

1 Wstęp

Narzędzia Oracle Designer w wersji 6i pozwalają na całkowite oparcie prac projektowych o repozytorium i generatory w zakresie dokumentacji projektu, budowy kodu i zarządzania wersjami dokumentów i kodu. Niniejszy dokument ma na celu praktyczne zaprezentowanie możliwości Designera w generowaniu aplikacji z uwzględnieniem nowych cech wersji 6i.

100% generacji jest praktyczną potrzebą takiego budowania aplikacji, aby zminimalizować koszty przyszłych zmian poprzez pełną dokumentację kodu, automatyczne śledzenie zależności oraz narzucaną w tym procesie standaryzację.

Jednym z najważniejszych generatorów Oracle Designera jest Forms Generator. Pozwala on na całkowitą generację aplikacji poprzez:

- Możliwość przechowywania kodu użytkownika w repozytorium,
- Oparciu generacji na właściwościach obiektów, szablonach oraz bibliotekach obiektowych.
- Wersjonowanie i zarządzanie zależnościami pomiędzy składowymi wewnątrz repozytorium.

Dokument opisuje dalej szczegółowo proces generacji koncentrując się na tym właśnie generatorze.

2 Nowe cechy Designera 6i

Rozdział ten omawia tylko te cechy Oracle Designera 6i, które są istotne dla głównego tematu tego dokumentu.

Celem zmian Oracle Designera w wersji 6i było:

- Zwiększenie możliwości obsługi dużych projektów przy pomocy repozytorium,
- Zwiększone wsparcie dla rozwoju Oracle Application
- Wprowadzenie i rozszerzenie współpracy Oracle 8i

Zmiany te zostały osiągnięte poprzez istotne zmiany w samym repozytorium wprowadzając zarządzanie konfiguracjami oraz zależnościami. Zostały także rozszerzone możliwości generatorów kodu głównie w zakresie generatora Oracle Forms oraz generatora Serwera.

Zmiany w repozytorium

Repozytorium jest integralną częścią Oracle Designera służy ono do zarządzania strukturami danych właściwymi dla metamodelu Designera.

Aby w pełni spełniać swoją rolę repozytorium zostało wyposażone w następujące cechy:

- Analiza wpływu zmian,
- Możliwość rozszerzeń,
- Wsparcie dla plików i katalogów,
- Wersjonowanie na poziomie poszczególnych obiektów,
- Możliwość definiowania konfiguracji,
- Obszary robocze.

Analiza wpływu zmian

Dotąd Designer realizował analizę wpływu zmian w ramach swoich danych strukturalnych. Na przykład pozwalał na określenie jakie moduły trzeba będzie zmodyfikować w przypadku zmiany kolumny tabeli. Nie było to wystarczające dla danych, które z różnych względów nie mogły być opisane strukturalnie. Na przykład procedury PL/SQL składowane w bazie danych, Raporty, Formatki, kod PRO*C, JSQL itd. Obecnie wprowadzono funkcjonalność pozwalającą na przechowywanie i utrzymywanie także takich informacji.

Wraz z narzędziem dostarczone są standardowe analizatory pozwalające na analizę niektórych typów np. Oracle Forms. Istnieje także możliwość na tworzenie swoich własnych analizatorów.

Możliwość rozszerzeń

Możliwość rozszerzeń pozwala na umieszczanie w repozytorium obiektów typów określonych przez użytkownika. Typy takie mogą być używane jako typy elementów, związków itd. Dane tak zdefiniowanych typów mogą być kontrolowane przez mechanizmy repozytorium.

Dostęp do nowych typów możliwy jest zarówno przez narzędzia Oracle Designera jak i przez API.

Wsparcie dla plików i katalogów

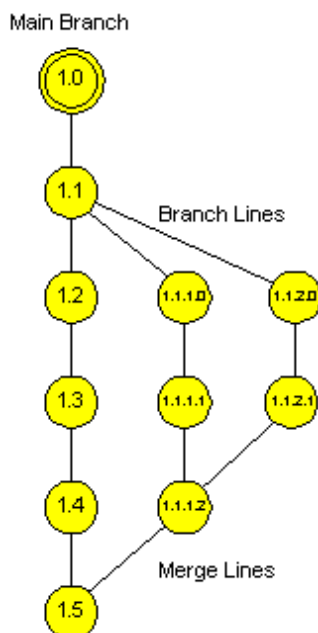
Oprócz danych strukturalnych jak definicję tabel, modułów itp. repozytorium pozwala na przechowywanie informacji niestukturalnych podobnie jak zwykły system plików.

Istnieje możliwość tworzenia plików i katalogów, wiązania ich z rzeczywistymi systemami plików i ich wersjonowania.

