

Designer 6i – Nowe Cechy

Najnowsza wersja narzędzia Designer firmy Oracle, Designer 6i, posiada szereg rozszerzeń w stosunku do poprzednich wersji. Najważniejsze zmiany dotyczą repozytorium. Wreszcie możliwa jest kontrola wersji obiektów i przechowywanie w repozytorium zarówno modelu analitycznego i logicznego aplikacji jak i tekstowych lub binarnych plików. Także generatory posiadają szereg ulepszeń, które przyspieszają tworzenie aplikacji oraz pozwalają na generację obiektów Oracle 8i, także kodu w języku Java. Najmniej zmian wprowadzono w narzędziach wykorzystywanych na etapie analizy.

Poniżej zostaną pokrótce omówione najistotniejsze rozszerzenia Designera 6i w stosunku do poprzednich wersji. Dokładne opisanie wszystkich zmian i ulepszeń wymagałoby napisania przeze mnie średniej objętości książki. Bardzo intensywny kurs dotyczący wyłącznie nowych cech Designera w wersji 6i prowadzony przeze mnie w Centrum Edukacyjnym Oracle Polska trwa 3 dni a podręcznik kursowy liczy kilkaset stron.

Nowa struktura repozytorium

Rosnąca złożoność projektów informatycznych i konieczność równoczesnego utrzymania więcej niż jednej wersji systemu oraz wzrost liczebności zespołów tworzących te systemy spowodowały, że dotychczasowa struktura repozytorium narzędzia Designer nie spełniała wymagań analityków, projektantów, programistów oraz szefów projektów. Nowa wersja narzędzia to przede wszystkim całkowicie zmieniona struktura repozytorium. Najważniejsze nowe cechy repozytorium Designera to możliwość kontroli wersji obiektów oraz możliwość przechowywania w repozytorium plików i zarządzania z poziomu Designera plikami oraz podkartotekami.

Typy obiektów

W poprzednich wersjach Designera cała dokumentacja techniczna systemu, skrypty wykorzystywane podczas testów, ikony niezbędne do uruchomienia formatek i inne pliki wykorzystywane w procesie tworzenia, wdrażania i utrzymania systemu musiały być przechowywane poza repozytorium – w systemie plików.

- W przypadku Designera 6i obiekty przechowywane w repozytorium dzielą się na dwie grupy:
- obiekty strukturalne (znane z poprzednich wersji Designera - definicje encje, funkcji, modele logiczne tabel, modułów i inne)
 - nie spotykane w poprzednich wersjach - obiekty niestukturalne (tekstowe i binarne pliki)

Designer 6i umożliwia:

- przechowywanie w repozytorium plików,
- uruchamianie i przeglądanie tych plików z poziomu Designera,
- określanie, w przypadku plików o nietypowych dla standardowej funkcjonalności Designera rozszerzeniach, które pliki są traktowane jako tekstowe a które jako binarne,
- synchronizację podkartotek systemu plików i repozytorium,
- **kontrolowanie wersji plików, porównywanie wersji plików i ich łączenie** (ostatnie dwie operacje nie są możliwe w przypadku plików w formacie ASCII i UNICODE oraz plików formularzowych)

Zatem – od wersji 6i można:

- porównywać ze sobą dwie formatki (wreszcie łatwo można otrzymać odpowiedź na pytanie: ‘Co zmieniłem(am) w formatce, że działa niezgodnie z oczekiwaniami/lepiej niż poprzednio? Jaki fragment kodu? Jaką właściwość, którego z elementów formularza?’)

- wyeksportować do pliku wszystkie elementy składające się na system lub jego część – nie tylko definicje modułów ale też gotowe formatki – i zaimportować zawartość tego pliku do innego repozytorium

Kontenery i przestrzenie robocze

W poprzednich wersjach Designera obiekty repozytorium były umieszczane w *systemach aplikacyjnych*. System aplikacyjny miał swojego właściciela, inni użytkownicy repozytorium mogli mieć nadane prawa umożliwiające im wykonywanie określonych operacji na systemie aplikacyjnym. Istniała możliwość tworzenia hierarchii systemów aplikacyjnych oraz kopiowania i współdzielenia obiektów. Praca nad systemem zawsze odbywała się w kontekście jednego systemu aplikacyjnego ewentualnie ‘rozszerzonego’ o obiekty współdzielone. Żeby wiele osób mogło bezpiecznie pracować nad wyodrębnionymi ale mającymi wspólne fragmenty (na przykład definicje encji, tabel lub dziedzin) fragmentami jednego systemu należało opracować zasady współdzielenia obiektów lub ich kopiowania. Problemem było wyeksportowanie z repozytorium do pliku fragmentu aplikacji zawierającej na przykład wyłącznie model logiczny bazy danych.

