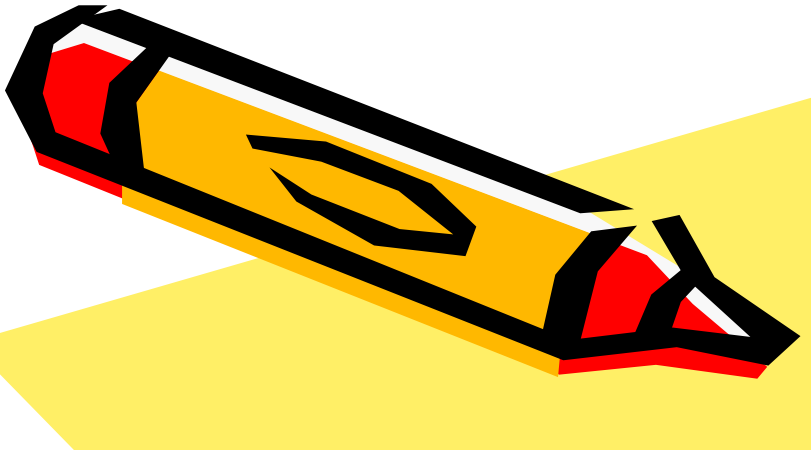
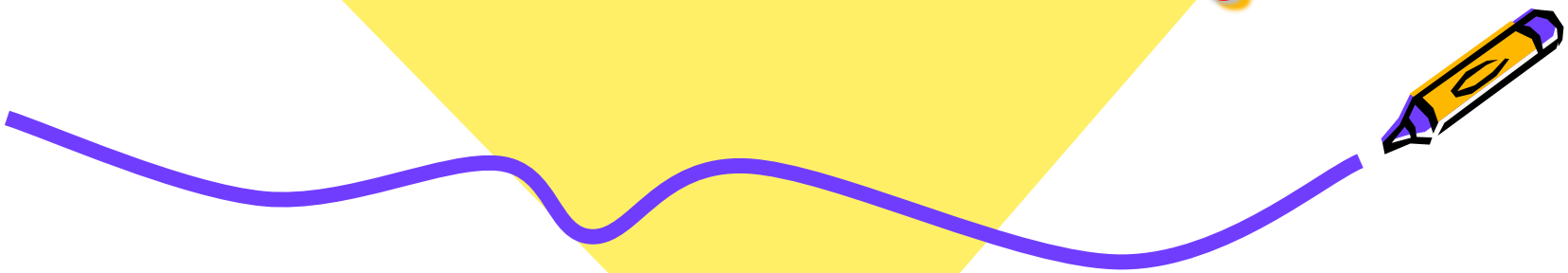


SQL
SECTIUNILE 7 şi 8



Data Manipulation Language (DML)

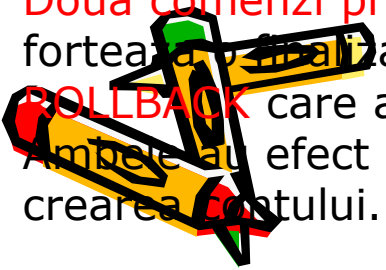
INSERT, UPDATE, DELETE si MERGE

DML statements enable users to make changes in the database.
Executing one DML statement is considered a transaction.

SGBD Oracle se bazeaza pe tranzactii.Unele au Commit automat (finalizare) altele nu .

TRANZACTIE= din punct de vedere al codului sursa, o tranzactie este o unitate logica de lucru continand una sau mai multe comenzi DML sau DDL. Din punct de vedere functional, daca setul de comenzi se executa normal, efectele tranzactiei sunt salvate in baza de date (finalizate- *commit).*Daca cel putin o comanda nu se poate finaliza(din diverse motive), efectele tranzactiei nu sunt salvate in baza de date (anulare- *rollback)* iar tranzactia se incheie.

Doua comenzi principale sunt utilizate in cadrul lucrului tranzactional: COMMIT care forteaza finalizare cu salvare a modificarilor efectuate pana in acel moment si ROLLBACK care anuleaza modificarile efectuate(de asemenea finalizand tranzactia). Ambele au efect in functie de modul cum a fost setata schema proprie, generata la crearea contului.



When does a transaction start or end?

A transaction **begins** with the first DML (INSERT, UPDATE, DELETE or MERGE) statement.

A transaction **ends** when one of the following occurs:

- A COMMIT or ROLLBACK statement is issued
- A DDL(CREATE, ALTER, DROP, RENAME or TRUNCATE) statement is issued
- A DCL(GRANT or REVOKE) statement is issued
- The user exits iSQL*Plus or SQL*Plus
- A machine fails or the system crashes

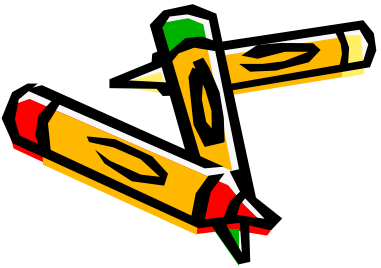
Comenzile DML nu au COMMIT automat
Comenzile DDL, DCL au COMMIT automat
(sunt finalizate automate si ireversibile)

Cand o declaratie DML este finalizata , schimbarile efectuate sunt vizibile la executia oricarei declaratii SELECT.
Daca tranzactia este refacuta, schimbarile sunt anulate

When a DML statement is committed, the change made to the database becomes visible to anyone executing a SELECT statement.

If the transaction is rolled back, the changes are undone:

- The original, older version of the data in the undo segment is written back to the table.
- All users see the database as it existed before the transaction began.



Fiecare dintre dvs. are o schema proprie, schema este o colectie de obiecte, cum ar fi tabele, view (viziuni) si secvente. Numele schemei este o combinatie din tara_scoala_nume.curs_cont.student activa din momentul crearii contului , schema proprie are forma **RO_INST36_SQL02_S01.**

Securitatea bazelor de date poate fi clasificata in 2 categorii: securitatea sistemului si securitatea datelor, pentru aceasta DBA acorda privilegii care permit sau interzic anumite operatii asupra tabelelor.Exista mai mult de 100 de privilegii care pot fi acordate de DBA (creare tabele, stergere tabele, creare rol, spatiu alocat, creare view……).

To keep your schema tables in their original state you will make a copy of each table to complete the practice activities in this and later lessons

Then, if you wrongly alter a copy table, you can restore a correct version of it from the original table.

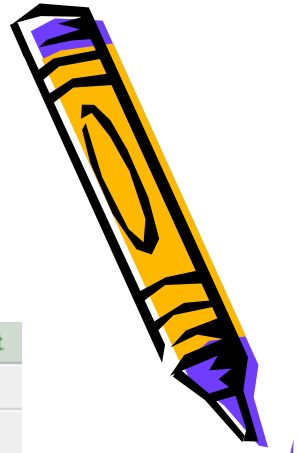
You should name each table copy_tablename. The table copies will not inherit the associated primary-to-foreign-key integrity rules (relationship constraints) of the original tables. The column data types, however, are inherited in the copied tables.

Ca sa pastrezi integritatea originala a tabelelor, vom face copy ale acestora, (in cazul in care doresti sa restaurezi tabela originala nu vor fi probleme)

Copiile tabelelor **NU** vor mosteni constrangerile referitoare la relatii (Regulile de integritate asociate PK-FK) tabelelor originale. Desi tipul de date al coloanelor sunt mostenite in tabela copie.

create table copy_employees
as (select * from employees)

subquery



```
describe copy_employees
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
COPY_EMPLOYEES	EMPLOYEE_ID	Number	-	6	0	-	✓	-	-
	FIRST_NAME	Varchar2	20	-	-	-	✓	-	-
	LAST_NAME	Varchar2	25	-	-	-	-	-	-
	EMAIL	Varchar2	25	-	-	-	-	-	-
	PHONE_NUMBER	Varchar2	20	-	-	-	✓	-	-
	HIRE_DATE	Date	7						

```
describe employees
```

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEES	EMPLOYEE_ID	Number	-	6	0	1	-	-	Primary key of employees table.
	FIRST_NAME	Varchar2	20	-	-	-	✓	-	First name of the employee. A not null column.
	LAST_NAME	Varchar2	25	-	-	-	-	-	Last name of the employee. A not null column.
	EMAIL	Varchar2	25	-	-	-	-	-	Email id of the employee
	PHONE_NUMBER	Varchar2	20	-	-	-	✓	-	Phone number of the employee; includes country c and area code
	HIRE_DATE	Date	7	-	-	-	-	-	Date when the employee started on this job. A not column.
	JOB_ID	Varchar2	10	-	-	-	-	-	Current job of the employee; foreign key to job_id c of the jobs table. A not null column.
	SALARY	Number	-	8	2	-	✓	-	Monthly salary of the employee. Must be greater tha

Tabela copy_employees, va avea aceleasi inregistrari cu tabela originala
se vor pastra doar restrictiile NOT NULL dar **nu** si PK respectiv FK.

INSERT Statement

INSERT

The INSERT statement is used to add new rows to a table. The statement requires three values:

- the name of the table
- the names of the columns in the table to populate
- corresponding values for the column

How can we INSERT the data below to create a new customer in the copy_f_customers table?

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
145	Katie	Hernandez	92 Chico Way	LA	CA	98008	8586667641