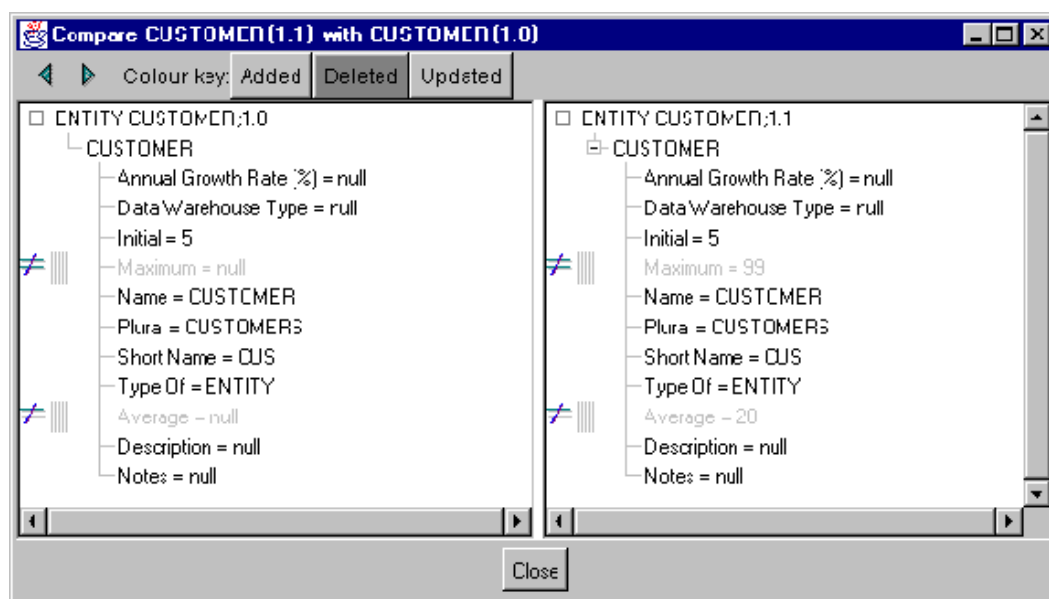
Wersjonowanie na poziomie poszczególnych obiektów

Repozytorium pozwala na wersjonowanie nie tylko całych systemów aplikacji ale poszczególnych obiektów.

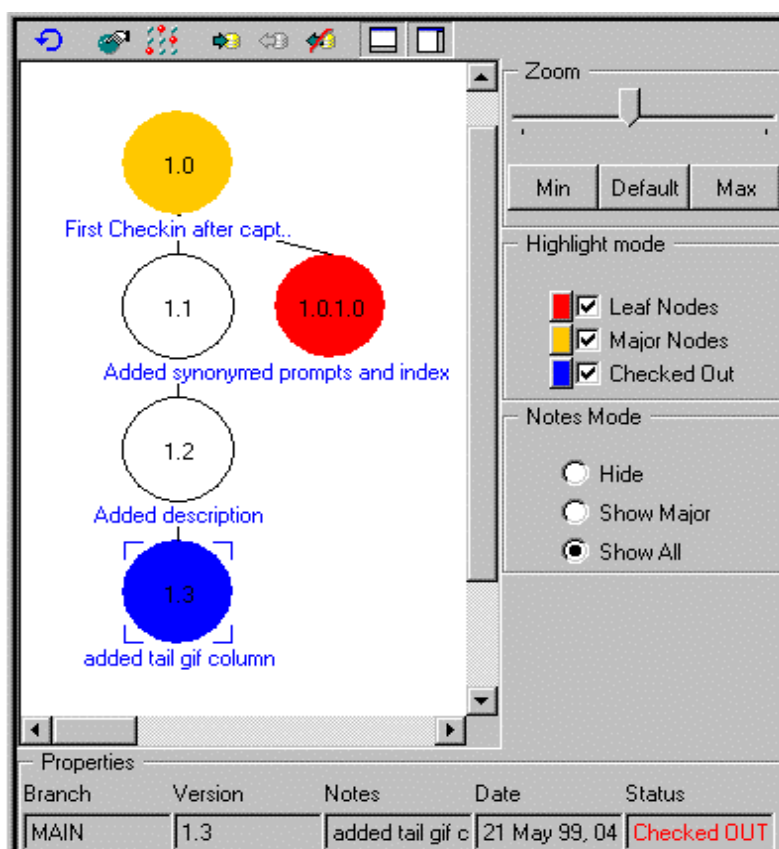
Dzięki wprowadzeniu możliwości definiowania gałęzi zmian możliwe jest prowadzenie prac projektowych na jednym obszarze w wielu niezależnych ścieżkach. Na przykład zespół zajmujący się dodawaniem nowych cech i zespół poprawiające błędy w wersjach już działających muszą widzieć różne wersje tych samych elementów.



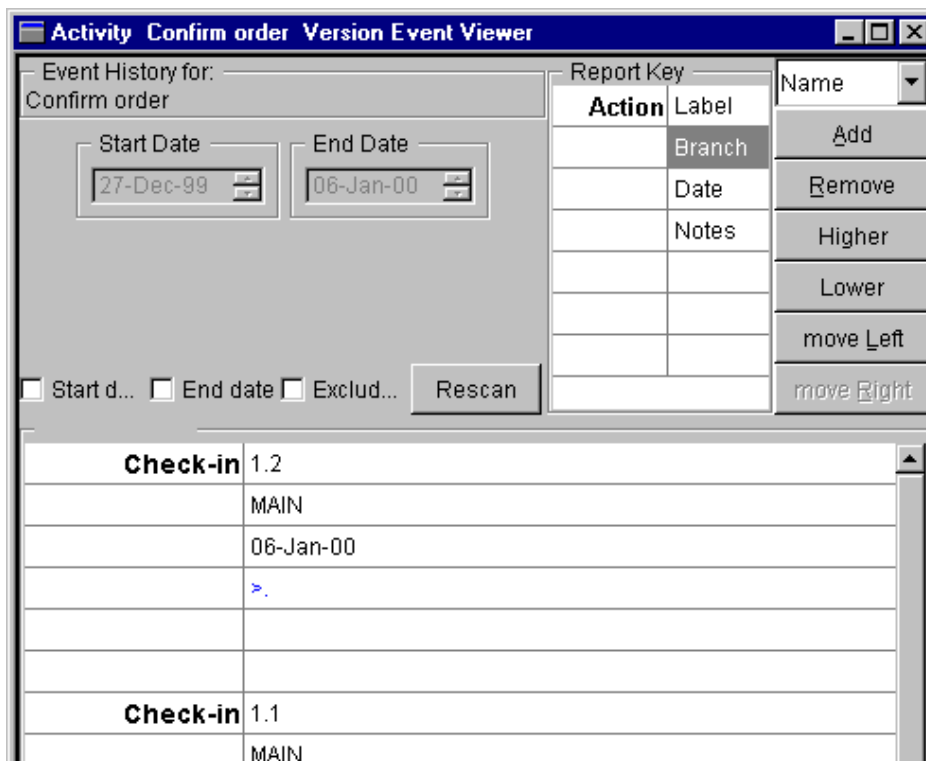
Proces łączenia różnych wersji w całość jest wspierany przez automatyczne narzędzia porównywania i łączenia.