Wyraźnie brakowało mechanizmu przypominającego bazodanową perspektywę – spojrzenia na repozytorium jak na zestaw ‘tabel’ zawierających ‘wiersze’, które można by widzieć przez ‘perspektywę’, oczywiście perspektywy z klauzulą ‘FROM’ pozwalająca na wybór z wielu ‘tabel’.

W wersji 6i istnieje mechanizm ‘perspektyw’ repozytorium. Posługując się analogią:

- wiersze to obiekty strukturalne i niestukturalne,
- tabele to kontenery,
- perspektywy to przestrzenie robocze,
- klauzule FROM można określać samemu wskazując odpowiednie obiekty lub stosując reguły przestrzeni roboczych i konfiguracje.

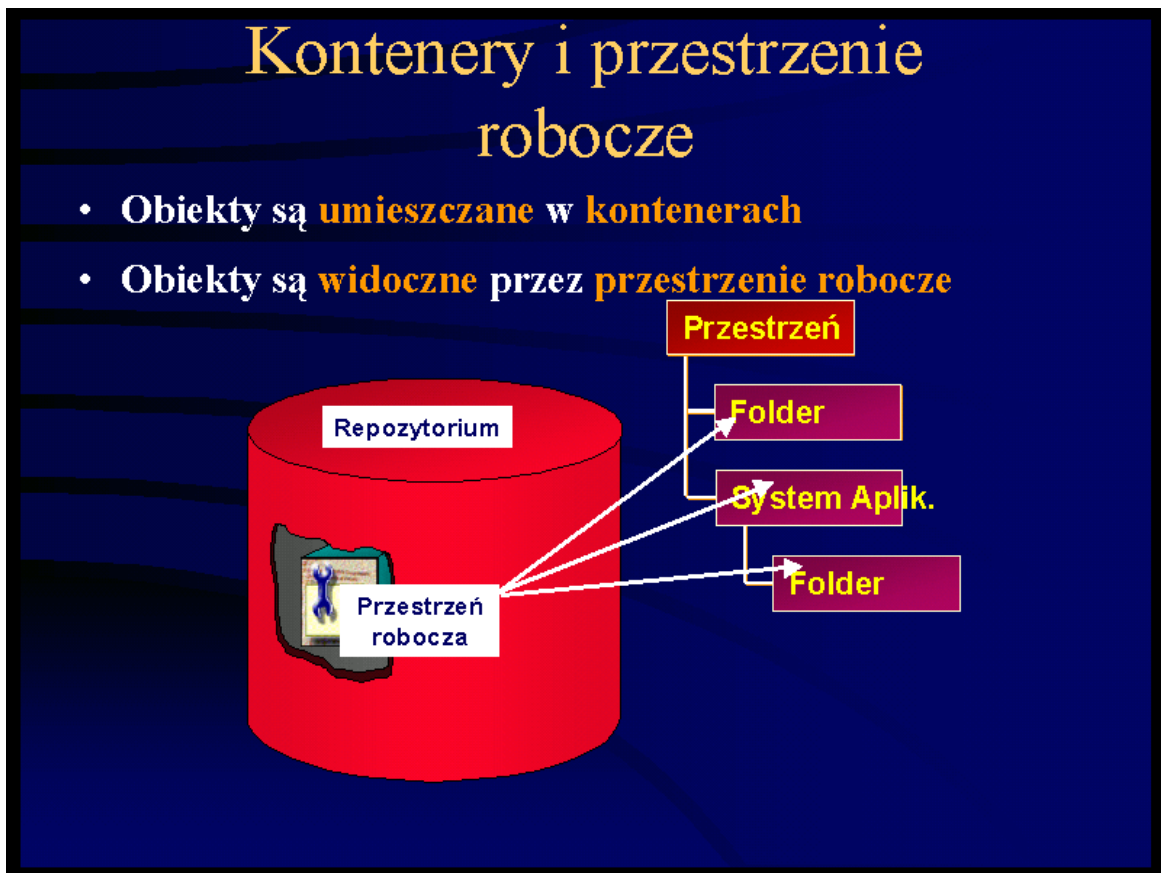
W wersji 6i narzędzia Designer obiekty (encje, funkcje, definicje tabel, pliki) są umieszczone w *kontenerach*. Mówimy, że obiekty *należą do kontenerów* (jak wiersze w bazodanowej tabeli należą do jednej tabeli). Zawartość kontenerów jest *widoczna* poprzez *przestrzenie robocze*. Przestrzenie robocze niczego nie zawierają (jak perspektywy w bazie danych). W przeciwieństwie do tabel i perspektyw – obiekt repozytorium musi być widoczny przez przestrzeń, by można było wykonywać na nim jakiegokolwiek operacje.

