Instrukcja do zajęć laboratoryjnych

Bazy danych, ORACLE

wersja 3.0

Nr ćwiczenia:	**3**
Temat:	**Zapoznanie się z przykładowym modelem relacyjnym, projektowanie modeli relacyjnych**
Cel ćwiczenia:	Celem ćwiczenia (podzielonego na dwie części) jest zapoznanie studenta z przykładowym modelem relacyjnym (o nazwie SUMMIT2), oraz nauczenie projektowania „od podstaw" modeli relacyjnych opisujących pewien fragment świata rzeczywistego.
Uwagi:	Dokładne zrozumienie modelu przykładowego SUMMIT2 jest o tyle istotne, że większość ćwiczeń w dalszej części kursu będzie oparta o ten właśnie model.

I. Część pierwsza – model SUMMIT2

Uwaga: podane niżej punkty 1 i 2 przedstawione są jedynie po to, aby studenci mogli na ew. swoich instalacjach baz ćwiczyć z poziomu bezpiecznych (czyli nie `SYS`) kont. W czasie zajęć laboratoryjnych każdy student otrzymał już swoje konto (labxxx) na uczelnianym serwerze Oracle.

1. Zalogować się na użytkownika `SYS`. Utworzyć nowego użytkownika `dblab` z hasłem `lab`. Wykonujemy to wydając z poziomu programu SQL*Plus następujące polecenia:

```
SQL> CREATE USER dblab IDENTIFIED BY lab;
SQL> GRANT CONNECT, RESOURCE TO dblab;
```

   Gdy z jakiegoś powodu trzeba będzie usunąć stworzonego użytkownika należy wydać polecenie:

```
SQL> DROP USER dblab CASCADE;
```

   Uwaga 1: tworzenie i kasowanie użytkowników może wykonywać TYLKO użytkownik z uprawnieniami administratora systemu – standardowo jest to użytkownik `SYS` oraz `SYSTEM`.

   Uwaga 2: każde polecenie języka SQL musi być zakończone średnikiem.

2. Połączyć się do bazy jako użytkownik `dblab` wydając polecenie:

```
SQL> CONNECT dblab/lab;
```

3. Zapoznać się z dostarczonym rysunkiem przykładowej struktury relacyjnej (SUMMIT2). Zrozumieć sens istniejących w nim powiązań relacyjnych. Zapoznać się również z

dostarczonym skryptem `summit2.sql`. Postarać się zrozumieć poszczególne polecenia w nim zawarte oraz odnieść je do rysunku.

Uwaga 1: Dokładnie omówienie poleceń zawartych w skrypcie `summit2.sql` będzie tematem kolejnych ćwiczeń. Nie przejmuj się więc, że wielu z tych poleceń nie rozumiesz, lub tylko domyślasz się ich działania (język SQL – przynajmniej w swej podstawowej postaci – jest bardzo intuicyjny i zbliżony do mówionego języka angielskiego, stąd dość łatwo domyślić się co dane polecenia wykonują).

4. Wykonać skrypt `summit2.sql`. W systemie ORACLE, aby wykonać polecenia zapisane w pliku (zwanym właśnie skryptem) należy w linii poleceń wpisać znak `@` oraz ścieżkę i nazwę pliku (patrz też instrukcja nr 2):

```
SQL> @ścieżka_dostępu\summit2.sql
```

   Uwaga: Skrypt należy wykonać jako użytkownik `dblab`. Jeżeli omyłkowo wykonamy go jako użytkownik `SYS` lub `SYSTEM` to generalnie nic złego się nie stanie, jednak należy dbać o to, aby schematy użytkowników systemowych nie były „zaśmiecane" niepotrzebnymi danymi.

5. Zapoznać się z obiektami, które zostały utworzone w wyniku wykonania skryptu `summit2.sql`. Informacje te można znaleźć w następujących tabelach *słownika bazy danych*:

```
user_tables
user_tab_columns
user_catalog
user_constraints
user_cons_columns
user_indexes
user_ind_columns
user_objects
```

   Opisać maksymalnie trzema zdaniami jakiego typu informacje są zapisane w każdej z powyższych tabel słownikowych. W razie wątpliwości sięgnąć do dokumentacji.

   Przykładowo wydając polecenie:

```
SQL> SELECT table_name FROM user_tables;
```

   otrzymamy wynik:

```
TABLE_NAME
------------------------------
CUSTOMER
DEPT
EMP
IMAGE
INVENTORY
ITEM
LONGTEXT
ORD
PRODUCT
REGION
TITLE
WAREHOUSE

12 rows selected.
```

6. Zapoznać się z zawartością poszczególnych tabel schematu SUMMIT2. Wyznaczyć ponadto ilość rekordów zarejestrowanych w każdej tabeli. Powyższe wykonujemy wydając polecenia:

```
SQL> select * from nazwa_tabeli;
```

oraz

```
SQL> select max(rownum) from nazwa_tabeli;
```

7. Przetłumaczyć nazwy tabel na język polski. Maksymalnie trzema zdaniami opisać zawartość każdej z tabel.

II. Część druga – projektowanie modeli relacyjnych

1. W ćwiczeniu należy zaprojektować relacyjny model danych wg. założeń podanych poniżej. Każde z tych założeń musi być uwzględnione w projektowanym modelu. Efektem końcowym powinien być **rysunek** pokazujący poszczególne **tabele** (wraz z wyszczególnieniem ich kolumn) oraz **powiązania integralnościowe** między nimi.

