

KBD2 — Komercyjne bazy danych

dr inż. Tomasz Traczyk

6. Wprowadzenie do Oracle Forms

Grudzień 2006

Copyright © Tomasz Traczyk
Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej
Materiały dydaktyczne przeznaczone są wyłącznie do indywidualnego użytku studiujących.
Rozpowszechnianie kopii bez pisemnej zgody autora jest zabronione.

Streszczenie

Wykład podaje podstawowe informacje o narzędziu Oracle Forms – tradycyjnym narzędziu 4GL do tworzenia aplikacji formularzowych.

Oracle Forms i Reports – cechy podstawowe

Przeznaczenie narzędzi

- Profesjonalne tworzenie aplikacji
- Rynek korporacyjny

Architektury

- wersja 6i – praca w środowiskach GUI, znakowych, w architekturze terminalowej, klient-serwer i trójwarstwowej
- od wersji 9i – Forms tylko w architekturze trójwarstwowej, Reports także w klient-serwer

Otwartość środowiska

- Współpraca z b.d. innych producentów i źródłami innymi niż bazy danych
- Możliwość oprogramowania nietypowej obsługi (np. wyzwalacze transakcyjne)
- Możliwe dołączanie *foreign functions* w innych językach
- Możliwe wykorzystanie Java Beans

Cechy użytkowe

- Dobra integracja z SZRBD Oracle — dobra wydajność aplikacji
- Niezła wydajność pracy programisty dla prostych aplikacji
- Dość łatwa integracja aplikacji (menu, formularzy, raportów, grafiki)
- Jednolite programowanie w języku PL/SQL
- NLS — pełna obsługa języków narodowych (także nieeuropejskich)
- Współpraca z systemami zarządzania wersjami
- Niezła integracja z pakietem CASE: Oracle Designer

Oracle Forms

Przeznaczenie

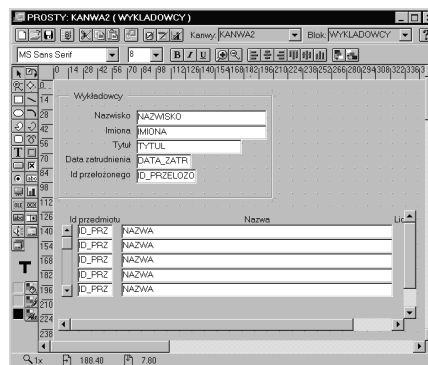
- Tworzenie ekranów do wprowadzania i przeglądania danych
- Tworzenie menu aplikacji

Moduły aplikacji

- Formularze
 - postać źródłowa: `.fmb`
 - program skompilowany (wykonywany przez *runtime*): `.fmx`
- Menu
 - postać źródłowa: `.mmb`
 - moduł skompilowany (wykonywany tylko razem z formularzem): `.mmx`
- Biblioteki PL/SQL
 - postać źródłowa + skompilowana: `.p11`
 - moduł skompilowany: `.p1x`
- Biblioteki obiektów
 - postać źródłowa `.o1b`

Narzędzia projektowe (*Forms Builder*)

- Nawigator obiektów i paleta właściwości
- Edytor układu
- Edytor menu
- Edytor PL/SQL
- Debugger z interpreterem PL/SQL
- Edytor bibliotek obiektów
- Kreatory
 - bloków danych
 - układu
 - wykresów
- Administracja
 - kompilacja i uruchamianie
 - kompilacja PL/SQL



Organizacja formularza

Organizacja wizualna

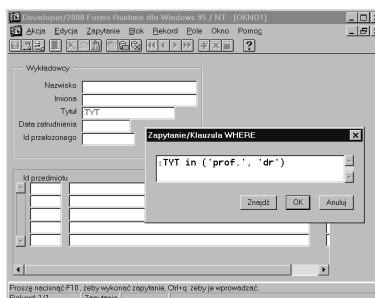
- Okna (*windows*)
 - okno konsoli
 - pozostałe okna
 - w środowisku Windows możliwa praca w MDI (okno MDI jest konsolą)
- Kanwy (*canvases*) i widoki (*viewports*)
 - wypełniające
 - nakładane
 - paski narzędzi
 - karty
- Ramki (*frames*)
- Grafika, elementy (*items*) i etykiety