Są dwa typy kontenerów:

- systemy aplikacyjne
- foldery

Pierwsze przeznaczone są zwykle do przechowywania obiektów strukturalnych (encje, definicje tabel itp.), drugie do przechowywania plików.

Pomiędzy kontenerami a przestrzeniami roboczymi istnieje związek wiele-do-wielu. Obiekt umieszczony w jednym kontenerze może być widoczny przez wiele przestrzeni roboczych. Przez przestrzeń roboczą mogą być widoczne obiekty umieszczone w więcej niż jednym kontenerze. Pojęcie współdzielenia obiektów właściwie straciło rację bytu, zastąpiło je pojęcia *włączania obiektu* do przestrzeni roboczej.

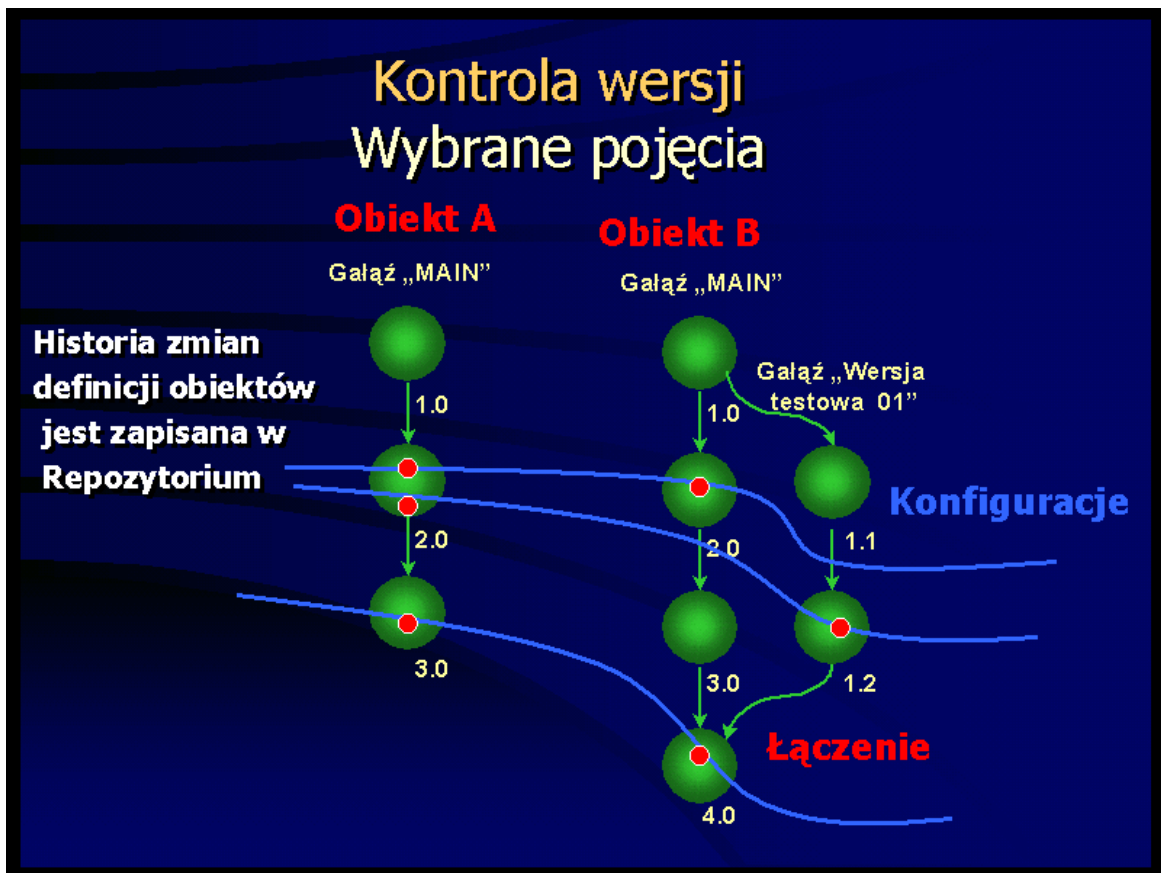


Kontrola wersji obiektów

Przechowywana w repozytorium Designera definicja obiektu (strukturalnego i niestrukturalnego) może i zwykle podlega zmianom. W przypadku Designera 6i historia zmian definicji obiektu może być przechowywana w repozytorium. Należy przy tym podkreślić, że historia wersji obiektu jest przechowywana w repozytorium w sposób 'przyrostowy'. Oznacza to, że przestrzeń zajmowana przez dwie wersje obiektu jest mniejsza niż przestrzeń zajmowana przez jedną wersję obiektu pomnożoną przez dwa.

Tworzenie nowych wersji obiektów z zachowaniem poprzedniej wersji umożliwia:

- uzyskanie odpowiedzi na pytanie typu 'co będzie, jeśli wprowadzę zmiany w definicji tabeli?' (kontrola wpływu zmian)
- porównanie dwóch wersji (aplikacji, formularza)
- równoległa pracę nad wieloma gałęziami aplikacji (np. wersja produkcyjna i testowa)
- powrót do poprzedniej wersji



Operacje check-in i check-out

Nowo utworzony obiekt jest ‘jednowymiarowy’. Jest widoczny tylko przez jedną przestrzeń roboczą, zmieniona definicja obiektu zamazuje poprzednią definicję.