```
INSERT INTO copy_f_customers
  (id, first_name, last_name, address, city, state, zip, phone_number)
VALUES
  (145, 'Katie', 'Hernandez', '92 Chico Way', 'Los Angeles', 'CA', 98008, 8586667641);
```

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
145	Katie	Hernandez	92 Chico Way	LA	CA	98008	8586667641

EXPLICIT INSERT

Coloanele sunt enumerate

EXPLICIT

- The INSERT statement is used to add new rows to a table. The statement requires three values:
  - the name of the table
  - the name of the column in the table to populate
  - a corresponding value for the column

EXPLICT INSERT

```
INSERT INTO copy_departments
(department_id, department_name, manager_id,
location_id)
VALUES (70, 'Public Relations', 100, 1700);
```

Another way to insert values in a table is to implicitly add them by omitting the column names.

IMPLICIT

One precaution: the values for each column must match exactly the default order in which they appear in the table (as shown in a DESCRIBE statement), and a value must be provided for each column.

Un alt mod de a insera valori intr-o tabela este de a omite numele coloanelor.
Atentie! Valorile pentru fiecare coloana trebuie sa corespunda exact cu ordinea implicita in care acestea apar in tabela si trebuie furnizata o valoare pentru fiecare coloana

CHECK THE TABLE FIRST

Before inserting data into a table, you must check several table details. The DESCRIBE tablename syntax will return a table summary.

the following information about the columns in the table:

- columns that can have a NULL value
- columns that cannot have duplicate values
- allowable data type
- amount of data that can be entered in a column

INSERT INTO f_customers
VALUES (475, 'Angelina', 'Wright', '7425 Redwood St', 'San Francisco', 'CA', 94162, '4159982010');

IMPLICIT

ID	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NO
475	Angelina	Wright	7425 Redwood St	San Francisco	CA	94162	4159982010

INSERT WITH NULL VALUES

- Naming columns (implicit)

```
INSERT INTO copy_departments (department_id,
department_name )
VALUES (30, 'Purchasing');
```

- Using NULL keyword (explicit)

```
INSERT INTO copy_departments
VALUES (100, 'Finance', NULL, NULL);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
30	Purchasing	(null)	(null)
100	Finance	(null)	(null)

Ca o alternativa de a folosi cuvantul cheie NULL, se pot folosi doua caractere apostrof fara spatiu intre acestea.

Daca in lista de coloane se omite o coloana care are restrictia NOT NULL va aparea o eroare, deoarece pentru coloanele omise sistemul pune implicit NULL

ORA-01400: cannot insert NULL into ("USQA_JOHN_SQL01_S01"."COPY_F_STAFFS"."SALARY")

If a column can hold null values, it can be omitted from the INSERT clause. An implicit insert will automatically insert a null value in that column. To explicitly add null values to a column, use the NULL keyword in the VALUES list for those columns that can hold null values.

INSERTING SPECIAL VALUES

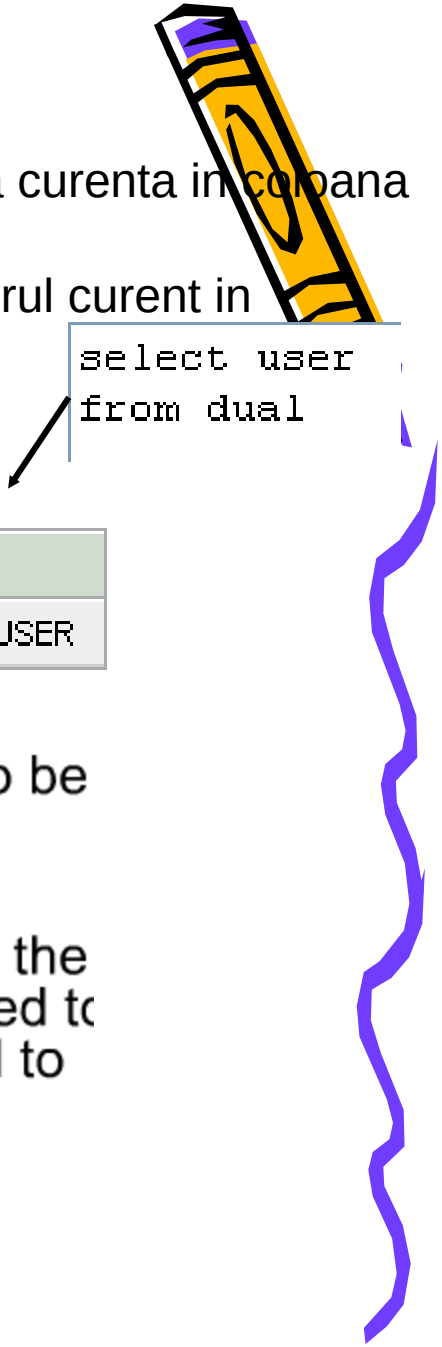
Special values such as SYSDATE and USER can be entered in the VALUES list of an INSERT statement.

- SYSDATE puts current date and time in a column.
- USER places current username (HTML DB will put PUBLIC_USER).

```
INSERT INTO copy_employees (employee_id,
last_name, email,hire_date, job_id)
VALUES ( 1001, USER, 'Test',  SYSDATE,
'IT_PROG');
```

Sysdate pune data curenta in coloana specificata

User pune utilizatorul curent in coloana



In addition, functions and calculated expressions can also be used in the VALUES clause.

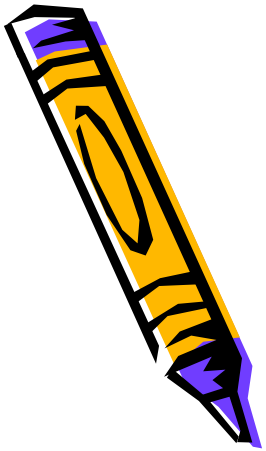
The example below illustrates two types of insertions into the Global Fast Foods copy_f_orders table. SYSDATE is used to insert the order_date and a calculated expression is used to enter the order_total.

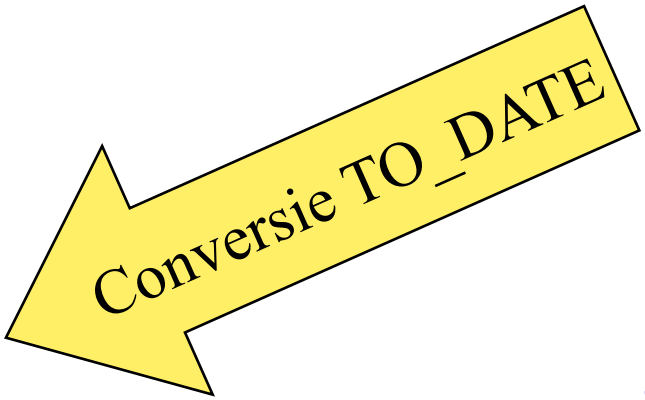
```
INSERT INTO copy_f_orders
  (order_number, order_date, order_total, cust_id, staff_id)
VALUES
  (1889, SYSDATE, 87.92*1.08, 123, 19)
```

INSERTING SPECIFIC DATE VALUES

- The default format date before Oracle9*i* was DD-MON-YY.
- The default format for Oracle9*i* is DD-MON-RR
  - century defaults to the current century
  - default time of midnight (00:00:00
  - formats other than the default format use TO_DATE function.

```
INSERT INTO copy_employees
VALUES (114, 'Den', 'Raphealy', 'DRAPHEAL',
  '515.127.4561', '03-FEB-49', 'AC_ACCOUNT',
  11000, NULL, 100, 30)
```






```
INSERT INTO f_staffs
   (first_name, TO_DATE(birthdate, 'Month fmDD, RRRR')
VALUES
   ('Sue', 'July 1, 1980');
```

FIRST_NAME	TO_DATE(BIRTHDATE, 'MONTHFMDD', RRRR')
Sue	01-JUL-1980 00:00:00

```
INSERT INTO f_staffs
   (first_name, TO_DATE(birthdate, 'Month fmDD, RRRR HH24:MI')
VALUES
   ('Sue', 'July 1, 1980 17:20');
```

FIRST_NAME	TO_DATE(BIRTHDATE, 'MONTHFMDD', RRRR HH24:MI')
Sue	01-JUL-1980 17:20:00

USING A SUBQUERY TO COPY ROWS

Each INSERT statement we have seen so far adds only one row to the table. But suppose we want to copy 100 rows from one table to another ?