Organizacja logiczna

- Formularz(*form*) — moduł
- Bloki (*blocks*) — dane z jednego źródła
 - jednorekordowe
 - wielorekordowe
- Elementy danych (*items*)
 - elementy bazy danych — możliwa edycja
 - elementy inne — tylko wyświetlanie
- Przypisania
 - elementy → blok,
 - elementy → kanwa
 - suwak bloku (do *danych*) → kanwa
 - kanwy → okno
 - elementy → ramka
 - ramka ↔ blok
 - ramka → kanwa

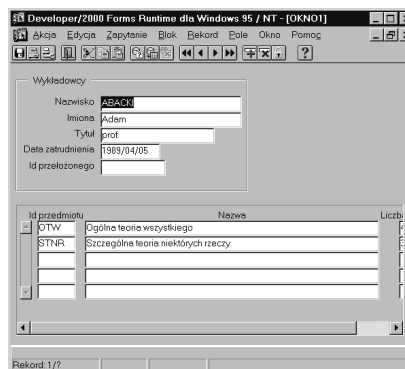
Wbudowana funkcjonalność formularza

- Menu standardowe
 - nawigacja po danych
 - sterowanie zapytaniem i transakcjami
 - dodawanie i usuwanie rekordów
 - pomoc i inne funkcje pomocnicze
- Standardowy pasek narzędzi
- Standardowe klawisze (zależne od platformy, spis w menu Help - Keys)
- QBE, czyli *query by form* (!)
 - proste wzorce z % i _
 - warunki z operatorami relacji
 - użycie zmiennych :zmienna i klauzuli WHERE
- Automatyczne podpowiedzi do pól: *hints* i *popup help*
- Ostrzeżenia przed operacjami grożącymi utratą danych
- Automatyeczna obsługa blokad
- Synchronizacja między blokami
- Walidacja i utrzymywanie spójności referencyjnej



Nawigacja na formularzu

- Nawigacja między polami
 - określona jest kolejność bloków i pól w blokach
 - „nawigowalność” za pomocą klawiatury oraz myszki może być różna
 - style nawigacji: ten sam rekord, zmiana rekordu, zmiana bloku
- Zmiana kanw
 - ukazuje się kanwa na której jest aktywne pole
 - ukazuje się ta część kanwy, która zawiera aktywne pole
 - kanwa wypełniająca zajmuje miejsce kanwy dotychczas widocznej
 - kanwa nakładana pojawia się „nad” kanwą wypełniająca
 - karta pojawia się na miejscu dotychczas widocznej karty
- Zmiana okien
 - aktywne staje się okno, w którym jest kanwa zawierająca aktywne pole



Budowa formularza

Struktura formularza

- Hierarchia składników
- Właściwości
 - każdy obiekt ma nazwę oraz wewnętrzny identyfikator (*object ID*)
 - zestaw właściwości zależy od rodzaju obiektu
 - niektóre właściwości mogą być zmieniane programowo

Rodzaje składników

- Moduł
- Obsługa danych: bloki, relacje, elementy
- Układ: okna, kanwy, ramki, elementy i grafika, atrybuty wizualne

- Programowanie: wyzwalacze, jednostki programu
- Zmienne: globalne, systemowe, parametry
- Biblioteki: dołączone biblioteki PL/SQL
- Obiektowość: klasy właściwości, grupy obiektów, biblioteki obiektów
- Obiekty pomocnicze
 - listy wartości (LOV) i grupy rekordów
 - alerty
 - edytory
 - budziki (*timers*)
 - menu podręczne (*pop-up menus*)
 - listy parametrów

Układ (*layout*)

Składniki układu

- Okna
- Kanwy
- Ramki
 - używane przez kreator układu
 - przypisane do bloku i kanwy
 - organizują układ *wszystkich* elementów bloku
- Grafika
 - teksty
 - prostokąty, linie i strzałki, łamane, owale, elipsy, łuki
- Elementy z etykietami
 - etykieta powiązana z elementem

– elementy jednego bloku mogą być na różnych kanwach (o ile nie użyto ramek)