Pierwsza operacja *check-in* wykonana na obiekcie oznacza, że od tej pory obiekt podlega kontroli wersji. Pierwsza wersja obiektu ma oznaczenie 1.0 (o ile zasady etykietowania wersji obiektów nie zostaną zmienione) i jest wersją tylko do odczytu. Może być widoczna przez wiele przestrzeni roboczych.

Jeśli definicja obiektu ma podlegać modyfikacjom, na obiekcie musi być wykonana operacja *check-out*. Operacja ta tworzy *prywatną kopię* obiektu. Wiele osób może równocześnie modyfikować ten sam obiekt – efekty tych zmian podlegają operacji *łączenia*.

Kolejny krok to operacja *check-in* (jeśli wprowadzone zmiany mają być zatwierdzone) lub wycofanie zmian i powrót do definicji obiektu sprzed operacji *check-out*. Wykonanie operacji *check-in* tworzy nową wersję obiektu. Inni użytkownicy repozytorium mogą widzieć zmiany, nowa wersja może być widoczna przez przestrzenie robocze. Poprzednia wersja nadal jest w repozytorium.

Rozgałęzienia

Rozgałęzienia

- Operacja *check-in* do gałęzi 'TEST_001'

The image illustrates the process of branching in Oracle Designer. It consists of three sequential screenshots of the Repository Object Navigator window:

- Initial State:** The window shows the 'MAIN' branch with three versions: 1.0 (blue), 1.1 (blue), and 1.2 (red). The text 'wersja 0/0' is visible below version 1.0, and 'dodanie min. 1 s...' is below version 1.1.
- Branching:** Version 1.1 is highlighted with a blue circle and a blue arrow, indicating it is being checked out to a new branch.
- Result:** A new branch named 'TEST_001' is created, containing version 1.1.0 (red) of the object. The original version 1.1 remains in the 'MAIN' branch.