Repozytorium jest wymoszone także w narzędzia zarządzania wersjami jak na przykład przeglądanie historii zmian:



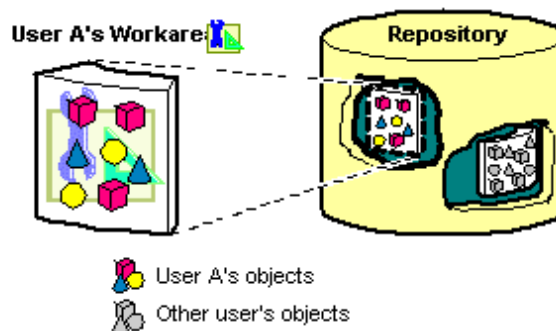
i zdarzeń jakie je powodowały:



Obszary robocze

Obszary robocze zostały wprowadzone aby umożliwić pracę na wersjonowanym repozytorium. W danej chwili pracująca osoba musi widzieć dokładnie określoną wersję obiektu z dokładanie określonej ścieżki.

Obszary robocze definiuje się podając zestaw reguł np. „pokaż wszystkie najnowsze obiekty z gałęzi produkcja oraz wersja 1.15 tabeli EMP”.

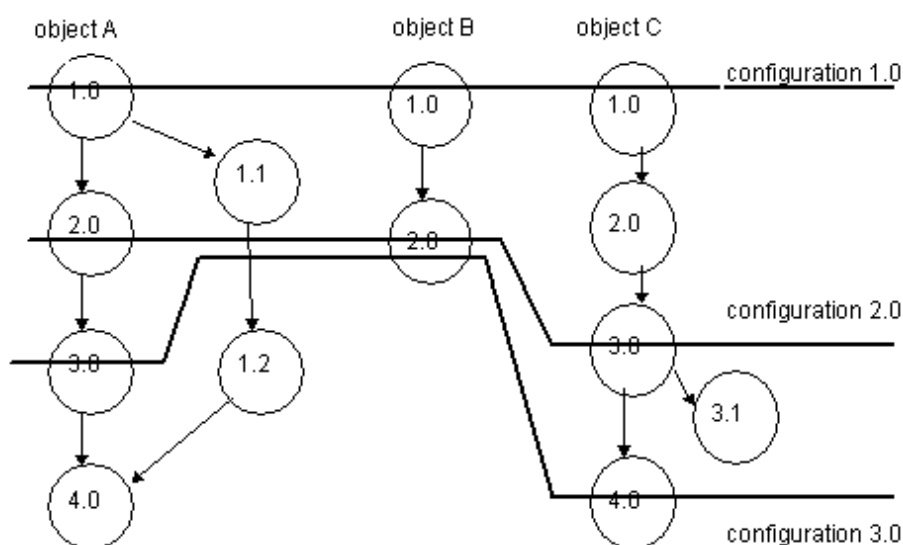


Obszary robocze mogą być definiowane przez poszczególnych użytkowników albo współdzielone i administrowane przez zarządzającego repozytorium.

Możliwość definiowania konfiguracji

W czasie życia systemu publikowane są zestawy obiektów repozytorium jako wydanie oprogramowania.

Istotną informacją jest jakie wersje zostały umieszczone w takim wydaniu. Np. jaka wersja modułu edycji pracowników odpowiada wersji 3.14 tabeli EMP.



Zewnętrzny dostęp do repozytorium

W stosunku do wersji poprzednich rozszerzono i poprawiono metody dostępu do struktur repozytorium z API w PL/SQL. Obecnie planuje się opieranie o repozytorium także innych niż Oracle Designer narzędzi.

Oracle Designer

Niezależnie od zmian samego repozytorium zostały zmienione komponenty samego Designera.

Generator Serwera

Java

Wprowadzono możliwość przechowywania kodu w Javie. Jego edycja odbywa się przy pomocy tego samego edytora kodu który służy do edycji PL/SQL.

Hurtownie danych

Wprowadzono wsparcie dla następujących obiektów wspierających tworzenie hurtowni danych:

1. Materializowane perspektywy,
2. Indeksy bazujące na funkcjach,
3. Użycie klauzuli SAMPLE
4. Rozszerzenia klauzuli CREATE / REBUILD INDEX o opcję COMPUTE STATISTICS.

System Designer

Rozszerzony o możliwość pokazywania nowych typów wprowadzonych przez generator serwera i forms.

Zmieniony edytor PL/SQL.