Edycja układu

- Edycja układu dotyczy jednej kanwy
- W edytorze można wyświetlić także kanwy nakładane lub karty

Atrybuty wizualne

- Definiują cechy wizualne: kolory, czcionkę i wypełnienie
- Używane do grupowania cech wizualnych
- Mogą być przypisane do kanw, ramek, elementów i etykiet
- Bieżący rekord może być wyróżniony specjalnym atrybutem wizualnym

Bloki

- Rodzaje
 - bazowe — dane z jednego źródła (np. tabeli)
 - kontrolne — elementy niebazowe (wyliczone, pomocnicze)
- Źródło i cel danych (mogą być różne)
- Właściwości obsługi danych
 - `WHERE`, `ORDER BY` i *hint* dołączane do zapytania
 - dozwolone operacje
 - maksymalny czas zapytania i liczba pobieranych wierszy
- Prezentacja
 - jeden rekord albo wiele rekordów (pionowo lub poziomo)
- pasek przewijania danych (przypisany do *kanwy*)
- Bufor rekordów
 - ustawiane parametry
 - dla bloków bazowych wypełniany automatycznie
 - możliwe wypełnianie programowe
 - operacja czyszczenia (`CLEAR_*`) czyści bufor bez zapisu zmian
- Status rekordów
 - informuje o tym czy rekord jest pobrany z zapytania, zmieniony czy nowy
 - statusy bloku i formularza wynikają ze statusu rekordów

Źródło danych

- Możliwe źródła danych (*data sources*)
 - tabela
 - klauzula `FROM` (zapytanie zamiast nazwy tabeli)
 - procedura zwracająca (przez pierwszy parametr)
 - * kursor (`REF CURSOR`)
 - * tablicę rekordów (`INDEX BY BINARY_INTEGER`)
 - wyzwalacze transakcyjne (`ON-*`)
- Sposób definiowania
 - kreator bloku — dla tabel i procedur
 - właściwości bloku
 - * typ i nazwa źródła (lub zapytanie)
 - * kolumny i argumenty (dla procedur)

Cele dla danych

- Możliwe cele dla danych (*data targets*)
 - tabela
 - procedura przyjmująca (przez pierwszy parametr) tablicę rekordów
 - wyzwalacze transakcyjne (ON-*)
- Sposób definiowania
 - kreator bloku — dla tabel i procedur
 - właściwości bloku
 - * typ i nazwa celu
 - * nazwy procedur kolumny i argumenty dla wstawiania, modyfikacji, usuwania i blokowania

Uwarunkowania wyboru rodzaju źródła i celu

- QBE dostępne dla tabeli i klauzuli FROM
- WHERE i ORDER BY nie używane przez procedury
- Przetwarzanie tablicowe (*array processing*) niedostępne dla procedur zwracających
- Wyzwalacze transakcyjne pozwalają całkowicie zastąpić domyślne działanie Forms

Walidacja danych

- Jednostka walidacji (*Validation Unit*) — właściwość formularza: element, rekord, blok, formularz
- Klucz główny
 - właściwość bloku „wymuszanie klucza głównego” (*Enforce Primary Key*)
 - właściwość elementu „klucz główny”
- Klucz obcy → koordynacja bloków
- Walidacja programowa — wyzwalacze: WHEN-VALIDATE-RECORD i WHEN-VALIDATE-ITEM
- Walidacja elementu
 - typ danych
 - maska formatu
 - maksymalna długość
 - zakres wartości
 - elementy wymagane:
 - * własność elementu „wymagany” (*required*)
 - * własność formularza „odroczone wymuszenie” — odroczone sprawdzenie do czasu walidacji rekordu
 - walidacja za pomocą listy LOV — właściwość elementu tekstowego

Koordinacja bloków

Relacje (*relations*)

- Określenie sposobu koordynacji między parą bloków
 - blok podrzędny i warunek złączenia
 - właściwości koordynacji i integralności referencyjnej
- Automatycznie tworzone są wyzwalacze koordynujące
- Można budować zależności złożone, np. macierzowe
- Możliwa także koordynacja na podstawie referencji obiektowych