At the bottom of the screenshots, the following text is visible: '17-maj-01', 'Designer 6i - Nowe Cechy', and '37'.

Rozgałęzienia dają możliwość równoległej pracy nad kilkoma wersjami aplikacji. Dzięki temu możliwe jest na przykład utrzymywanie dwóch wersji systemu – produkcyjnej i testowej. Przestrzenie robocze mogą obejmować wersje obiektów należące do różnych gałęzi.

Konfiguracje

Konfiguracja - to zbiór określonych wersji wybranych obiektów repozytorium. Przykładowo, konfiguracja może obejmować:

- wersje obiektów tworzące wspólnie pewną wersję aplikacji,
- wersje obiektów tworzące uzupełnienie pewnej wersji aplikacji

Obiekty dołączamy do konfiguracji 'ręcznie' wskazując która wersja którego obiektu ma należeć do konfiguracji lub dołączając do konfiguracji obiekty z pewnej przestrzeni i/lub gałęzi.

Konfiguracje podlegają kontroli wersji, mogą być w całości dołączane do przestrzeni roboczej i eksportowane do plików.

Operacje porównania i łączenia

Dwie wersje tego samego obiektu lub dwóch różnych obiektów tego samego typu mogą być porównywane i łączone. Operacje te dotyczą obiektów strukturalnych oraz plików w formacie ASCII, UNICODE i plików formularzowych.

Jak wspomniano wyżej, możliwość wykonania operacji porównania jest bardzo przydatna w przypadku plików formularzowych. Dzięki temu możemy znaleźć różnice zarówno w kodzie PL/SQL jak i we właściwościach poszczególnych elementów formularza.

Operacja łączenia:

- zachodzi zawsze, gdy dwóch użytkowników wykona operację *check-out* dla tej samej wersji obiektu (osoba wykonująca operację *check-in* jako druga musi wykonać łączenie),
- może być wykonana na żądanie, np. w celu połączenia pracy dwóch zespołów

Operacja łączenia wykorzystuje zawsze 3 wersje obiektu:

- **źródło** - zostanie utworzona nowa wersja tej wersji (z uwzględnionymi zmianami)
- **współuczestnik** - wersja dostarczająca zmian do źródła
- **wspólny poprzednik** - określany automatycznie po wybraniu źródła i współuczestnika

Zasady wykonywania operacji łączenia są następujące:

- **Elementy dodane** (w stosunku do wspólnego poprzednika) w którejś z wersji łączonych są dodawane do wyniku
- **Elementy usunięte** (w stosunku do wspólnego poprzednika) w którejś z wersji łączonych nie są dodawane do wyniku
- **Zmiany** (w stosunku do wspólnego poprzednika) wprowadzone tylko w **jednej z wersji** łączonych są uwzględniane w wyniku
- **Zmiany** (w stosunku do wspólnego poprzednika) wprowadzone w **obydwu łączonych wersjach** powodują konflikt, który użytkownik musi rozstrzygnąć (którą zmianę należy uwzględnić, czy odrzucić obydwie zmiany)

Operacja łączenia może odbywać się automatycznie lub wynik porównania może być wyświetlany na ekranie, co pozwala twórcy aplikacji podjąć decyzję, którą zmianę należy uwzględnić i jak rozwiązać konflikty. Jeśli łączone obiekty leżą na różnych gałęziach jest możliwość określenia w której z gałęzi znajdzie się obiekt wynikowy.

Generator serwera

Najważniejsze nowe cechy generatora bazy danych obejmują:

- Możliwość tworzenia dynamicznych dziedzin wykorzystujących tabele i więzy spójności dziedzin
- Modelowanie i generację obiektów bazy danych w języku Java
- Modelowanie i generację obiektów Oracle8i i Oracle8 (tabele zorganizowane indeksowo, indeksy oparte na wyrażeniach, opcja *compute statistics* dla indeksów, zmaterializowane perspektywy i migawki - możliwość generacji poleceń DDL zgodnych z wersją Oracle 8 lub 8i).

