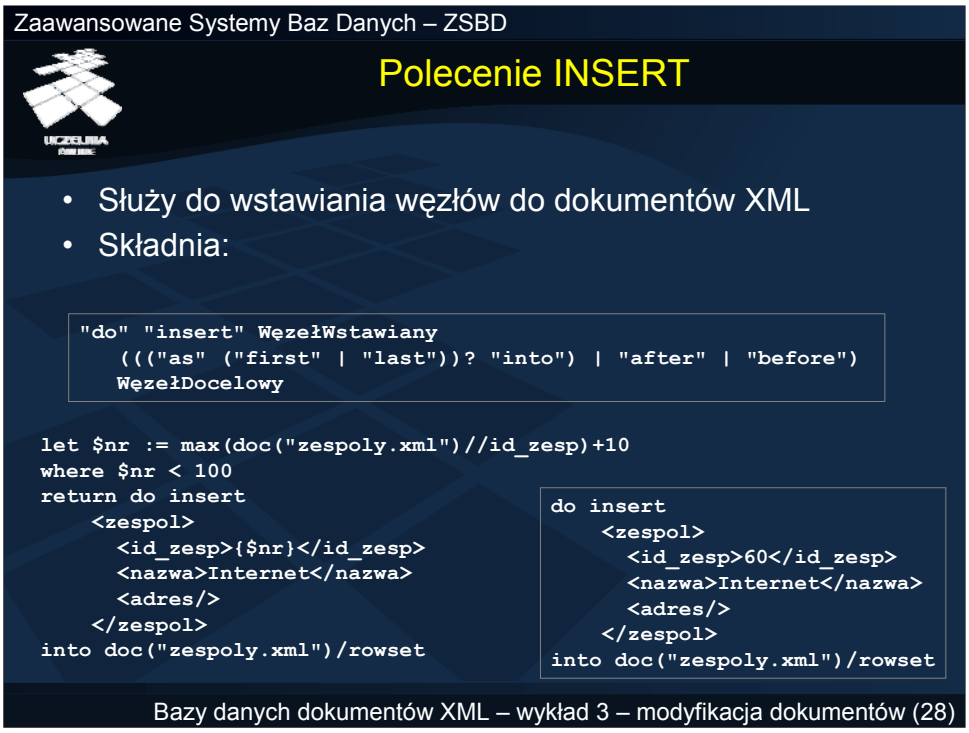
W poleceniach XQuery klauzula RETURN konstruuje wynik zapytania.

Rozszerzenia języka XQuery dotyczące modyfikacji idą w kierunku rozszerzenia funkcjonalności klauzuli RETURN o możliwość zmiany dokumentów XML.

W chwili obecnej następujące polecenia są brane pod uwagę: INSERT, DELETE, REPLACE, RENAME.

Polecenia te umieszczane są w klauzuli RETURN , przy czym klauzula RETURN może być niejawna (tak jak pozostałe klauzule zapytania XQuery). Dzięki temu, że polecenia modyfikacji umieszczane są w ramach klauzuli return i dotyczą węzłów zwracanych przez klauzulę return, za pomocą jednego polecenia możemy zdefiniować sekwencję wyrafinowanych modyfikacji.

Na następnych slajdach zostaną skrótowo omówione wspomniane polecenia.