Mechanizm koordynacji

- Fazy koordynacji
 - zdarzenie wywołujące: zmiana rekordu *master*

- faza czyszczenia (*clear*)
- faza wypełniania (*population*)

- Sterowanie chwilą koordynacji
 - koordynacja odroczone
 - automatyczne zapytanie

Wymuszanie integralności referencyjnej

- Operacje bez elementu nadrzędnego zabronione — właściwość relacji
- Wypełnianie klucza obcego: właściwość elementu „wartość kopiowana z elementu” (*Copy Value from Item*)
- Usuwanie elementu nadrzędnego — właściwość relacji: nie izolowane, kaskada, izolowane

Zapis danych (*posting*)

- Czas zapisu
 - normalnie zapis wykonywany bezpośrednio przed COMMIT-em
 - zapis może być wymuszony przez procedurę POST
- Zakres zapisywania
 - normalnie zapisywane całe zmienione wiersze
 - można wymusić zapis tylko zmienionych kolumn (właściwość bloku)
- Sprawdzanie, czy są zmiany do zapisania:
`IF system.form_status = 'CHANGED' THEN`

Blokady

- Tryby blokowania (właściwość bloku)
 - natychmiast przy zmianie
 - opóźniony do COMMIT

Elementy

Typy elementów

- Pole tekstowe
- Lista
- Grupa radiowa
- Pole wyboru (*check box*)
- Pole wyświetlane (*display item*)
- Przycisk
- Wykres Oracle Graphics (*chart item*)
- Obraz (*image*), dźwięk (*sound item*)
- Kontener OLE, format ActiveX

Ważniejsze własności elementów

- Cechy wizualne i nawigacja
 - element włączony — osiągalny w nawigacji
 - obiekt widoczny
 - etykieta powiązana z elementem
- Dozwolone operacje
 - wstawianie, modyfikacja
 - modyfikacja (tylko) wartości NULL dozwolona

Elementy i dane

- Pochodzenie danych
 - element bazowy — przypisany do kolumny
 - element niebazowy: wyliczany, pomocniczy — do wprowadzania, ukryty — p.o. zmiennej
- Dane wyliczane
 - synchronizowany (lustrzany)
 - wyliczany formułą w PL/SQL
 - podsumowanie
 - wyliczany wyzwalaczem (POST-QUERY)
- Podsumowania
 - funkcja podsumowująca — jak funkcje grupowe SQL
 - podsumowanie obejmuje wartości elementu w ramach bloku
- podsumowanie na ogół umieszcza się w bloku podsumowywanym, ale wizualnie — wśród danych bloku nadrzędnego
- wymagane cechy bloku: pobieranie wszystkich rekordów lub wcześniejsze wyliczanie podsumowań
- Typ danych (właściwość elementu) — powinien odpowiadać typowi źródła
- Wartość domyślna
 - stała
 - wartość elementu (:nazwa_elementu)
 - zmienna systemowa (np. \$\$DATE\$\$), globalna lub parametr
 - odwołanie do sekwencji:
:SEQUENCE.nazwa_sekwencji.NEXTVAL

Listy wartości (LOV)

- Zastosowanie: wstawianie wartości
 - na podstawie innej tabeli (klucze obce)
 - ze statycznej listy
- Funkcjonalność
 - auto-redukcja
 - wyszukiwanie wg wzorca
 - pobieranie wierszy: zwykłe (pełne) i z filtrowaniem przed wyświetleniem
 - użycie LOV do walidacji
- Definiowanie
 - obiekt typu LOV
 - odwzorowania kolumn^a
 - * nazwa i tytuł
 - * szerokość wyświetlania (0 = ukryta)
 - * element zwracany (zwracanie kilku wartości — dla kluczy złożonych)
 - * walidacja przez LOV i auto-redukcja dotyczy pierwszej kolumny
- Pochodzenie wartości
 - obiekt typu „Grupa rekordów”:
zapytanie lub statyczna lista rekordów
 - możliwe programowe wypełnianie listy
- Wywołanie
 - dowiązanie za pomocą właściwości elementu tekstowego
 - wywołanie programowe: LIST_VALUES (dla bieżącego pola) lub SHOW_LOV