Dynamiczne dziedziny w Designer 6i

Implementacja dziedzin w przypadku wersji 6i Designera może być wykonywana poprzez:

- Statyczne więzy spójności CHECK
- Tabelę CG_REF_CODES (dynamiczne dziedziny 'starego' typu)
- Tabele i więzy spójności dziedzin (dynamiczne dziedziny 'nowego' typu)

Dynamiczne dziedziny 'nowego' typu są stosowane wtedy, gdy w bazie danych jest już tabela 'słownikowa' wypełniona danymi, na przykład lista miast, kodów, skrótów itp. Poprzednie wersje Designera wymagały, by 'słownik' był umieszczany w tabeli CG_REF_CODES tworzonej na podstawie definicji dziedzin zapisanych w repozytorium.

W wersji 6i tabela wypełniana danymi musi być ‘zarejestrowana’ w repozytorium jako tzw. *Domain Table*. Twórca aplikacji określa, która kolumna w jego tabeli przechowuje wartość, która skrót i znaczenie (tylko wartość jest obowiązkowa). W jednej tabeli może być wiele dziedzin.

Następnie należy utworzyć obiekt repozytorium typu *Domain Key Constraint*. Obiekt ten jest obsługiwany albo przez kod dodawany do aplikacji formularzowej albo kod umieszczony w API tabeli – nie jest tworzony bazodanowy więz spójności typu klucz obcy.

Designer 6i i Java

W bazie danych Oracle8i mogą być przechowywane:

- skompilowane klasy Java (ang. Java class schema objects),
- kod źródłowy Java (ang. Java source schema objects),
- zasoby (ang. Java resource schema objects)

Serwer Oracle8i zawiera wirtualną maszynę Java - Aurora JVM. Aurora JVM ładuje i wykonuje obiekty bazy danych w języku Java. Aby Aurora JVM mogła ‘wykonać’ klasę (aplikację lub metodę klasy) w języku Java należy załadować skompilowaną wersję klasy do schematu bazy danych

Jeśli składowany podprogram w języku Java ma być wywoływany z SQL lub PL/SQL należy upublicznić ten podprogram. Operacje ładowania i upubliczniania to dwa odrębne kroki - jeśli metoda klasy ma być wywoływana tylko z kodu Java, nie trzeba jej upubliczniać (ale trzeba załadować klasę)

Są dwa sposoby ładowanie klasy do schematu bazy danych:

1. Zapisanie **kodu źródłowego** w bazie danych (bezpośrednio lub z pliku zewnętrznego), z równoczesnym skompilowaniem i załadowaniem klasy do schematu bazy danych (polecenie **CREATE JAVA SOURCE ...** lub narzędzie *loadjava*)
2. Utworzenie skompilowanej klasy (*.class) w pliku zewnętrznym (na przykład z wykorzystaniem narzędzia JDeveloper) i załadowanie tej **skompilowanej klasy** do schematu bazy danych (**CREATE JAVA CLASS ...** lub narzędzie *loadjava*).

Upublicznienie metody z klasy Java odbywa się poprzez wykonanie polecenia **CREATE OR REPLACE procedure ... AS ... LANGUAGE JAVA NAME**

Designer 6i zawiera następującą nową funkcjonalność dotyczącą języka Java:

- Edytor Logiki obsługuje zarówno PL/SQL jak i język Java,
- w repozytorium może być przechowywany kod źródłowy języka Java; kod ten wprowadza się albo poprzez edytor logiki albo z pliku zewnętrznego,
- istnieje możliwość stworzenia definicji obiektu typu Class Definition, jednak skompilowane klasy w języku Java nie są wprowadzane do repozytorium; Design Editor umożliwia generację polecenia **CREATE JAVA CLASS USING BFILE ...**, którego wykonanie ładuje klasę z pliku zewnętrznego do bazy danych,
- upublicznianie metod Java odbywa się poprzez stworzenie definicji podprogramu PL/SQL i powiązanie tego podprogramu z określoną metodą klasy Java

Generator aplikacji formularzowych

Nowa funkcjonalność generatora aplikacji formularzowych zdecydowanie skraca proces tworzenia formatek. Rozszerzenia w wersji 6i dotyczą między innymi:

- Zupełnie nowego potraktowania obiektów typu lista wartości

- Możliwości generacji w krótkim czasie formularzy o zdecydowanie ładniejszym układzie (bloki wieloregionowe, bloki złożone z wielu komponentów, bloki obok siebie, formularze typu nawigator (drzewka), względne tabulatory)
- Rozszerzenia funkcjonalności bibliotek obiektów (możliwość definiowania obiektów poziomego modułu, takich jak alerty, bloki, okna, grupy rekordów dodawanych do wygenerowanego formularza bez potrzeby użycia szablonu; możliwość definiowania listy bibliotek wykorzystywanych podczas generacji formularza).

Listy wartości (LOVs)

W poprzednich wersjach Designera 6i aby blok formularzowy utworzony z komponentu modułu zawierał listę wartości w komponencie modułu, z którego powstał ten blok musiało istnieć użycie podglądane.

Lista wartości:

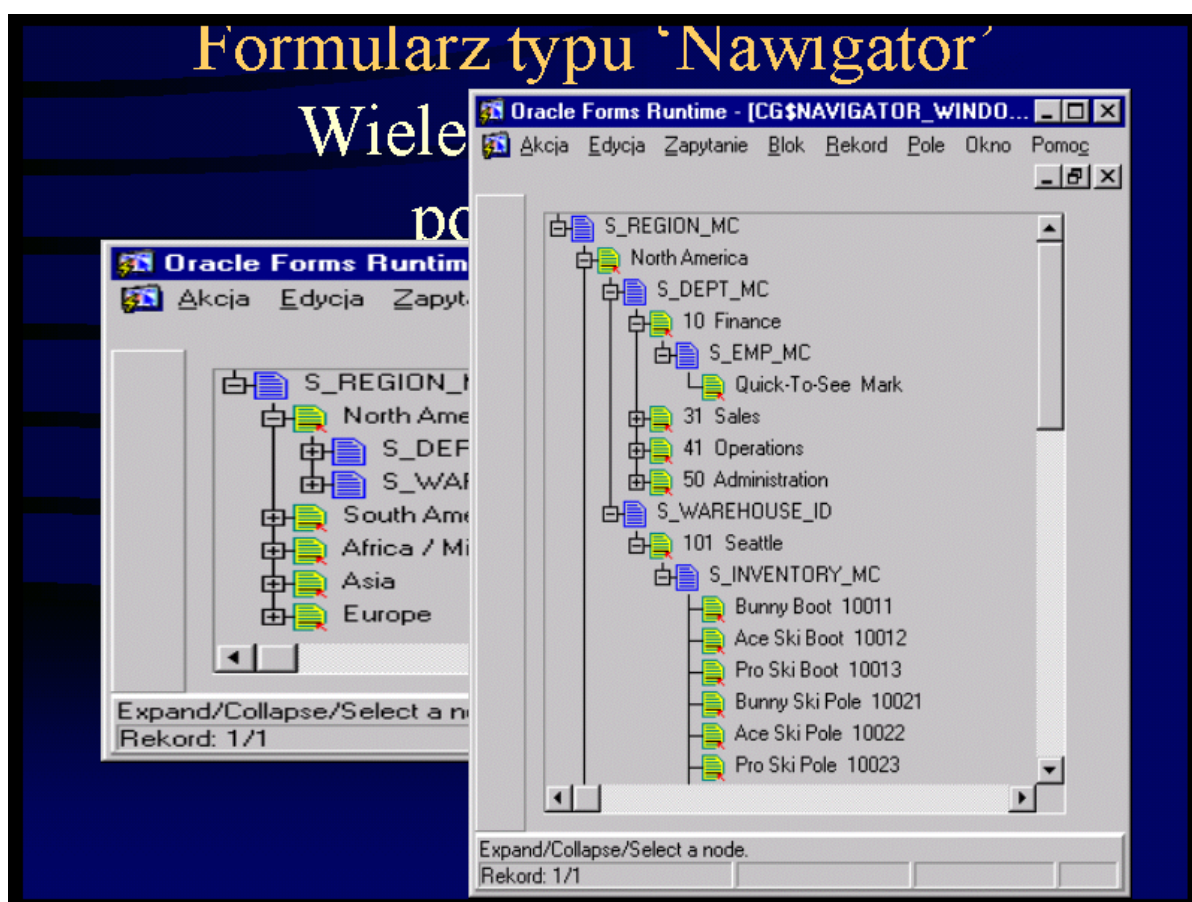
- nie mogła zawierać elementów niezwiązanych,
- nie mogła być powiązana z elementem niezwiązanym,
- była 'jednokrotnego użycia' (nie można było zdefiniować listy wartości na poziomie aplikacji),
- wymagała dwóch klauzul WHERE na poziomie użycia podglądanego.