Generator Forms

Wprowadzono następujące nowe cechy:

1. Relatywne zatrzymania tabulatora,
2. Możliwość rozłożenia jednego bloku pomiędzy wiele obszarów,
3. Współrzędne rzeczywiste przy definiowaniu dekoracji,
4. Generacja drzewek,
5. Generowanie bloków obok siebie,
6. Generacja do Oracle Forms 6i,
7. Możliwość użycia rozszerzeń Javy,
8. Automatyczny podgląd wyniku generacji w środowisku Javy.

3 Środowisko projektowe

Przed rozpoczęciem etapu budowy kodu konieczne jest zdefiniowanie, utworzenie i budowa odpowiedniego środowiska projektowego. W skład takiego środowiska wchodzi następujące komponenty:

1. Biblioteka projektu,
2. Standardy tworzenia kodu i dokumentowania,
3. Parametry dla generatora,
4. Wspólne biblioteki z podstawową funkcjonalnością,
5. Szablony,
6. Skrypty publikujące kod do środowiska lub środowisk docelowych,
7. Procedury tworzenia kopii zapasowych i odtwarzania.

Biblioteka projektu

Biblioteka ma fundamentalne znaczenie dla porządku w projekcie. Zawiera ona:

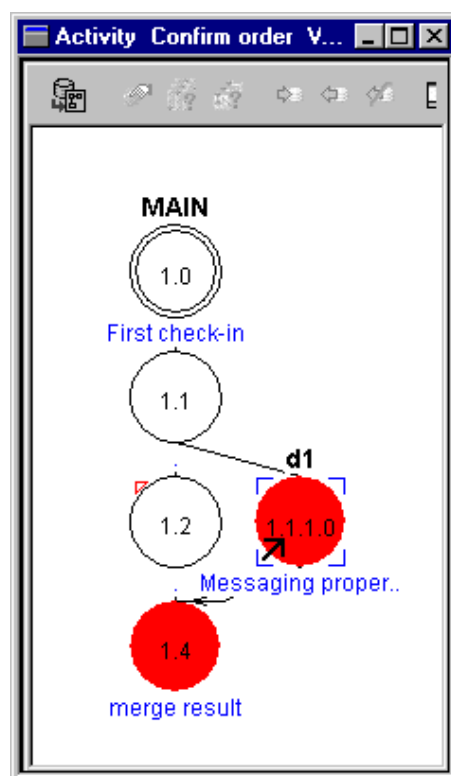
- Dokumenty,
- notatki projektowe,
- notatki ze spotkań,
- arkusze pracy,
- skrypty testowe oraz
- gotowy kod programów.

Dzięki zastosowaniu Oracle Designera możliwe jest umieszczenie biblioteki w repozytorium. Pozwala to na centralne zarządzanie jej zawartością oraz wersjonowanie.

Zastosowanie wersjonowania do kontroli kodu

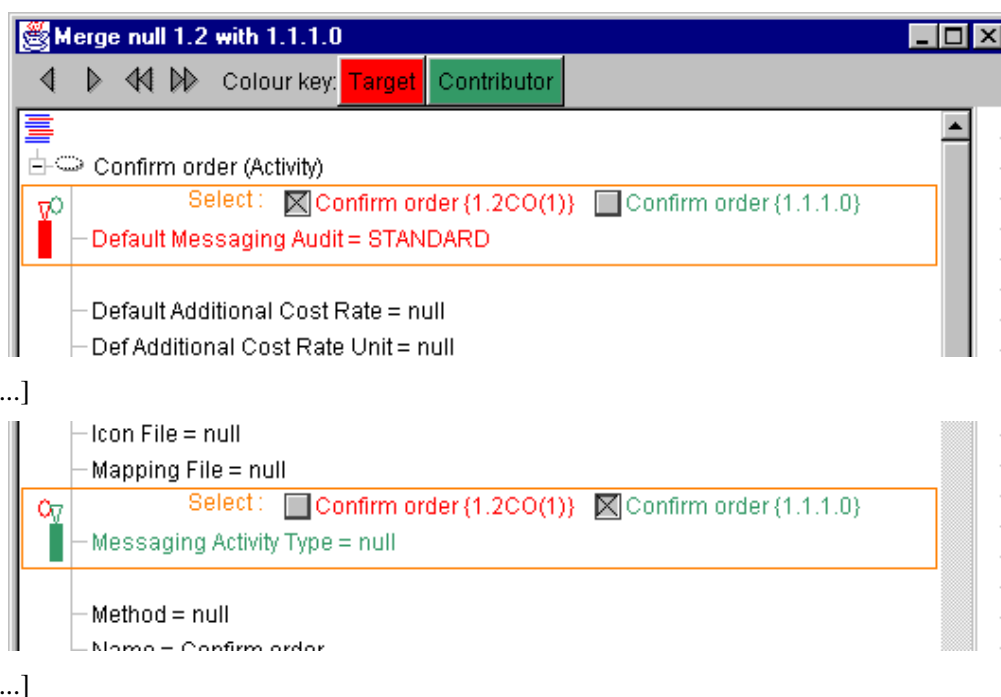
Możliwości wersjonowania repozytorium mogą łatwo posłużyć do implementacji wersjonowania kodu w projekcie zarówno w zakresie wygenerowanego kodu jak i metamodelu.