^aPozycje z listy kolumn usuwa się przez Ctrl-←, pomoc dla tego typu poleceń — Ctrl-K

Listy

- Przeznaczenie: do wybierania z nielicznego i raczej statycznego zbioru wartości
- Style listy
 - *t-list* – lista zwykła
 - *pop-list* – lista rozwijana
 - *combo box* – pole tekstowe z rozwijaną listą odpowiedzi
- Definiowanie wartości
 - statyczne — właściwość „elementy na liście”
 - zmieniane programowo
 - wypełniane programowo na podstawie grupy rekordów (jak w LOV)
- Specjalna obsługa wartości
 - wartość NULL dopuszczalna gdy podana na liście, nawet jeśli element jest *required*
 - odwzorowanie innych wartości (*other values*)
- Zmiana wartości może wyzwać wyzwalacz WHEN=typ_elementu-CHANGED
- Grupy radiowe i pola wyboru
 - Podobne funkcjonalnie do listy
 - Nie mogą być zmieniane dynamicznie
 - Przyciski radiowe definiowane jako obiekty podrzędne grupy

Programowanie formularza

Ogólne zasady

- Język PL/SQL, ale uboższy niż w bazie danych
- Rodzaje kodu
 - wyzwalacze
 - jednostki programu (procedury, funkcje, pakiety) lokalne i biblioteczne
 - kody elementów menu
- Rola wyzwalaczy
 - obsługa zdarzeń wywołanych przez operatora i bazę danych
 - obsługa przycisków
 - zastępowanie typowych działań Forms
- Podprogramy wbudowane (*built-ins*)
 - wykonują różne działania w formularzu i na danych
 - możliwe ograniczenia:
 - * podprogramy zastrzeżone (*restricted built-ins*)
 - powodują nawigację
 - nie są dozwolone w niektórych wyzwalaczach, np. PRE-* i POST-*
 - * podprogramy zabronione w trybie wprowadzania zapytania

Wyzwalacze (*triggers*)

Rodzaje wyzwalaczy

- Wbudowane — obsługa zdarzeń i klawiszy
- Zdefiniowane przez użytkownika (*user named triggers*) — wywoływane programowo przez
`EXECUTE_TRIGGER('nazwa_wyzwalacza');`

Cechy wyzwalaczy

- Cechy opisujące wyzwalacz
 - dopuszczalny zakres
 - zastępowane działanie standardowe
 - dopuszczalna zawartość: SELECT, DML, procedury zastrzeżone
 - czy może działać w trybie wprowadzania zapytania
 - skutek niepowodzenia
- Właściwości wyzwalacza
 - dopuszczalność w trybie wprowadzania zapytania
 - hierarchia wykonywania

Klasyfikacja wyzwalaczy

- Wyzwalacze obsługi bloków, np. WHEN-CLEAR-BLOCK, WHEN-CREATE-RECORD
- Wyzwalacze zdarzeń interfejsu, np. WHEN-BUTTON-PRESSED, WHEN-CHECKBOX-CHANGED
- Wyzwalacze obsługi klawiatury, np. KEY-COMMIT, KEY-OTHERS
- Wyzwalacze nawigacyjne
 - typu PRE- lub POST-, np. PRE-BLOCK — zakaz użycia procedur zastrzeżonych!
 - typu WHEN-NEW-*-INSTANCE, np. WHEN-NEW-BLOCK-INSTANCE
- Wyzwalacze koordynujące, np. ON-POPULATE-DETAILS
- Wyzwalacze obsługujące komunikaty: ON-ERROR, ON-MESSAGE
- Wyzwalacze trybu zapytania: PRE-QUERY, POST-QUERY
- Wyzwalacze transakcyjne
 - typu PRE- lub POST-, np. PRE-INSERT — uzupełniają działanie Forms
 - typu ON-, np. ON-INSERT — zastępują typowe działanie Formsnp. PRE-INSERT, ON-INSERT, POST-INSERT
- Wyzwalacze walidacyjne: WHEN-VALIDATE-ITEM, WHEN-VALIDATE-RECORD