We do not want to have to write and execute 100 INSERT statements, one after the other. That would be very time-consuming.

Fortunately, SQL allows us to use a subquery within an INSERT statement. All the results from the subquery are inserted into the table. So we can copy 100 rows – or 1000 rows – with one multiple-row subquery within the INSERT.

As you would expect, you don't need a VALUES clause when using a subquery to copy rows, because the inserted values will be exactly the values returned by the subquery.

O declaratie INSERT permite introducerea unei singure linii. Daca doresti sa copiezi mai multe linii din alte tabele foloseste subcererea

A SUBQUERY TO COPY ROWS

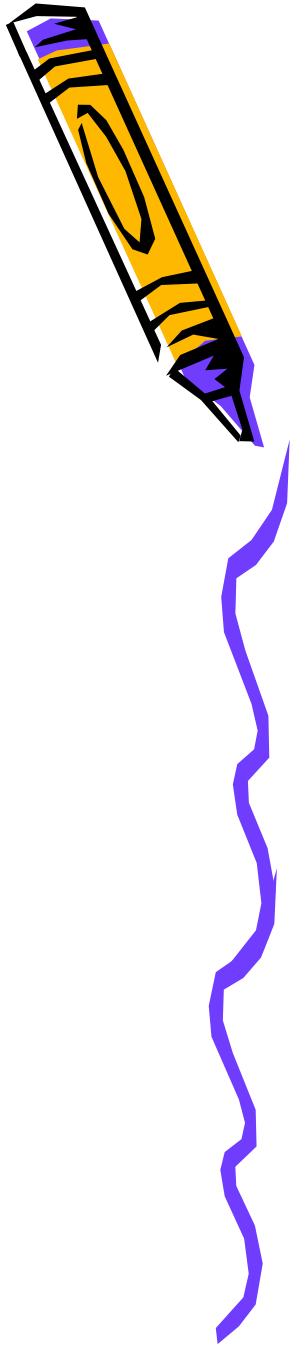
- Copy values from an existing table
- No VALUES clause

```
INSERT INTO sales_reps(id, name, salary,
commission_pct)
  SELECT employee_id, last_name, salary,
   commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';
```

Atentie, clauza VALUES **nu** mai exista

Acest exemplu nu se va executa cu succes in Aplicatia Express, deoarece nu exista tabela sales_reps

Obs: subcererea nu este inclusa in paranteze

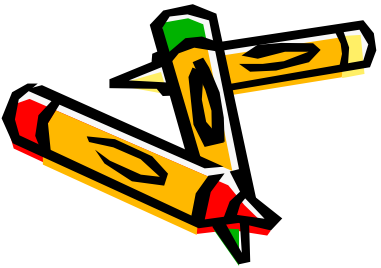


If we want to copy all the data – all rows and all columns – the syntax is even simpler.

To select all rows from the EMPLOYEES table and insert them into the SALES_REPS table, the statement would be written as shown:

```
INSERT INTO sales_reps
   SELECT *   FROM   employees;
```

Again, this will work only if both tables have the same number of columns, in the same order, with the same data types.



TRY IT / SOLVE IT

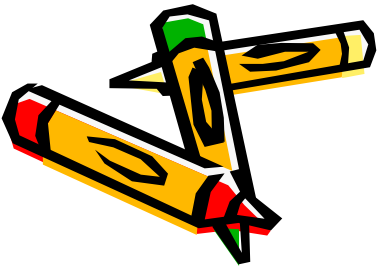
- 1. DJs on Demand just purchased four new CDs. Use an explicit INSERT statement to add each CD to the copy_d_cds table. After completing the entries, execute a SELECT * statement to verify your work.

Problem 1

CD_NUMBER	TITLE	PRODUCER	YEAR
97	Celebrate the Day	R&B Inc.	2003
98	Holiday Tunes for All Ages	Tunes Are Us	2004
99	Party Music	Old Town Records	2004
100	Best of Rock and Roll	Old Town Records	2004

INSERT INTO copy_d_cds(cd_number, title, producer, year)
VALUES (97, 'Celebrate The Day', 'R&B Inc.', 2003);

To verify the entry, SELECT* FROM copy_d_cds/



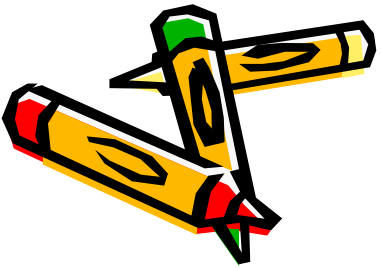
- 2. DJs on Demand has two new events coming up. One event is a fall football party and the other event is a sixties theme party. The DJs on Demand clients requested the songs shown in the table for their events. Add these songs to the copy_d_songs table using an implicit INSERT statement.

TRY IT / SOLVE IT

Problem 2

ID	TITLE	DURATION	TYPE_CODE
52	Surfing Summer	not known	12
53	Victory Victory	5 min	12

```
INSERT INTO copy_d_songs
VALUES(52, 'Surfing Summer', null, null, 12);
```



TRY IT / SOLVE IT

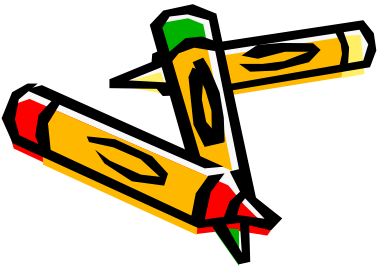
- 3. Add the two new clients to the copy_d_clients table. Use either an implicit or an explicit INSERT.

Problem 3

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
6655	Ayako	Dahish	3608859030	dahisha@harbor.net
6689	Nick	Neuville	9048953049	nnicky@charter.net

INSERT INTO copy_d_clients

VALUES(6656, ‘Sanda', ‘Popescu', 0722407712,‘sandap@yahoo.com');



TRY IT / SOLVE IT

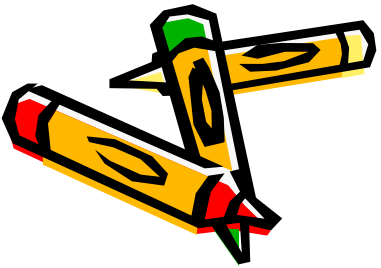
- 4. Add the new client's events to the copy_d_events table. The cost of each event has not been determined at this date.

Problem 4

ID	NAME	EVENT_DATE	DESCRIPTION	COST	VENUE_ID	PACKAGE_CODE	THEME_CODE	CLIENT_NUMBER
110	Ayako Anniversary	07-JUL-04	Party for 50, sixties dress, decorations		245	79	240	6655
115	Neuville Sports Banquet	09-SEP-04	Barbecue at residence, college alumni, 100 people		315	87	340	6689

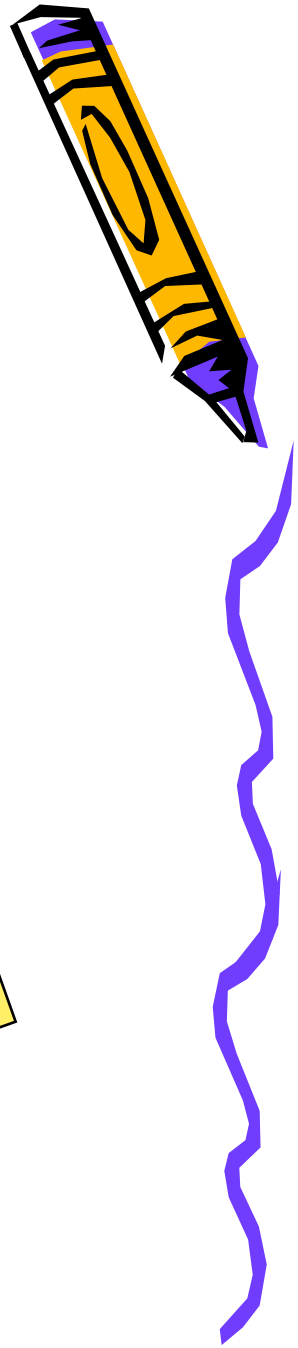
The COST column is mandatory, but the cost is not known at the time of insert. Zero (0) will have to be inserted as the default cost. Demonstrate inserting a new events using:

```
INSERT INTO copy_d_events (ID, NAME, EVENT_DATE, DESCRIPTION, COST,
VALUE_ID, PACKAGE_CODE, THEME_CODE, CLIENT_NUMBER)
VALUES(110, 'Ayako Anniversary', TO_DATE('07-JUL-04','DD-MON-RR'), 'Party
for 50, sixties dress, decorations', NULL, 0, 245,79,240,6655);
```



UPDATE statements

TRY IT / SOLVE IT



- The UPDATE statement is used to modify existing rows in a table. It requires four values:
  - the name of the table
  - the name of the column in the table to populate
  - a corresponding value or subquery for the column
  - a condition that identifies the columns and the changes for each column

MODIFICARE

UPDATING ONE COLUMN

- Specific row or rows are modified if you specify the WHERE clause.

```
UPDATE copy_employees
SET department_id = 70
WHERE employee_id = 113;
```

(One row is updated)

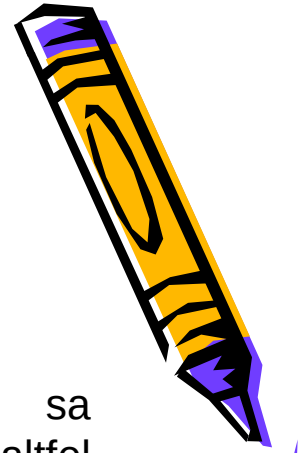
- All rows in the table are modified if you omit the WHERE clause.

```
UPDATE copy_employees
SET department_id = 110
```

(All rows are updated)

Atentie la WHERE

UPDATING COLUMNS WITH SUBQUERIES



- You can update one or more columns in the SET clause of an UPDATE statement by writing subqueries.

```
UPDATE copy_employees
  SET job_id = (SELECT job_id
                FROM employees
                WHERE employee_id = 205),
      salary = (SELECT salary
                FROM employees
                WHERE employee_id = 205)
  WHERE employee_id = 114;
```

Mai multe coloane vor fi modificate, separate printr-o virgula.