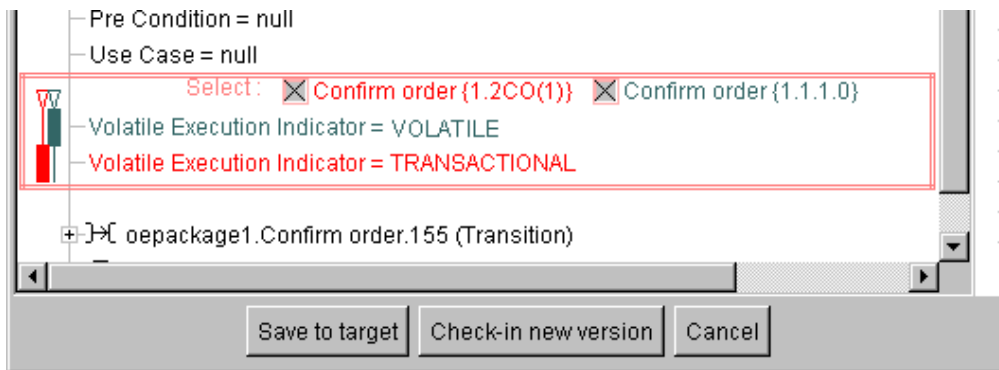
W pracach projektowych wersjonowanie jest używane przede wszystkim do prowadzenia prac równoległych oraz do szybkiego wycofywania modyfikacji (np. do poprzedniej stabilnej wersji).



Na powyższym rysunku wersja 1.1 została opublikowana jako wydanie oprogramowania i była rozwijana dalej jako wersja 1.2 równocześnie powstała wersja 1.1.1.0 jako patch 1.0 poprawiająca błędy z produkcji. Po zakończeniu prac powstała wersja 1.4 jako wspólna wersja uwzględniająca zmiany 1.2 i 1.1.1.0 (po rozwiązaniu konfliktów).

W przypadku wykrycia konfliktów jest automatycznie uruchamiane narzędzie do rozwiązywania konfliktów jak na poniższym przykładzie:





Współpraca pomiędzy członkami zespołu oparta o repozytorium

Repozytorium w sposób naturalny wspiera użycie wspólnego kodu i dokumentacji. Dostęp do danych odbywa się w oparciu o interfejs zapewniający możliwości wersjonowania z porównywaniem wersji dla standardowych typów modułów, łączenia gałęzi zmian oraz analizy zależności pomiędzy obiektami i kodem.

Użycie wspólnych obiektów strukturalnych jak Reusable Module Components czy Reusable LOV pozwala na zmniejszenie kosztów zmian kodu dla całego systemu.

4 Standardy generacji Oracle Forms

Elementy standardów

Generacja Oracle Forms dokonywana jest w oparciu o meta-model składowany w repozytorium, właściwości obiektów również z repozytorium oraz szablon generacji. Do tworzenia modułów używane są gotowe obiekty wchodzące w skład bibliotek obiektowych.

W skład standardów wchodzi także kod realizujący podstawową i wspólną funkcjonalność systemu. Kod może być przechowywany w repozytorium, szablonie i bibliotece obiektowej.

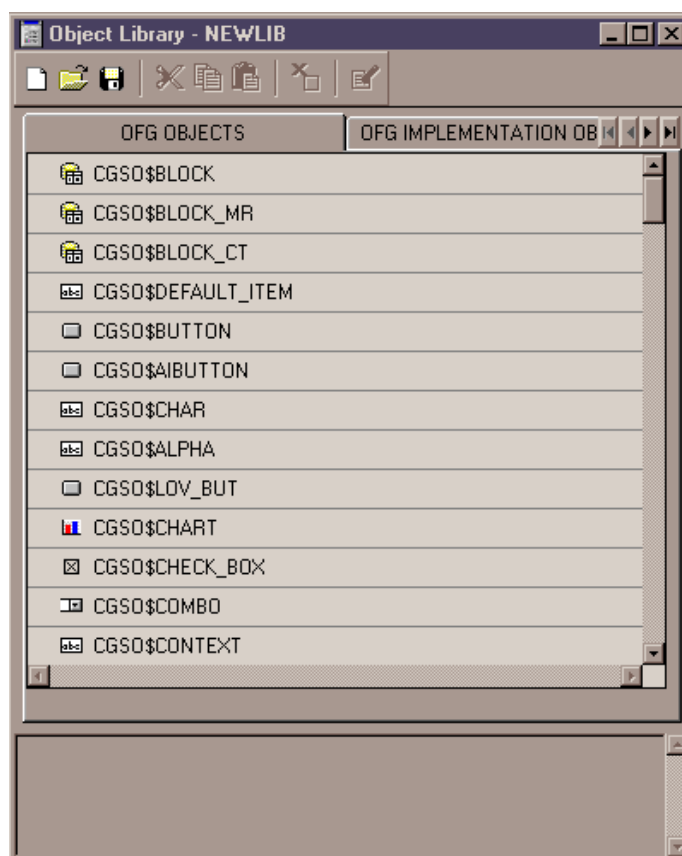
Na standard generacji składa się:

1. Zdefiniowanie i implementacja podstawowej funkcjonalności,
2. Szablon generacji (mogą się różnić dla różnych typów obiektów),
3. Biblioteki obiektowa zawierającej standardowe typy obiektów oraz wspólny kod,
4. Współdzielone komponenty w repozytorium,
5. Preferencje i współdzielone zbiory preferencji składowane w repozytorium.

Komponenty standardu muszą być szczegółowo przetestowane i sprawdzone przed rozpoczęciem właściwej generacji.

Użycie bibliotek obiektowych jako metoda standaryzacji

Oracle Forms od kilku już wersji wyposażone są w możliwość definiowania i używania bibliotek obiektowych. Praktycznie dowolny typ obiektu używanego w forms może zostać umieszczony w bibliotece i stamtąd używany w wielu miejscach. Ponieważ z większością obiektów mogą być związane akcje PL/SQL jest to również możliwość współdzielenia kodu.