W narzędziu Designer 6i wreszcie lista wartości jest obiektem, który można modelować niezależnie od modułu i/lub komponentów modułu. Istnieje możliwość definiowania list wartości wielokrotnego użycia, które podobnie jak komponenty modułów wielokrotnego użycia mogą być używane w wielu modułach.

Lista wartości może zawierać elementy niezwiązane i może być związana z elementem niezwiązanym. Klauzule WHERE definiuje się na poziomie listy wartości a nie na poziomie użycia tabeli.

Wygląd formularzy

Nie jest tajemnicą, że wielu twórców aplikacji używa Designera do generacji plików .fmb w wersji 'prototypowej'. Dopracowywanie wyglądu formatek odbywa się z wykorzystaniem narzędzia Developer. Umiejętnie wykorzystane nowe możliwości Designera 6i dotyczące określania wyglądu generowanych formatek oznaczają przede wszystkim skrócenie czasu tworzenia aplikacji. Efekty, które do tej pory można było osiągnąć 'rzeźbiąc' formularz z wykorzystaniem Developera teraz mogą być uzyskano poprzez ustawienie jednej do kilku właściwości lub preferencji generatora.



Przykładowo, jeśli formularz zawiera komponenty nadrzędne z którymi są związane poprzez powiązania (*key-based links*) komponenty podrzędne, ustawienie wartości *Naviagtor* dla właściwości *Layout format* wystarcza, by wygenerowany formularz miał wygląd jak na rysunku powyżej. Po wybraniu dowolnego węzła jest wyświetlany rekord bazy danych odpowiadający temu węzłowi.

Generator aplikacji Web PL/SQL

Nowe cechy generatora aplikacji Web PL/SQL to możliwość dynamicznego tworzenia dokumentów HTML zawierających:

- wielowierszowe formularze umożliwiające wykonywanie operacji DML,
- grupy oraz podsumowania (funkcjonalność zbliżona do grup łamiących oraz pól sumarycznych w raportach),
- pola akcji (nawigacyjne oraz związane ze zdarzeniami JavaScript),
- kalendarz związany z każdym polem typu data wyświetlany w dodatkowym oknie JavaScript,
- listy wartości modelowane identycznie jak w przypadku aplikacji formularzowych, wyświetlane w odrębnym oknie JavaScript lub w odrębnej ramce HTML.

Ponadto rozbudowano możliwości ochrony dostępu do danych z poziomu aplikacji Web PL/SQL. Dane o klientach mających dostęp do aplikacji mogą być przechowywane w tabelach

bazy danych. Dane identyfikacyjne są podawane przez klienta podczas pierwszego wywołania określonego modułu (*self-service security*) lub są określone przez administratora systemu (*authorized security*). Dostarczone wraz z Designerem 6i skrypty `wsgsec.sql` (*authorized security*) i `wsgsr.sql` (*self-service security*) generują niezbędne obiekty bazy danych (tabele oraz pakiety).

Podsumowanie

Czy Designer 6i może spełniać wymagania twórców dużych systemów informatycznych? Czy umożliwia 100-procentową generację kodu? Tak, ale dopiero wtedy, gdy możliwości tego narzędzia zostaną dobrze poznane i zrozumiane.

I zawsze należy pamiętać o tym, że samo narzędzie CASE nie gwarantuje sukcesu projektu. Konieczna jest jeszcze metodyka i technika. Tym bardziej w przypadku wersji 6i, która wreszcie ma rozszerzenia specjalnie dedykowane kierownikom projektów.