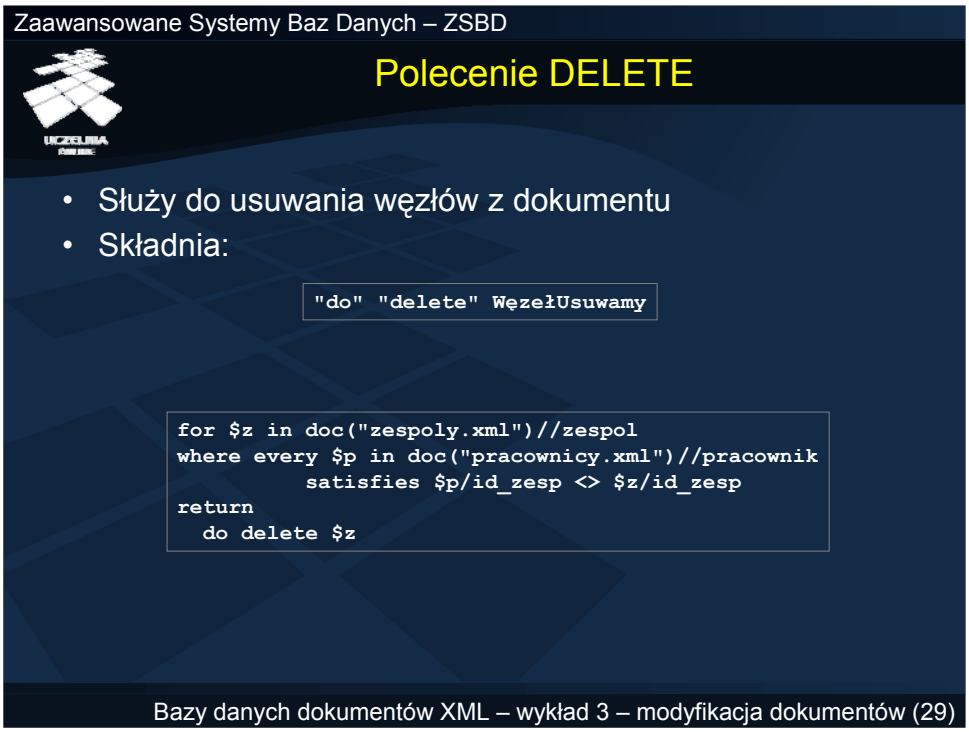


Polecenie INSERT służy do wstawiania węzłów do dokumentów XML. Jego składania została przedstawiona na slajdzie. W zależności od użytego słowa kluczowego (INTO, AFTER, BEFORE) umożliwia ono wstawienie nowych węzłów do wnętrza, bezpośrednio za lub bezpośrednio przed wskazywanym węzłem. Można zatem powiedzieć, że polecenie INSERT jest odpowiednikiem instrukcji: xu:insert-before, xu:insert-after i xu:append występujących we wnętrzu poleceń XUpdate.

Przykładowe polecenia wykorzystujące polecenie INSERT zostały przedstawione na slajdzie.

Pierwsze polecenie za pomocą klauzuli LET do zmiennej $nr przypisuje największą wartość elementu id_zesp zwiększoną o wartość 10. Następnie po sprawdzeniu czy uzyskana wartość nie jest większa niż 100 przystępujemy do modyfikacji. Modyfikacja polega na wstawieniu nowego węzła zespol, w którego wnętrzu zostanie wykorzystana wartość wcześniej utworzonej zmiennej $nr. Nowy węzeł zespol zostanie wstawiony do wnętrza elementu rowset.

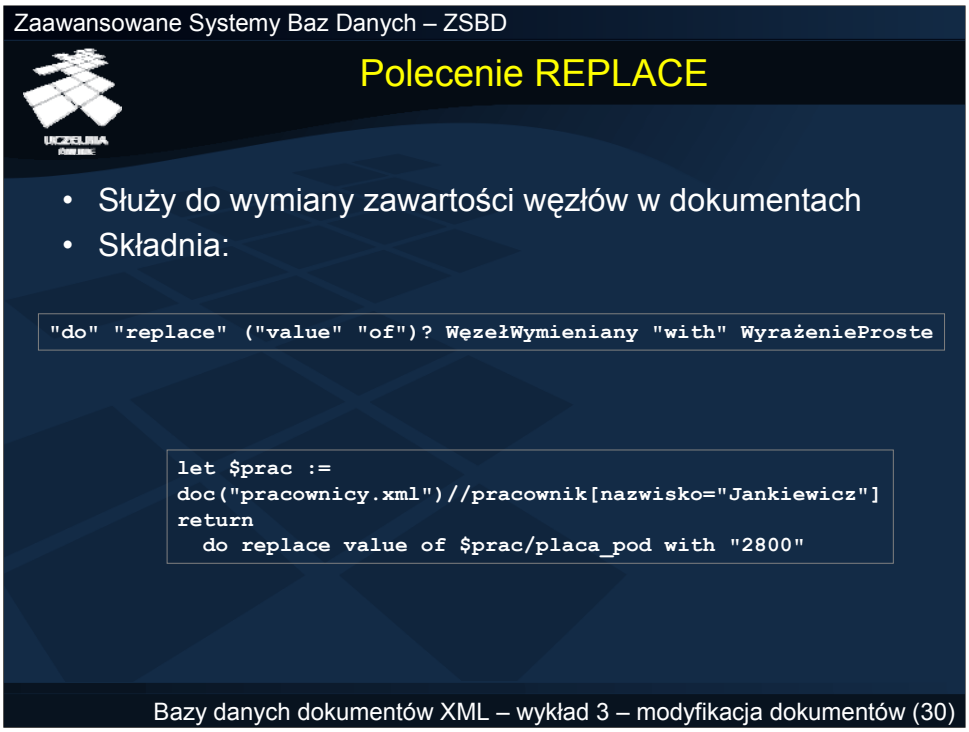
Drugi przykład to proste wstawienie złożonego elementu do dokumentu „zespoly.xml”. Zwróćmy uwagę na możliwość pominięcia klauzuli RETURN. Tego typu polecenie wykona się tylko jeden raz.