Użycie obiektu z biblioteki obiektowej w module oznacza powołanie do życia jego instancji dziedziczącej właściwości rodzica. Dziedziczenie odbywa się na etapie kompilacji. To znaczy, że po zmianie biblioteki konieczne jest zbudowanie wszystkich modułów, które jej używają. Sprowadza się to utworzenie FMX na podstawie FMB (bez zmian w samym FMB).

Generator specjalnie wspiera użycie bibliotek poprzez automatyczne używanie obiektów z biblioteki na podstawie ich nazwy. Na przykład: tworząc pole tekstowe zostanie użyty element biblioteki CGSO\$CHAR (o ile taki istnieje w bibliotece).

Odpowiednie nazwy istnieją dla poszczególnych typów (CGSO\$CHAR, CGSO\$DATE...). Dla dodatkowych cech jak wielorekordowość, wymagalność itd. istnieją odpowiednie końcówki. Na przykład CGSO\$CHAR_MR_DO zostanie użyty w bloku wielorekordowym (MR) dla pól znakowych tylko do wyświetlania (DO).

Jeśli generator nie jest w stanie odszukać właściwego obiektu w bibliotece pobiera najbliższy zgodnie z hierarchią. Na przykład jeśli nie można odszukać CHSO\$NUMBER_MD zostanie użyty CGSO\$NUMBER a jeśli ten nie jest zdefiniowany to CGSO\$CHAR_MD. Pełne drzewo jest opisane w systemie pomocy elektronicznej w rozdziale „Hierarchy of standard source objects”.

Niezależnie użytkownik może tworzyć swoje obiekty i używać je jawnie w kodzie. W repozytorium dla każdego obiektu można podać nazwę obiektu z biblioteki obiektowej jaka ma zostać użyta do jego implementacji. Nadpisuje to standardowo przyjmowane użycia. Przykładem jest obiekt STD\$CHAR_UPPER do obsługi pól wymuszających wprowadzanie tekstu dużymi literami.

Table Usage		ACME_MEMBERS
+	Column	CITY
x	Version	
	Name	CITY
	Usage Sequence	790
	Template/Library Object	STD\$CHAR_UPPER
	Operations	

Szablony generacji

Od wersji poprzednich znaczenie szablonów generacji zmalało. Obecnie większość funkcjonalności zostało przeniesione do biblioteki obiektowej.

Szablon jest wykorzystywany przez generator, który na jego podstawie buduje cały moduł z instancji obiektów pobieranych z biblioteki.

Hierarchie bibliotek aplikacyjnych

Pomimo łatwości zmian w bibliotece obiektowej należy unikać umieszczania tam złożonego kodu PL/SQL, ponieważ nie jest go łatwo testować a zmiany kodu, które mogą być dość częste wymagają sporego nakładu pracy na regeneracji całego kodu.

Lepszym rozwiązaniem jest umieszczenie w bibliotece obiektowej wywołań kodu składowanego we wspólnej bibliotece PL/SQL typu PLL.

Biblioteka PLL ma tą zaletę, że można ją łatwo modyfikować i stanowi pojedynczy punkt wprowadzania zmian.

Najczęściej dobrym pomysłem jest umieszczenie podstawowego wspólnego dla całego systemu kodu w jednej wspólnej bibliotece np. SYSTEM.PLL.

Następnie podczas tworzenia poszczególnych aplikacji wchodzących w skład systemu pojawiają się fragmenty kodu, które również należy współdzielić. W szczególności mogą to być fragmenty kodu sprzeczne ze standardem systemu. Wówczas kod taki umieszczamy w bibliotece specyficznej dla aplikacji np. APPL01.PLL i dołączamy do modułów po bibliotece systemowej. Jeśli jakiś kod w bibliotece aplikacji nadpisuje kod biblioteki systemu nastąpi odpowiednie przysłonięcie wynikające z kolejności dołączenia.

Takie podejście do uwspólnienia kodu wiąże się z wprowadzeniem hierarchii bibliotek PLL.

Wspólna funkcjonalność w bazie danych

Przeniesienie kodu PL/SQL do bazy danych daje jeszcze łatwiejszą możliwość zmiany kodu wspólnego oraz pozwala na implementacje kodu podstawowego współdzielonego przez warstwę serwera i klienta.

Należy pamiętać, że nie zawsze przeniesienie kodu do bazy danych wiąże się ze zwiększeniem wydajności, w praktyce może być odwrotnie.

Przykładem funkcjonalności która doskonale nadaje się do tej warstwy jest kompleksowy mechanizm obsługi błędów albo monitorowania pracy aplikacji. Do generowanego kodu mechanizm taki można dołączyć poprzez preferencje MSGSFT ustawianą na poziomie systemu aplikacji. Designer zawiera prosty przykład takiej obsługi udokumentowany w pomocy elektronicznej pod hasłem „About customizing the CG\$FORM_ERRORS package”.