Atentie! Subcererile trebuie sa returneze o singura valoare, altfel se va genera eroare!
(single row subquery)

INTEGRITY CONSTRAINT ERRORS

```
UPDATE employees
SET department_id = 55
WHERE department_id = 110;
```

ORA-02291: integrity constraint (USCA_INST_SQL03_T01.EMP_DEPT_FK) violated - parent key not found

Cand adăugăm date noi in tabelă sau modificăm datele existente, integritatea constrângerilor trebuie sa fie respectate. In acest exemplu, in tabela employees coloana department_id este FK care face legatura cu coloana department_id care este PK pentru tabela DEPARTMENTS. Incercarea de a modifica department_id esueaza, deoarece tabela DEPARMENTS nu are nici un rand pentru care department_id este 55. Mesajul de eroare spune exact tabela in care a fost incalcata constangerea.

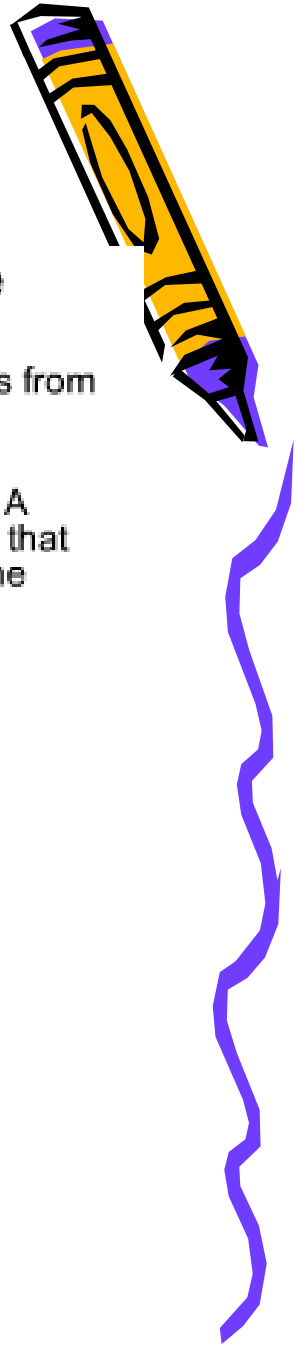
DELETE statement

The DELETE statement is used to remove existing rows in a table. The statement requires two values:

– the name of the table

– the condition that identifies the rov deleted

```
DELETE
FROM copy_employees
WHERE employee_id=200
```



Deleting Rows Based On Another Table

The subquery capability can be used to remove rows from one table based on values from another table.

What if a business decided to close an entire plant? A subquery could be used to identify the employees in that plant and use that information to delete them from the employees table.

```
DELETE FROM emp
WHERE  plant_id =
            (SELECT plant_id
             FROM   locations
             WHERE  plant_loc = 'Anyvilla');
```

SUBQUERY DELETE

- Use subqueries in DELETE statements to remove rows from a table based on values from another table.

```
DELETE FROM copy_employees
WHERE  department_id =
        (SELECT department_id
        FROM   departments
        WHERE  department_name LIKE
        '%Sal%');
```

Subquery devine parte a clauzei WHERE

INTEGRITY CONSTRAINT ERRORS

Integrity constraints ensure that the data conforms to a needed set of rules. The constraints are automatically checked whenever a DML statement which could break the rules is executed. If the any rule would be broken, the table is not updated and an error is returned.

This example violates a NOT NULL constraint, because first_name has a not null constraint and id=123 does not exist, so the subquery returns a null result.

```
UPDATE copy_f_staffs
SET first_name = (SELECT first_name
                  FROM copy_f_staffs
                  WHERE id = 123);
```

ORA-01407: cannot update ("USQA_JOHN_SQL01_S01"."COPY_F_STAFFS"."FIRST_NAME") to NULL

When modifying your copy tables (for example copy_f_customers) you may see not null constraint errors, but you will not see any primary key – foreign key constraint errors.

This is because the CREATE TABLE …. AS (SELECT …) statement used to create the copy tables copies the rows to the copy table and copies the not null constraints, but does **not** copy primary key – foreign key constraints.

Therefore at present there are no primary key – foreign key constraints on the copy tables.

Later in the course you will learn how to add these constraints.

When will primary key - foreign key constraints be checked?

The EMPLOYEES table has a foreign key constraint on department_id, which references the department_id of the DEPARTMENTS table. This ensures that every employee belongs to a valid department.

In the DEPARTMENTS table, department_ids 10 and 20 exist but 15 does not.

Which of the following statements will return an error?

1. UPDATE employees SET department_id = 15
   WHERE employee_id = 100;
2. DELETE FROM departments WHERE department_id = 10;
3. UPDATE employees SET department_id = 10
   WHERE department_id = 20;

1. Nu exista departament =15
2. Nu putem sterge din tabela parinte o linie care e FK in alta tabela
3. Va modifica toate liniile care sunt din departamentul 20 cu departamentul 10

UPDATING Column Values and DELETING Rows

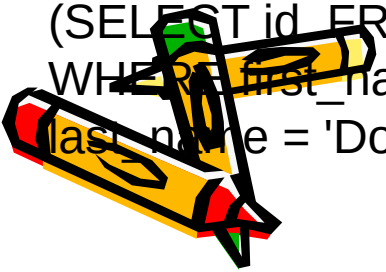
TRY IT/ SOLVE IT

1. Monique Tuttle, the manager of Global Fast Foods, sent a memo requesting an immediate change in prices. The price for a strawberry shake will be raised from $3.59 to $3.75, and the price for fries will increase to $1.20. Make these changes to the copy_f_food_items table.

```
1. UPDATE copy_f_food_items
SET price = 3.75
WHERE description = 'Strawberry Shake';

UPDATE copy_f_food_items
SET price =1.20
WHERE description = 'Fries';
```

2. Bob Miller and Sue Doe have been outstanding employees at Global Fast Foods. Management has decided to reward them by increasing their overtime pay. Bob Miller will receive an additional $0.75 per hour and Sue Doe will receive an additional $0.85 per hour. Update the copy_f_staffs table to show these new values. (Note: Bob Miller currently doesn't get overtime pay. What function do you need to use to convert a null value to 0?)

```
2. UPDATE copy_f_staffs
SET overtime_rate =
NVL(overtime_rate,0) + .75
WHERE id =
(SELECT id  FROM copy_f_staffs
WHERE first_name = 'Bob' and
last_name = 'Miller');

UPDATE copy_f_staffs
SET overtime_rate = overtime_rate + .85
WHERE id =
(SELECT id FROM copy_f_staffs
WHERE first_name = 'Sue' and
last_name = 'Doe');
```



OR

```
UPDATE copy_f_staffs
SET overtime_rate =
NVL(overtime_rate,0) + 0.75
WHERE last_name = 'Miller' AND
first_name = 'Bob';

UPDATE copy_f_staffs
SET overtime_rate =
overtime_rate + .85
WHERE last_name = 'Doe'
AND first_name = 'Sue';
```

TRY IT / SOLVE IT

TRY IT / SOLVE IT

3. Add the orders shown to the Global Fast Foods copy_f_orders table:

Problem 3

ORDER_NUMBER	ORDER_DATE	ORDER_TOTAL	CUST_ID	STAFF_ID
5680	June 23, 2004	159.78	145	9
5691	09-23-04	145.98	225	12
5701	July 4, 2004	229.31	230	12

3. INSERT INTO copy_f_orders(order_number, order_date, order_total, cust_id, staff_id)
VALUES (5680,TO_DATE('June 23, 2004', 'Month DD, YYYY'),159.78, 145, 9);

INSERT INTO copy_f_orders(order_number, order_date, order_total, cust_id, staff_id)
VALUES (5691,TO_DATE('09-23-04', 'MM-DD-YY'),145.98, 225, 12);

INSERT INTO copy_f_orders(order_number, order_date, order_total, cust_id, staff_id)
VALUES (5701,TO_DATE('July 4, 2004', 'Month DD, YYYY'),229.31, 230, 12);

- 4. Add the new customers shown at right to the copy_f_customers table. You may already have added Katie Hernandez. Will you be able to add all these records successfully?