Polecenie DELETE służy do usuwania węzłów z dokumentu. Składnia tego polecenia została przedstawiona na slajdzie. Można powiedzieć, że polecenie DELETE jest odpowiednikiem xu:remove występującego w poleceniach XUpdate.

Na slajdzie przedstawiono przykładowe polecenie, które usunie zespoły, nie posiadające żadnych pracowników.

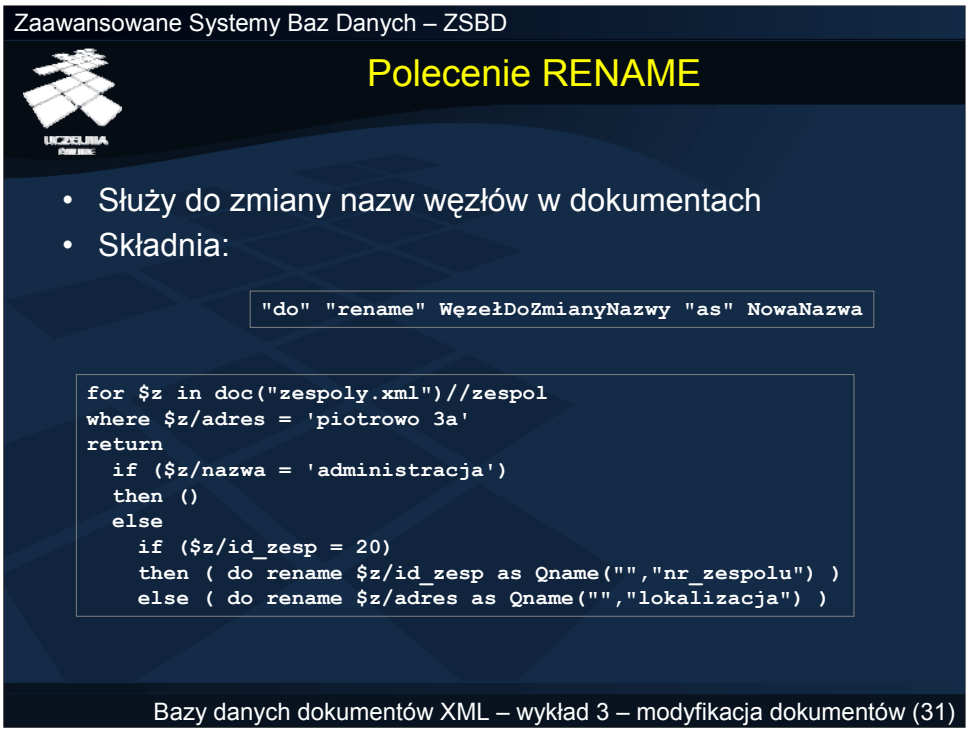
Polecenie DELETE zostanie wykonane tyle razy, ile krotek zostanie wygenerowanych przez konstrukcję FLWOR



Polecenie REPLACE służy do wymiany zawartości węzłów w dokumentach. Składnia polecenia REPLACE została zaprezentowana na slajdzie.

Podobnie jak w przypadku XUpdate zmiana może dotyczyć tylko węzłów prostych.

Na slajdzie przedstawiony przykład zmieni płacę podstawową pracownikowi o nazwisku Jankiewicz.