5 Budowa kodu

Bardzo rzadko zdarza się aby moduły nie zawierały złożonej funkcjonalności wychodzącej poza to co można dostać na podstawie meta – modelu (obsługa wstawień, aktualizacji,

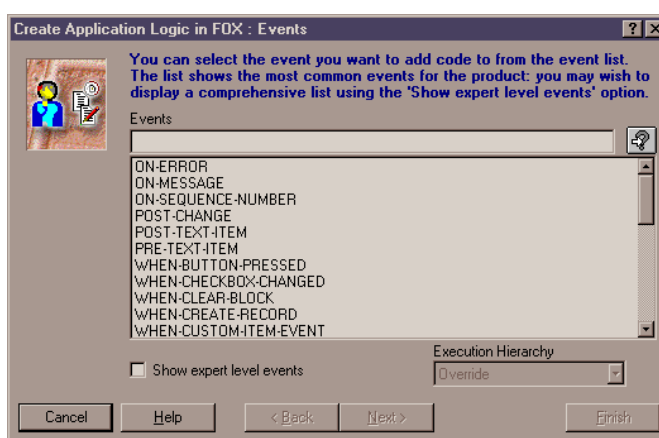
podstawowych walidacji itp.). Wówczas konieczne jest zaimplementowanie brakującego kodu w odpowiednim module.

Istotną cechą Oracle Designera jest możliwość zapisania tego kodu do repozytorium tak, że każda następną generacją polega tylko na uruchomieniu generatora. Żadne zmiany wygenerowanego kodu nie są potrzebne.

Użycie logiki użytkownika składowanej w repozytorium

Logika (kod) użytkownika składowany w repozytorium jest automatycznie umieszczany w procesie generacji w odpowiednich miejscach modułu w zależności od zdefiniowanej w repozytorium akcji. W przypadku Oracle Forms większość standardowych wyzwalaczy ma swój odpowiednik w repozytorium.

Użycie repozytorium bardzo upraszcza sytuację w przypadku konieczności powtórnego generowania modułu na przykład na skutek zmian modelu danych.



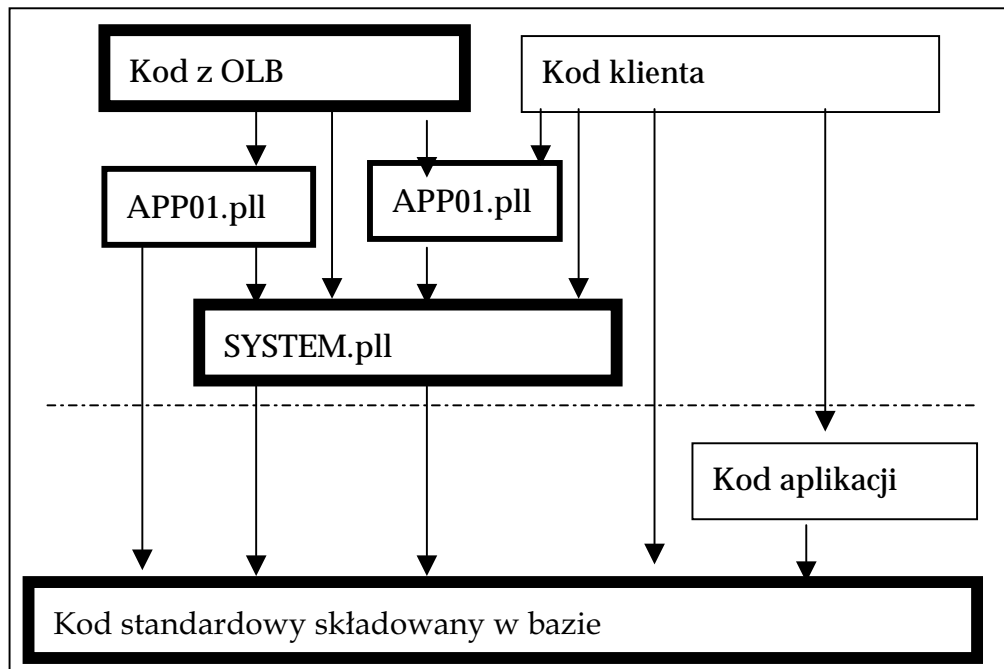
Logika użytkownika składowana w repozytorium powinna być w miarę możliwości umieszczana we współdzielonych bibliotekach lub w warstwie bazy danych, gdyż ułatwia to zarządzaniem takim kodem.

Wielokrotnie używane komponenty składowane w repozytorium i w bibliotekach

Oracle Designer daje możliwość uwspólnienia kodu już na etapie metamodelu. Komponenty modułów (MC) oraz listy wartości (LOV) mogą zostać oznaczone jako obiekty wspólne. Ich użycie polega na umieszczeniu odpowiedniego odwołania w modułach. Modyfikacje wspólnego komponentu są dziedziczone przez wszystkie jego użycia.

Mimo to komponenty nie nadają się najlepiej do umieszczania logiki użytkownika która powinna być składowana w miejscu najłatwiejszym dla utrzymania i szybkości modyfikacji, zatem w bibliotekach oraz w bazie danych.

Schemat położenia kodu aplikacji i standardów.



Generowanie wydajnego kodu

Wydajność aplikacji w dużej mierze zależy od środowiska gdzie aplikacja będzie uruchamiana oraz od charakteru funkcji.

W środowisku klient – serwer i w środowisku WEB będą używane inne techniki strojenia aplikacji.

W przypadku środowiska klient serwer kluczowe są następujące aspekty:

1. Odpowiednie umieszczenie kodu PL/SQL odwołującego się do obiektów bazodanych,
2. Użycie perspektyw i procedur bazodanowych jako źródło danych dla bloków bazodanowych
3. Natomiast w środowisku WEB ważniejsze stają się:
4. Użycie JAR albo CAB zamiast poszczególnych klas przy przesyłaniu kodu Javy na stronę użytkownika,
5. Użycie podobnych obiektów (zmniejsza konieczność przysyłania różnic),
6. Minimalizacja użycia tekstów i obiektów typu boilerplate,
7. Minimalizacja liczby wyzwalaczy – zwłaszcza wyzwalaczy związanych z nawigacją,
8. Użycie Java Beans do obsługi akcji po stronie klienta,
9. Użycie wielu małych modułów zamiast jednego dużego.