TRY IT / SOLVE IT

Problem 4

Id	FIRST_NAME	LAST_NAME	ADDRESS	CITY	STATE	ZIP	PHONE_NUMBER
145	Katie	Hernandez	92 Chico Way	Los Angeles	CA	98008	8586667641
225	Daniel	Spode	1923 Silverado Street	Denver	CO	80219	7193343523
230	Adam	Zum	5 Admiral Way	Seattle	WA		4256879009

INSERT INTO copy_f_customers(id, first_name, last_name, address, city, state, zip, phone_number)
VALUES (145, 'Katie', 'Hernandez', '92 Chico Way', 'Los Angeles', 'CA', 98008, 8586667641)

INSERT INTO copy_f_customers(id, first_name, last_name, address, city, state, zip, phone_number)
VALUES (225, 'Daniel', 'Spode', '1923 Silverado Street', 'Denver', 'CO', 98107, 4256879009)

You cannot insert the Adam Zum because the zip cannot be null.

•5. Sue Doe has been an outstanding Global Foods staff member and has been given a salary raise. She will now be paid the same as Bob Miller. Update her record in copy_f_staffs.

TRY IT / SOLVE IT

```
UPDATE copy_f_staffs
SET salary =
        (SELECT salary   FROM copy_f_staffs
        WHERE first_name = 'Bob' and last_name = 'Miller')
WHERE first_name = 'Sue' and last_name ='Doe';
```

Object Type **TABLE** Object **F_STAFFS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
F_STAFFS	ID	Number	-	5	0	1	-	-	-
	FIRST_NAME	Varchar2	25	-	-	-	-	-	-
	LAST_NAME	Varchar2	35	-	-	-	-	-	-
	BIRTHDATE	Date	7	-	-	-	-	-	-
	SALARY	Number	-	8	2	-	-	-	-
	OVERTIME_RATE	Number	-	5	2	-	✓	-	-
	TRAINING	Varchar2	50	-	-	-	✓	-	-
	STAFF_TYPE	Varchar2	20	-	-	-	-	-	-
	MANAGER_ID	Number	-	5	0	-	✓	-	-
	MANAGER_BUDGET	Number	-	8	2	-	✓	-	-
	MANAGER_TARGET	Number	-	8	2	-	✓	-	-

TRY IT / SOLVE IT

6. Global Fast Foods is expanding their staff. The manager, Monique Tuttle, has hired Kai Kim. Not all information is available at this time, but add the information shown at right.

Problem 6

ID	FIRST_NAME	LAST_NAME	BIRTHDATE	SALARY	STAFF TYPE
25	Kai	Kim	3-NOV-88	6.75	Order Taker

6. INSERT INTO copy_f_staffs(id, first_name, last_name, birthdate, salary, overtime_rate, training, staff_type, manager_id, manager_budget, manager_target)
VALUES (25, 'Kai', 'Kim', '03-NOV-88', 6.75, null, null, 'Order Taker', null, null, null)

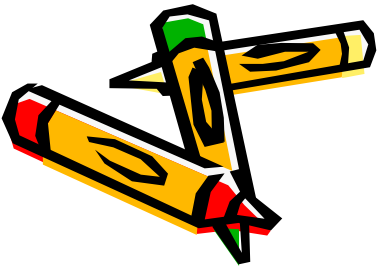
Object Type **TABLE** Object **F_STAFFS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
F_STAFFS	ID	Number	-	5	0	1	-	-	-
	FIRST_NAME	Varchar2	25	-	-	-	-	-	-
	LAST_NAME	Varchar2	35	-	-	-	-	-	-
	BIRTHDATE	Date	7	-	-	-	-	-	-
	SALARY	Number	-	8	2	-	-	-	-
	OVERTIME_RATE	Number	-	5	2	-	✓	-	-
	TRAINING	Varchar2	50	-	-	-	✓	-	-
	STAFF_TYPE	Varchar2	20	-	-	-	-	-	-
	MANAGER_ID	Number	-	5	0	-	✓	-	-
	MANAGER_BUDGET	Number	-	8	2	-	✓	-	-
	MANAGER_TARGET	Number	-	8	2	-	✓	-	-

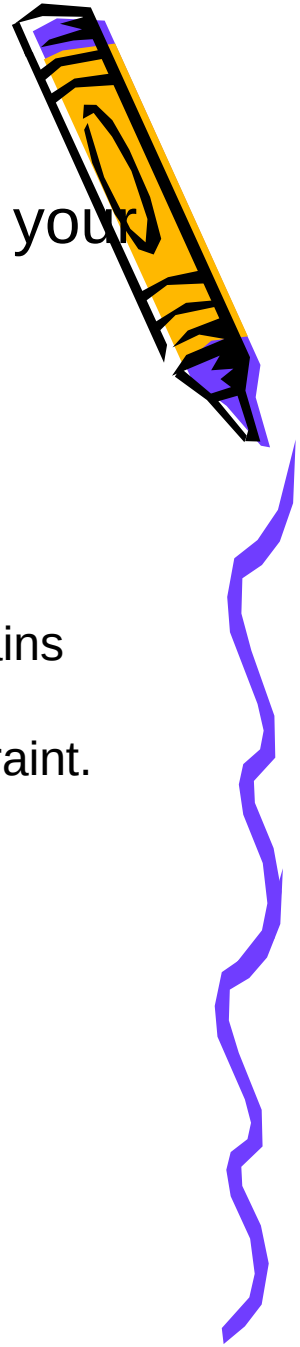
TRY IT / SOLVE IT

- 7. Now that all the information is available for Kai Kim, update his Global Fast Foods record to include the following: Kai will have the same manager as Sue Doe. He does not qualify for overtime. Leave the values for training, manager budget, and manager target as null.

7. UPDATE copy_f_staffs
SET manager_id = (SELECT manager_id
FROM copy_f_staffs
WHERE first_name = 'Sue' AND last_name = 'Doe')
WHERE ID = 25;



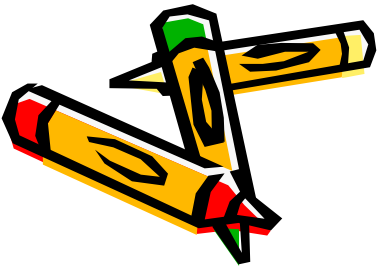
TRY IT / SOLVE IT



8. Execute the following SQL statement. Record your results.

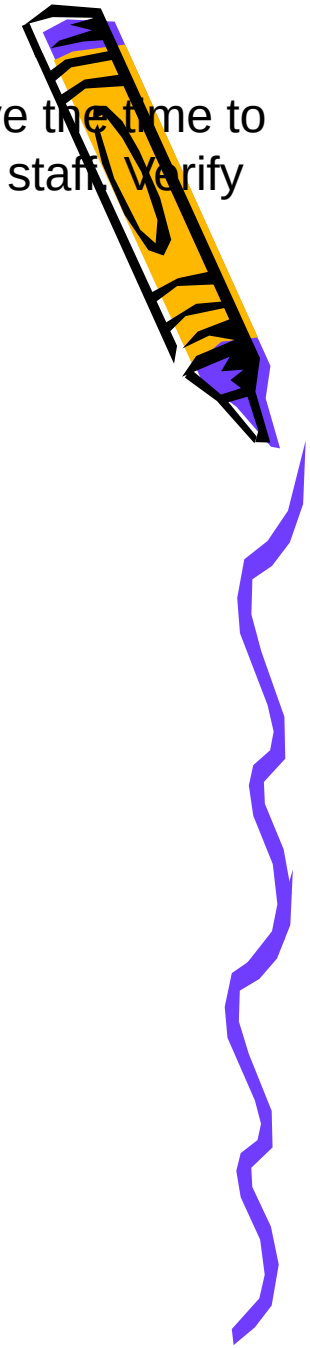
```
DELETE from departments
WHERE department_id = 60;
```

8. Department 60 cannot be deleted because it contains a primary key that is used as a foreign key in another table. Explain: This is a violation of an integrity constraint.



TRY IT / SOLVE IT

9. Kim Kai has decided to go back to college and does not have the time to work and go to school. Delete him from the Global Fast Foods staff. Verify that the change was made.