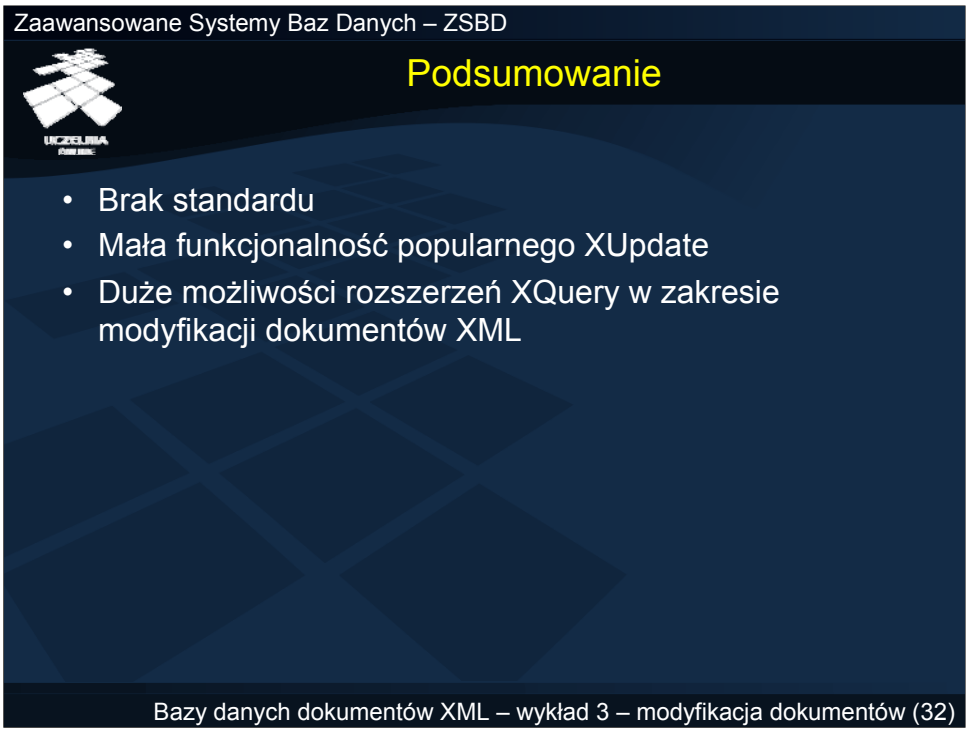
Ostatnim poleceniem, które omówimy jest polecenie RENAME. Służy ono do zmiany nazw węzłów w dokumentach XML, a jego składnia, tak samo jak w przypadku poprzednio omawianych poleceń została przedstawiona na slajdzie.

Przykład z wykorzystaniem polecenia RENAME jest trochę bardziej złożony

Za pomocą pętli FOR do zmiennej $z będą przypisywane kolejno elementy zespol. Dla każdego takiego przypisania zostanie wykonane polecenie RETURN z umieszczonymi w nim instrukcjami.

W przykładzie zastosowane zostały możliwości XQuery w zakresie stosowania instrukcji warunkowych. Dzięki temu możliwa była następująca modyfikacja.

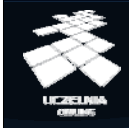
Jeżeli $z będzie przypisany do zespołu o nazwie administracja, nie zostanie wykonana żadna zmiana. W przeciwnym przypadku zostanie przeprowadzony test dotyczący wartości elementu id_zesp zespołu wskazywanego przez zmienną $z. Jeżeli id_zesp będzie miał wartość równą 20 wówczas element id_zesp zostanie zmieniony na nr_zespolu. Jeśli wartość id_zesp będzie inna wówczas zmianie nazwy będzie poddany element adres.



Podsumowując, brak na chwilę obecną standardu języka modyfikacji dla baz danych dokumentów XML. Stosunkowo popularne i powszechnie stosowane rozwiązanie oparte na języku XUpdate jest bardzo ograniczone.

Rozszerzenie języka XQuery o elementy modyfikacji daje duże możliwości w zakresie modyfikacji dokumentów XML, dlatego też coraz częściej jest ono implementowane w bazach danych dokumentów XML. Przykładem może być baza danych eXists, która pozwala na modyfikacje swojej zawartości przy wykorzystaniu zarówno języka XUpdate jak i XQuery.

Mimo, iż rozszerzenie języka XQuery o możliwości modyfikacji daje większą funkcjonalność niż XUpdate, należy zauważyć, że trzon poleceń pozostał ten sam. Zmianie uległ przede wszystkim sposób wskazywania węzłów podlegających zmianie (w XUpdate jest to XPath) oraz sposób użycia zmiennych (w XQuery sposób użycia zmiennych daje znacznie bardziej bogate możliwości).



Bilbliografia

- XUpdate
  - Specyfikacja
    http://xmldb-org.sourceforge.net/xupdate/index.html
  - X-Hive XUpdate demo
    www.x-hive.com/xupdate/
- Rozszerzenia XQuery
  - Funkcjonalność modyfikacji (working draft)
    http://www.w3.org/TR/xqupdate/
  - Przykłady użycia
    http://www.w3.org/TR/xqupdateusecases/