Ze względu na mechanizmy wbudowane w Serwer Forms a służące do optymalizacji użycia sieci aplikacje w środowisku WEB mogą działać znacznie bardziej efektywnie niż takie same aplikacje Klient-Serwer.

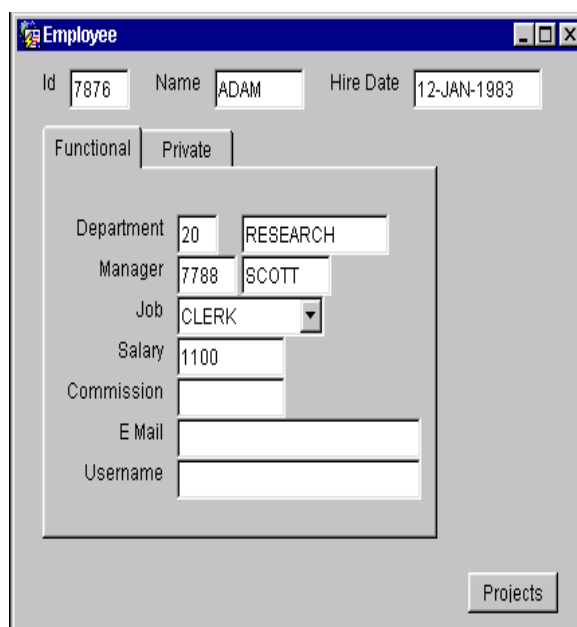
6 Uruchamianie aplikacji

Uruchamianie aplikacji w środowisku WEB

W środowisku WEB nie można używać pewnych funkcji dostępnych w klient serwer jak wyzwalaczy czasowych lub wyzwalaczy związanych z akcjami myszy lub komponentów OLE. W zamian za to pojawiają się nowe możliwości jak komponenty Java Beans czy zmieniony interfejs użytkownika.

Ze względu na implementację interfejsu użytkownika w Javie łatwe są zmiany wyglądu aplikacji przy bez jej modyfikacji. Domyślnie Oracle proponuje tzw. Oracle Look and Feel (tzw. OLAF) który wprowadza wygląd i zachowanie znane z Oracle Enterprise Manager czy Oracle Appliaction.

Poniższy rysunek przedstawia standardowy ekran aplikacji:



The screenshot shows a standard Java Swing window titled "Employee". At the top, there are three text input fields: "Id" with the value "7876", "Name" with the value "ADAM", and "Hire Date" with the value "12-JAN-1983". Below these are two tabs: "Functional" and "Private", with "Private" selected. The main area contains a form with the following fields: "Department" (value "20", dropdown menu showing "RESEARCH"), "Manager" (value "7788", dropdown menu showing "SCOTT"), "Job" (value "CLERK", dropdown menu), "Salary" (value "1100"), "Commission" (empty), "E Mail" (empty), and "Username" (empty). A "Projects" button is located at the bottom right of the window.

a tu ten sam ekran w OLAF:



The screenshot shows the same "Employee" form as above, but rendered using the OLAF (Oracle Look and Feel) theme. The window title bar and the overall appearance of the controls (text boxes, dropdowns, tabs, and button) are styled to match the Oracle Enterprise Manager interface. The data values and layout are identical to the previous screenshot.

istnieje również możliwość zarządzania schematami kolorów:

Zmian takich można dokonać przy pomocy znaczników HTML w pliku uruchamiającym aplikację np.:

```
lookAndFeel="oracle"
darkLook="true"
colorScheme="blue"
```

Zarządzanie innymi aspektami tego interfejsu realizowane jest poprzez plik `$ORACLE_HOME/forms/java/oracle/forms/Registry/Registry.dat`. Można tam ustawiać niestandardowe klasy Javy i czcionki.

Niezależnie interfejs może być rozszerzany w Javie. Można na przykład napisać obsługę pola realizującego dodatkowe sprawdzenia albo specjalne wyświetlanie.

Środowisko UNIX

Często migracja do środowiska WEB wiąże się z przenosinami aplikacji do UNIX. Problemy z taką migracją koncentrują się na:

1. Rozróżnianiu wielkości liter w nazwach plików,
2. Użyciu standardów GIF do wyświetlania liter,
3. Znaki narodowe.

Podczas konwersji należy:

1. Zainstalować i skonfigurować emulator terminala X-Windows
2. Zainstalować na serwerze Unix czcionki w standardzie iso-8859-2.
3. Ustawić w repozytorium nazwy modułów i odwołań do modułów na małe litery,
4. Przegenerować całą aplikację w środowisku Unix (f60genm),
5. Przenieść binarnie bibliotekę obiektową,
6. Przenieść biblioteki PLL w formacie PLT jako pliki tekstowe,
7. Skompilować powstałe biblioteki PLL – ten format zawiera zarówno tekst jak i kod wykonywany. Nie należy używać PLX,
8. Przenosić pliki FMB jako pliki binarne.

Aplikacji raczej nie należy próbować uruchamiać w środowisku Motif ani tekstowym ponieważ mogą pojawić się różnice w wyglądzie jakich nigdy nie będziemy oglądać w środowisku WEB.