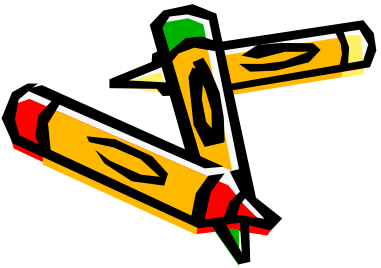


9. DELETE from copy_f_staffs
WHERE ID = 25;

OR

DELETE FROM copy_f_staffs
WHERE last_name = 'Kai'
AND first_name = 'Kim'

SELECT *
FROM copy_f_staffs;



DEFAULT Values

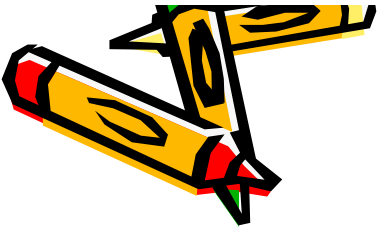
- A column in a table can be given a default value.
- Assigning default values prevents null values from existing in the column.
- Default values can be a literal value, an expression, or a SQL function, such as SYSDATE or USER
- Default values must match the data type of the column

Unei coloane I se poate da o valoare implicita, in momentul in care tabela este creata sau modificata. Aceasta poate fi: o constanta, o expresie sau o functie SQL (SYSDATE, USER). Valorile implicite trebuie sa fie de acelasi tip de date ca si tipul de date al coloanei

Default Values – Example

- **Default values are specified at the time the table is created:**

```
CREATE TABLE items(
 part_number VARCHAR2(10),
 description VARCHAR2(10),
 qty_on_hand NUMBER DEFAULT 0)
```



- Use DEFAULT when inserting values:

```
INSERT INTO items (part_number, description, qty_on_hand)
VALUES (300, 'Widget', DEFAULT)
```

- Use DEFAULT when updating values:

```
UPDATE items
SET qty_on_hand = DEFAULT
WHERE part_number = 200
```

- Now check the results!

```
SELECT *
FROM items
```

MERGE

Using the MERGE statement accomplishes two tasks at the same time. MERGE will INSERT and UPDATE simultaneously. If a value is missing, a new one is inserted. If a value exists, but needs to be changed, MERGE will update it.

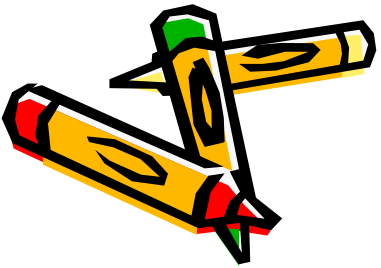
To perform these kinds of changes to database tables, you need to have INSERT and UPDATE privileges on the target table and SELECT privileges on the source table.

Aliases can be used with the MERGE statement.

Rezolva 2 sarcini in acelasi timp: INSERT si UPDATE

MERGE Statements

- MERGE will INSERT and UPDATE at the same time!
- If a value does not exist, one will be added
- If a value does exist, it will be updated



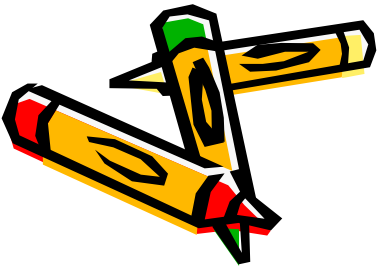
MERGE SYNTAX

MERGE INTO destination-table USING source-table
ON matching-condition
WHEN MATCHED THEN UPDATE
SET ……
WHEN NOT MATCHED THEN INSERT
VALUES (……);

One row at a time is read from the source table, and its column values are compared with rows in the destination table using the matching condition.

If a matching row exists in the destination table, the source row is used to update column(s) in the matching destination row.

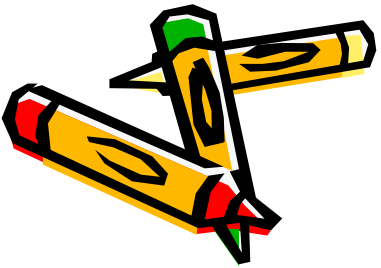
If a matching row does not exist, values from the source row are used to insert a new row into the destination table.



MERGE se uita la doua tabele. Compara datele din cele doua tabele si le corecteaza daca este necesar. MERGE este de asemenea folosita cand doua baze de date sunt combinate

MERGE EXAMPLE:

```
MERGE INTO copy_items c USING items i ON(
 c.part_number = i.part_number)
 WHEN MATCHED THEN UPDATE SET
 c.description = i.description,
 c.qty_on_hand = i.qty_on_hand
 WHEN NOT MATCHED THEN INSERT VALUES
 (i.part_number, i.description, i.qty_on_hand)
```



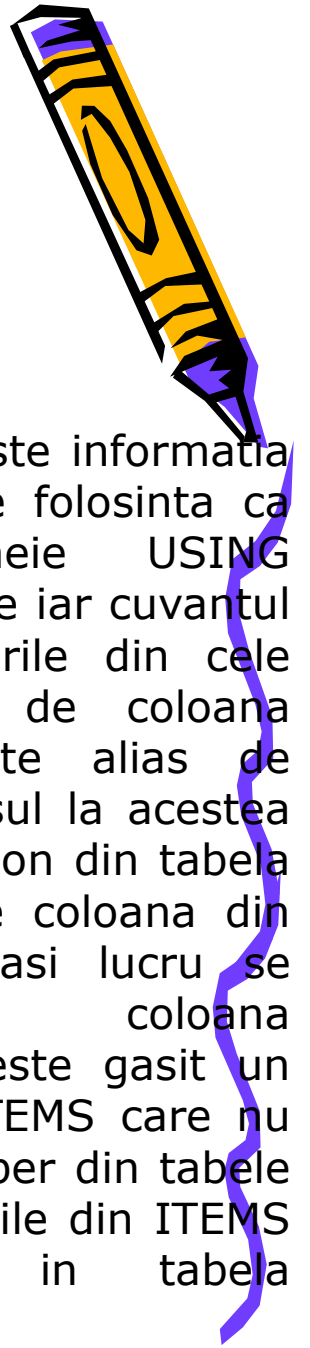


Tabela COPY_ITEMS primeste informatia noua iar tabela ITEM este folosinta ca referinta. Cuvantul cheie USING conecteaza cele doua tabele iar cuvantul cheie ON compara randurile din cele doua tabele in functie de coloana part_number. Se foloseste alias de tabela pentru a face accesul la acestea mai usor. Coloana description din tabela ITEMS va fi copiata peste coloana din tabela COPY_ITEMS. Acelasi lucru se intampla si cu coloana quantity_on_hand. Cand este gasit un part_number din tabele ITEMS care nu se potriveste cu part_number din tabele COPY_ITEMS atunci randurile din ITEMS se copiaza intocmai in tabela COPY_ITEMS.

```
MERGE INTO copy_emp  c   USING employees e
  ON (c.employee_id = e.employee_id)
 WHEN MATCHED THEN  UPDATE
    SET
        c.last_name   = e.last_name,
       c.department_id   = e.department_id
WHEN NOT MATCHED THEN INSERT
    VALUES   (e.employee_id, e.last_name,
e.department_id);
```

EMPLOYEES rows 100 and 103 have matching rows in COPY_EMP, and so the matching COPY_EMP rows were updated.

EMPLOYEE 142 had no matching row, and so was inserted into COPY_EMP.

EMPLOYEES (source table)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

COPY_EMP before the MERGE is executed

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	Smith	40
103	Chang	30

COPY_EMP after the MERGE has executed

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

TRY IT / SOLVE IT

- 1. Currently, the Global Foods F_PROMOTIONAL_MENUS table START_DATE column does not have SYSDATE set as DEFAULT. Your manager has decided she would like to be able to set the starting date of promotions to the current day for some entries. This will require three steps:
- a. In your schema, Make a copy of the Global Foods F_PROMOTIONAL_MENUS table using the following SQL statement:

```
CREATE TABLE copy_f_promotional_menus
AS (SELECT * FROM f_promotional_menus)
```

- b. Alter the current START_DATE column attributes using:

```
ALTER TABLE copy_f_promotional_menus
MODIFY(start_date DATE DEFAULT SYSDATE)
```

- c. INSERT the new information and check to verify the results.
- INSERT a new row into the copy_f_promotional_menus table for the manager's new promotion. The promotion code is 120. The name of the promotion is 'New Customer.' Enter DEFAULT for the start date and '01-JUN-05' for the ending date. The giveaway is a 10% discount coupon.

```
INSERT INTO copy_f_promotional_menus (code, name,
start_date, end_date, give_away)
VALUES( 120, 'New Customer', DEFAULT, '01-JUN-05', '10%
discount coupon')
```

2. Allison Plumb, the event planning manager for DJs on Demand, has just given you the following list of CDs she acquired from a company going out of business. She wants a new updated list of CDs in inventory in an hour, but she doesn't want the original D_CDS table changed. Prepare an updated inventory list just for her.