   Forma graficzna jest w zasadzie dowolna. Poniższy rysunek można potraktować jako wzór. Na rysunku użyto następujących skrótów, które oznaczają: `pk` – *primary key*, `uq` – *unique* (`uq1` oraz `uq2` oznacza, że ograniczenie *unique* posiadają obie kolumny jednocześnie a nie każda z osobna!), `nn` – *not null*, `fk` – *foreign key*, `ck` – *check* (wartości w nawiasie oznaczają listę dopuszczalnych wartości).

2. Przeprowadzić dyskusję podanych niżej założeń. Wskazać ich ew. braki, błędy, niedociągnięcia itp. Zaproponować ich rozbudowę celem zwiększenia funkcjonalności modelu i dostosowanie do realnych potrzeb obsługi akademików na Uniwersytecie Zielonogórskim (należy samodzielnie „zasięgnąć brakujących informacji").

Tabela PRACOWNICY		
Kolumny	**Typ**	**Ograniczenia**
Prac_ID	NUMBER(10)	**pk**
Imię	VARCHAR2(20)	**uq1, nn**
Nazwisko	VARCHAR2(30)	**uq2, nn**
Dział_ID	NUMBER(10)	**fk, nn**
Płeć	CHAR(1)	**ck(M,K)**

Tabela DZIAŁY		
Kolumny	**Typ**	**Ograniczenia**
Dział_ID	NUMBER(10)	**pk**
Nazwa	VARCHAR2(100)	**uq, nn**

<u>Założenia:</u>

Stworzyć strukturę relacyjnej bazy danych do obsługi **akademika**, która będzie umożliwiała:

1. **Gromadzenie danych o studentach wg założeń:**

   - każdy student należy do jednej (i tylko jednej) grupy studenckiej a w danej grupie jest z reguły więcej niż jeden student,
   - zakładamy, że każdy student może studiować na wyłącznie jednym wydziale a na danym wydziale jest zawsze więcej niż jeden student,
   - musi istnieć możliwość wpisania paru słów krótko opisujących daną grupę oraz wydział (np.: „grupa bardzo liczna z przewagą osób starszych", „wydział znajduje się na 5 piętrze Budynku Dydaktycznego" itp.).

2. **Gromadzenie danych o wyposażeniu poszczególnych pokoi w akademiku wg założeń:**

   - w każdym pokoju może być wiele sprzętów różnego rodzaju (np. 3 krzesła drewniane, 2 tapczany itd.) a dana grupa sprzętów (np. krzesła drewniane) może być na wyposażeniu wielu pokoi,
   - każda sztuka sprzętu ma swój unikalny numer inwentarzowy,
   - rejestrujemy również wartość (w zł) konkretnych elementów wyposażenia (przy czym może się np. zdarzyć, że dwa sprzęty należące do tej samej grupy – np. „krzesła drewniane" – mają różną cenę, gdyż jedno z nich jest trwale poplamione).

3. **Gromadzenie danych o rozmieszczeniu studentów w pokojach akademika wg założeń:**

   - każdy student może w danej chwili zajmować tylko jeden pokój,
   - pokoje są wieloosobowe i w związku z tym w danym pokoju może (ale nie musi) mieszkać kilku studentów,
   - musi istnieć możliwość wpisywania różnych uwag dotyczących pokoi (np. zapisano, że w pokoju 506 dnia 20.01.97 zgłoszono usterkę „zamek w drzwiach wejściowych zacina się", w tym samym pokoju w dniu 30.02.97 stwierdzono usterkę zlewozmywaka itp.),
   - cena za miejsce w pokoju jest uzależniona od konkretnego pokoju (np. cena pokoju 102 wynosi 100zł/miesiąc a pokoju 404 120 zł/miesiąc itd.).

Uwaga: Zaznaczyć wszystkie zdefiniowane ograniczenia integralnościowe.

LITERATURA

Concepts (Rozdział 2: Tables and Table Clusters, Rozdział 7: SQL) – oryginalna dokumentacja dołączana do systemu ORACLE

SQL Language Reference – oryginalna dokumentacja dołączana do systemu ORACLE

SQL Language Quick Reference – oryginalna dokumentacja dołączana do systemu ORACLE

Dokumentacja dostępna jest na stronie:
http://www.oracle.com/technetwork/indexes/documentation/index.html