Typowe zastosowania wyzwalaczy

- Inicjalizacja formularza (zmiennych itp.): WHEN-NEW-FORM-INSTANCE
- Walidacja danych: WHEN-VALIDATE-ITEM, WHEN-VALIDATE-RECORD
- Obsługa przycisków: WHEN-BUTTON-PRESSED
- Przechwytywanie komunikatów: wyzwalacze ON-ERROR, ON-MESSAGE i funkcje ERROR_TYPE, ERROR_CODE, ERROR_TEXT
- Sprawdzanie integralności: wyzwalacze transakcyjne PRE-
- Auditing: wyzwalacze transakcyjne POST-
- Wstawianie wartości wyliczonych do elementów bazowych: wyzwalacze transakcyjne PRE-
- Wyliczanie wartości elementów niebazowych: POST-QUERY
- Kontrolowanie nawigacji
 - zakazy: wyzwalacze nawigacyjne PRE- i POST-
 - wywoływanie (np. przeskoki): wyzwalacze nawigacyjne WHEN-NEW-*-INSTANCE
- Zdarzenia czasowe
 - powoływanie zegarów: procedury CREATE_TIMER, DELETE_TIMER, SET_TIMER
 - obsługa zdarzenia: wyzwalacz WHEN-TIMER-EXPIRED (nazwę timera pobrać funkcją GET-APPLICATION-PROPERTY)

Zmienne

- Zmienne systemowe
 - informują o stanie formularz
 - niektóre można zmieniać programowo
 - użycie: `:SYSTEM.nazwa_zmiennej`
- Zmienne globalne
 - powoływanie: *przez podstawienie*
 - inicjalizacja warunkowa:
`DEFAULT_VALUE(wartość, 'nazwa_zm')`
 - użycie: `:GLOBAL.nazwa_zmiennej`
 - usuwanie: `ERASE(nazwa_zmiennej)`
 - zmienne globalne trwają przez cały czas działania *runtime* — można przekazywać dane między kolejnymi formularzami
 - typ danych: znakowe
- Parametry
 - systemowe lub definiowane w czasie projektowania
 - nadanie wartości: w linii wywołania lub za pomocą obiektu *parameter list* przy wywołaniu programowym
 - użycie: `:PARAMETER.nazwa_parametru`
 - typ danych: znakowe
- Elementy formularza
 - użycie `:nazwa_bloku.nazwa_elementu`
 - elementy są dostępne tylko dla kodu zawartego w module formularza
 - w bibliotekach i modułach menu trzeba użyć odwołań pośrednich

Identyfikatory i odwołania pośrednie

Identyfikacja obiektów

- Identyfikatory obiektów
 - nazwa
 - wewnętrzny id specjalnego typu — nazwa typu zgodne z typem obiektu
- Znajdowanie wartości id

```
DECLARE
  id_var  typ_obiektu;
BEGIN
  id_var := FIND_typ_obiektu('nazwa');
  ...
END;
```
- Stwierdzanie istnienia obiektu i podanym id: ID_NULL(id)
- Użycie id: zamiast nazw w przeciążonych formach procedur wbudowanych

- Powody stosowania
 - lepsza wydajność
 - procedury generyczne

Odwołania pośrednie

- Dane tylko typu znakowego
- Pośredni zapis:
COPY(wartość, 'nazwa_zm')
- Pośredni odczyt: NAME_IN('nazwa_zm')
- Powody stosowania
 - brak dostępu do zmiennej w czasie kompilacji
 - procedury generyczne
 - wstawianie operatorów porównań do elementów typu nietekstowego w trybie *enter query*

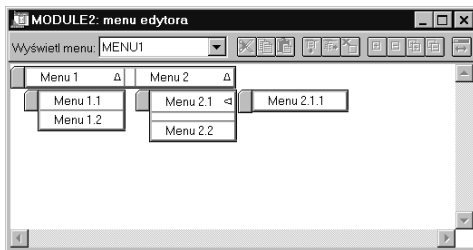
Programowanie interakcji z operatorem

- Okno konsoli
- Komunikaty
 - komunikat w linii komunikatów
 - * wywołanie:
MESSAGE(tekst,
ACKNOWLEDGE|NO_ACKNOWLEDGE)
 - * obsługa programowa: GET_MESSAGE,
CLEAR_MESSAGE
 - sygnał dźwiękowy: BELL
 - wymuszenie wyświetlania: SYNCHRONIZE
- Alerty
 - funkcja: wyświetlanie komunikatu z możliwością reakcji
 - możliwość użycia 1..3 przycisków
 - definiowanie: w czasie projektowania
 - obsługa programowa:
SET_ALERT_PROPERTY,
SET_ALERT_BUTTON_PROPERTY
 - wywołanie: SHOW_ALERT (zwraca wartość oznaczającą wciśnięty przycisk)

Menu

Użycie menu

- Uruchamiane tylko wraz z formularzem
- Menu standardowe — typowe operacje
- Menu użytkownika
 - zdefiniowane jako osobny moduł `.mmb`
 - przyłączone do modułu formularza (`.fmb`) — właściwość formularza



Definiowanie menu

- Narzędzie: edytor menu
- Budowa
 - menu – węzły drzewa
 - elementy – liście drzewa
- *Menu toolbar* – budowanie paska na podstawie menu

Elementy menu

- Rodzaje elementów: zwykłe, radiowe, zaznaczalne, separatory, specjalne (wytnij, wklej, cofnij, okno itp.)
- Rodzaje poleceń w elementach menu
 - puste
 - menu – wywołanie menu podrzędnego
 - PL/SQL
 - * realizuje polecenia
 - * używa odwołań pośrednich do elementów formularza
 - * może używać parametrów standardowych: `:UN` – nazwa użytkownika, `:PW` – hasło

Wywoływanie innych aplikacji z Forms

- Dynamiczny SQL: procedura `FORMS_DDL('zdanie_sql')`;
- Wywoływanie innych programów: procedura `HOST('polecenie')`;
- Wywoływanie raportów
 - obiekt typu „Raport” i procedura `RUN_REPORT_OBJECT(report_id)`;
- Integracja grafiki
 - element typu „Wykres” i kreator wykresów
 - procedura `RUN_PRODUCT`
- Aplikacje wieloformularzowe
 - sposoby wywoływania formularzy
 - * `OPEN_FORM` – niezależny formularz, wywołujący nadal działa
 - * `CALL_FORM` – nowy formularz, wywołujący nieaktywny, na „stosie”
 - * `NEW_FORM` – nowy formularz, wywołujący usunięty
 - zachowywana jest sesja (w `OPEN_FORM` można wymusić nową)
 - zachowywane są wartości zmiennych globalnych
 - uwaga na `POST`, `COMMIT` itp.
- Przekazywanie parametrów
 - obiekt typu `ParamList`
 - tworzenie programowe: `CREATE_PARAMETER_LIST`, `ADD_PARAMETER`
 - przekazywanie w `RUN_REPORT_OBJECT`, `RUN_PRODUCT` i `*_FORM`

Aplikacje Forms w architekturze trójwarstwowej

Architektura trójwarstwowa

- chudy klient — przeglądarka WWW + Java
- serwer aplikacyjny (*Oracle Application Server*)
- serwer danych

Forms w architekturze trójwarstwowej

- Serwer – działa na serwerze aplikacyjnym *Oracle Application Server*
 - serwlet pełniący rolę procesu nasłuchującego
 - aplikacja J2EE na instancji kontenera OC4J
- Klient: *Forms Client* – aplet (Java) pobierany z serwera
 - dla przyspieszenia startu na kliencie może się automatycznie zainstalować tzw. Jinitiator (biblioteki .jar)

- Moduł .fmx: skompilowany w środowisku serwera i rezydujący tamże
- Wywołanie aplikacji:
 - strona HTML (wspólna dla wszystkich modułów), wywołująca aplet *Forms Client*
 - parametry przekazywane
- Cechy rozwiązania
 - generyczne — jeden mechanizm do uruchamiania wszystkich modułów Forms
 - bezpieczne — kodowana transmisja klient – serwer aplikacji
 - zapewnia scentralizowane zarządzanie aplikacją
 - wymaga dużych zasobów na serwerze
 - nie wymaga specyficznego oprogramowania na kliencie