- a. Assign new cd_numbers to each new CD acquired.
  - b Create a copy of the D_CDS table called manager_copy_d_cds

```
CREATE TABLE manager_copy_d_cds
AS (SELECT * FROM d_cds)
```

c. INSERT into the manager_copy_d_cds table each new CD title using an INSERT statement. Make up one example or use this data:
120, 'Hello World Here I Am', 'Middle Earth Records', '1998'

```
INSERT INTO copy_d_cds (cd_number, title, producer, year)
VALUES( 120, 'Hello World Here I Am', 'Middle Earth Records', '1998')
```

d. Use a merge statement to add to the manager_copy_d_cds table, the CDs from the original table. If there is a match, update title and the year. If not, insert the data from the original table.

```
MERGE INTO manager_copy_d_cds c USING d_cds d
        ON (c.cd_number = d.cd_number)
    WHEN MATCHED THEN UPDATE
                        SET
                        c.year = d.year, c.title = d.title,
                        c.producer = d.producer
    WHEN NOT MATCHED THEN INSERT
            VALUES (d.cd_number, d.title, d.producer, d.year);
```

TRY IT / SOLVE IT

1. Create copies of the following Oracle database tables and name them as specified:
- employees copied as o_employees
- departments copied as o_departments
- jobs copied as o_jobs

2. As the DBA for Oracle, you have been asked to update the database with new information.

Last month, O created a new department in Seattle called Human Resources. The department was assigned ID 210. Employees in this department have the job title Human Resources Manager, and job ID HR_MAN. The salary for all new employees in this department ranges from a minimum of $4500 to a maximum of $5500.

Add the Human Resources job to the o_jobs table.

1. Create the three o_tables, jobs, employees and departments using the syntax:

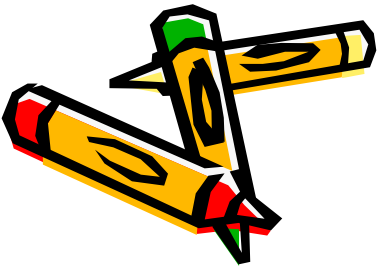
```
CREATE TABLE o_jobs
AS (SELECT * FROM jobs);
```

2. Add the Human Resources job to the jobs table:

```
INSERT INTO o_jobs (job_id, job_title, min_salary, max_salary)
VALUES('HR_MAN', 'Human Resources Manager', 4500, 5500);
```

Don't accept this assignment until the correct output is produced. Students should have added three employees to the o_employees table, added the 'HR_MAN' job to the o_jobs table, and added the 'Human Resources' department to the o_departments table. Verify that students were successful in producing the correct output (ask them to print or show you their output)

Changes to these tables will be done in Lesson 3.



TRY IT / SOLVE IT

3. Three new employees hired for this department are shown in the graphic. Add them to the o_employees table.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
210	Ramon	Sanchez
211	Tai	Sugita
212	Alina	Arcos

Each employee will need an email address created from the first letter of the employee's first name combined with the employee's last name (Bob Smith would be BSMITH). Use the current date as the hire date.

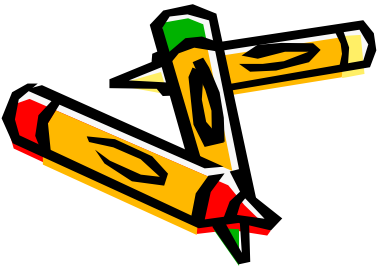
4. Add Human Resources to the o_departments table.

3.

```
INSERT INTO o_employees (employee_id, first_name, last_name, email, hire_date, job_id)
VALUES(210, 'Ramon', 'Sanchez', 'RSANCHEZ', SYSDATE, 'HR_MAN');
INSERT INTO o_employees (employee_id, first_name, last_name, email, hire_date, job_id)
VALUES(211, 'Tai', 'Sugita', 'TSUGITA', SYSDATE, 'HR_MAN');
INSERT INTO o_employees (employee_id, first_name, last_name, email, hire_date, job_id)
VALUES(212, 'Alina', 'Arcos', 'AARCOS', SYSDATE, 'HR_MAN');
```

4.

```
INSERT INTO o_departments(department_id, department_name)
VALUES (210,'Human Resources');
```



TRY IT / SOLVE IT

Several changes need to be made to the o_employees, o_departments, and o_jobs tables.

- Ramon Sanchez has a new phone number: 360-509-7132.

- The location for the Seattle Human Resources department is 1700.

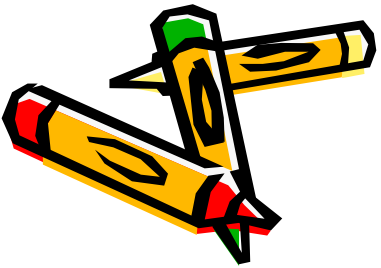
- Ramon Sanchez, Tai Sugita, and Alina Arcos need their department ID updated to 210 if it has not been done as yet.

- Ramon's salary is $5000; Tai earns $5100.

- Delete Alina Arcos, who has decided to return to college to finish her sociology degree.

Answers:

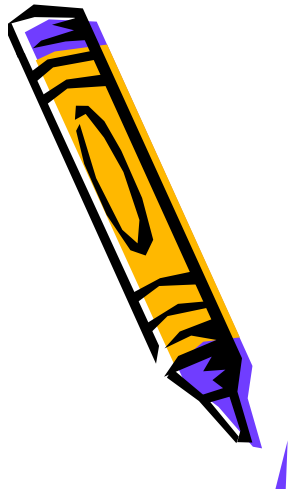
```
UPDATE o_employees
SET phone_number = '360.509.7132'
WHERE employee_id = 210;

UPDATE o_departments
SET location_id = 1700
WHERE department_id= 210;

UPDATE o_employees
SET department_id = 210
WHERE employee_id in (210,211,212);

UPDATE o_employees
SET salary = 5000
WHERE employee_id = 210;

UPDATE o_employees
SET salary = 5100
WHERE employee_id = 211;

DELETE FROM o_employees
WHERE employee_id = 212;
```



```
UPDATE o_employees
SET email ='jjames@charter.net'
WHERE employee_id = 210;
```

```
UPDATE o_employees
SET email ='jjames@charter.net'
WHERE last_name = 'James';
```

Creating Tables



Creating tables is part of SQL's data definition language (DDL). Other DDL statements used to set up, change, and remove data structures from tables include ALTER, DROP, RENAME, and TRUNCATE.

To create a new table, you must have the CREATE TABLE privilege and a storage area for it. The database administrator uses data control language (DCL) statements to grant this privilege to users and assign a storage area.

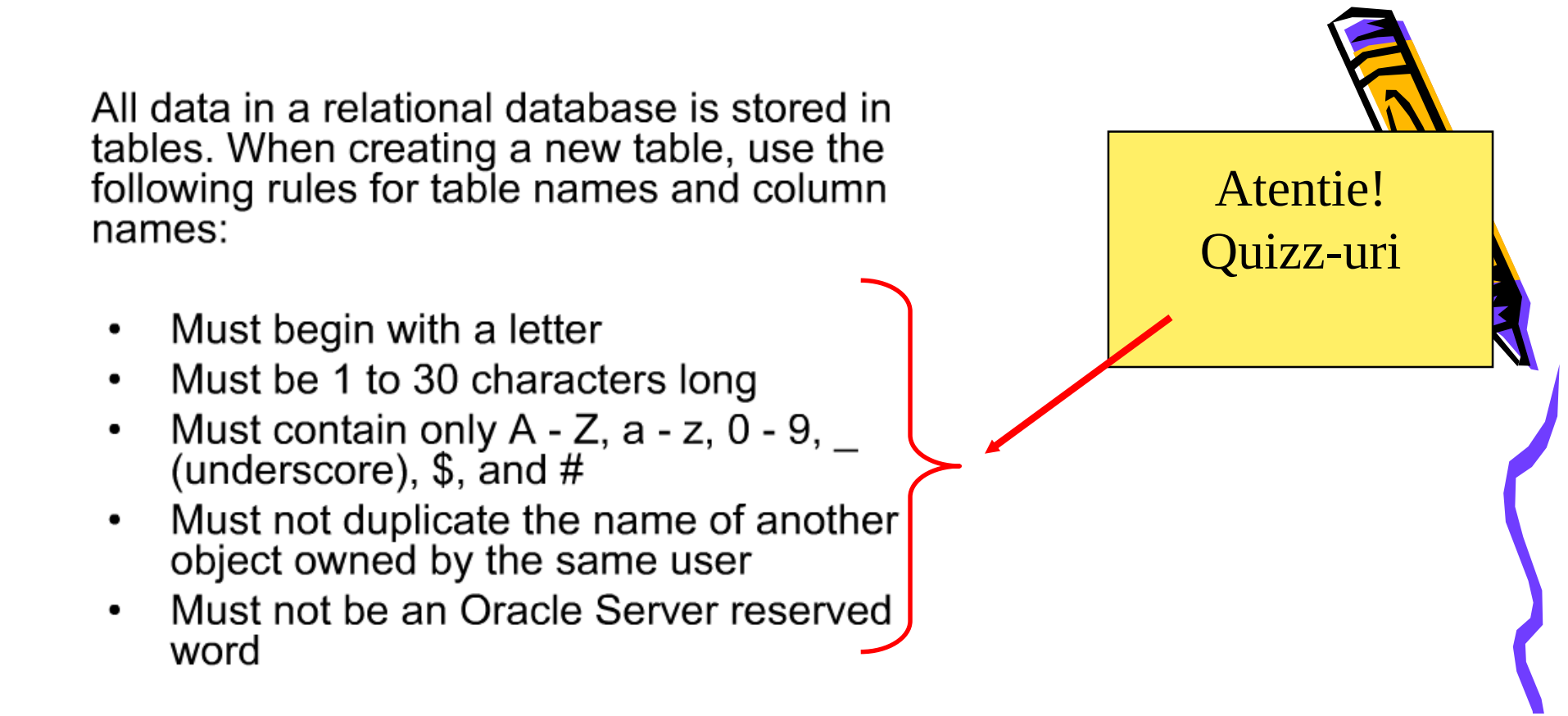
When a copy of a table is made using a subquery, the following rules are important:

- The column names in the new table will be identical to those in the original table, unless column aliases are used
- The column datatypes in the new table will be identical to those in the original table

```
create table
    Copy_employees
as (select * from
    employees)
```

All data in a relational database is stored in tables. When creating a new table, use the following rules for table names and column names:

- Must begin with a letter
- Must be 1 to 30 characters long
- Must contain only A - Z, a - z, 0 - 9, _ (underscore), $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle Server reserved word



NUMELE TABELEI
SA FIE SEMNIFICATIV
NU ESTE CASE SENSITIVE

It is best to use descriptive names for tables and other database objects. If a table will store information about students, name it STUDENTS, not PEOPLE or CHILDREN.

Also, names are not case sensitive. For example, STUDENTS is treated the same as STuDents or students.

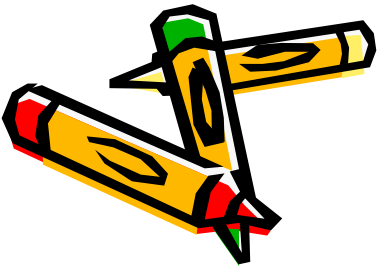
To create a new table consider the following syntax details:

- **table** is the name of the table
- **column** is the name of the column
- **datatype** is the column's data type and length
- **DEFAULT expression** specifies a default value if a value is omitted in the INSERT statement

```
CREATE TABLE table
(column datatype [DEFAULT expression],
(column datatype [DEFAULT expression],
(……[ ] );
```

For example:

```
CREATE TABLE cd_collection
(cd_number      NUMBER(2),
title           VARCHAR2(14),
artist          VARCHAR2(13),
purchase_date   DATE   DEFAULT SYSDATE);
```



If no schema is explicitly included in a CREATE TABLE statement, the table is created in the current user's schema.

Creating A Table Using A Subquery

A second method for creating a table is to apply the AS subquery clause, which both creates the table and inserts rows returned from the subquery.

This is an easy way to create a copy of a table to practice SQL statements. Note that you need to create a column alias for those columns in the subquery that contain expressions.

Only datatype definitions and NOT NULL constraints are passed on to a new table created from a subquery. This is because the new table could be used in a different context, in which existing PK-FK relationships may not be relevant.

```
CREATE TABLE tablename
[(column, column, ...)]
AS subquery;
```

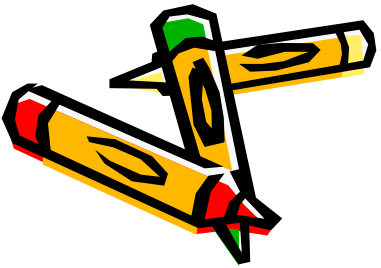
Two examples:

```
CREATE TABLE copy_mytable
AS
(SELECT code, name, start_date,
        end_date, give_away
FROM  f_promotional_menus);


CREATE TABLE dept80
AS
SELECT employee_id, last_name,
       salary*12 ANNSAL,
       hire_date
FROM  employees
WHERE department_id = 80;
```

In tabela noua doar 3 coloane sunt create dar exista toate randurile din tabela STUDENTS.
In tabela nou creata exista doar randurile care indeplinesc conditia ca GPA este mai mare decat 3.
Daca folosesti ca subquery SELECT *, toate randurile si toate coloanele vor exista in tabela nou creata

OBSERVATII/ CREATE TABLE -SUBQUERY

1. Numarul de coloane din tabela destinatie trebuie sa fie egal cu nr.de coloane din tabela sursa
2. Noua tabela este identica cu tabela originala, dar fara restrictiile de integritate din tabela sursa
3. Tipul datelor din sursa va fi copiat in destinatie
4. Daca o coloana din subcerere este o expresie, atunci foloseste alias



CREATING Tables

TRY IT/ SOLVE IT

1. Complete the GRADUATE CANDIDATE table instance chart. Credits is a foreign-key column referencing the requirements table.

Graduate Candidates

Column Name	student_id	last_name	first_name	credits	graduation_date
Key Type					
Nulls/Unique					
FK Column					
Data Type	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE
Length	6			3	

The length values are arbitrary; choose other values if desired.

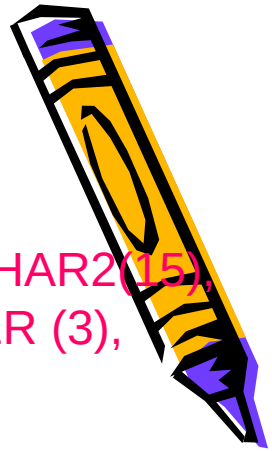
Column Name	student_id	last_name	first_name	credits	graduation_date
Key Type	pk				
Nulls/Unique	uk				
FK Column				fk	
Data Type	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE
Length	6	15	15	3	

TRY IT / SOLVE IT

2. Write the syntax to create the grad_candidates table.

Column Name	student_id	last_name	first_name	credits	graduation_date
Key Type					
Nulls/Unique					
FK Column					
Data Type	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE
Length	6			3	

```
CREATE TABLE grad_candidates
 (student_id NUMBER(6),last_name VARCHAR2(15),
first_name VARCHAR2(15),credits NUMBER (3),
graduation_date DATE);
```



3. Confirm creation of the table using DESCRIBE.

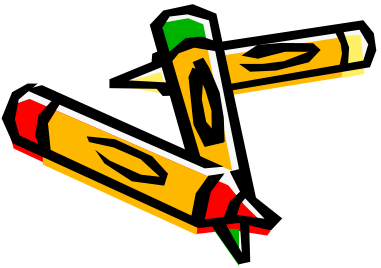
```
DESCRIBE grad_candidates;
```

4. Create a new table using a subquery. Name the new table your last name -- e.g., smith_table. Using a subquery, copy grad_candidates into smith_table.

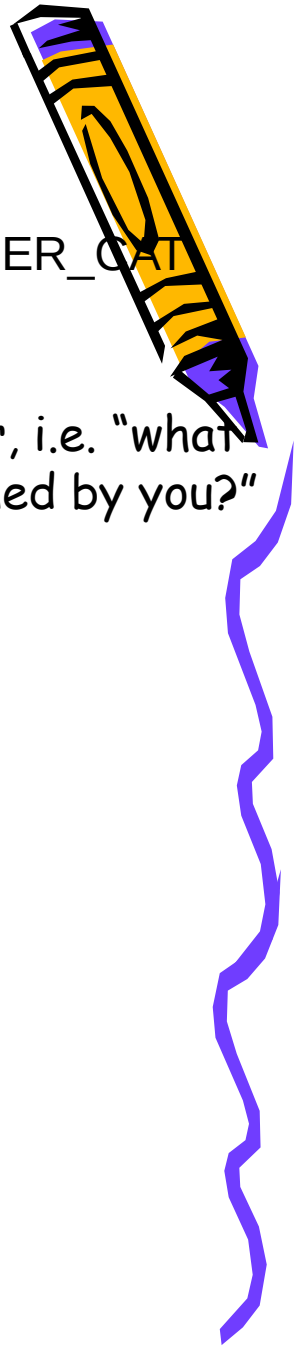
```
CREATE TABLE smith_table AS
(SELECT student_id,last_name,first_name,credits,
graduation_date   FROM grad_candidates);
```

5. Insert your personal data into the smith_table.

```
INSERT INTO smith_table
(student_id,last_name,first_name,credits,graduation_date)
 VALUES ( ANSWERS WILL VARY...)
```



TRY IT / SOLVE IT



6. Query the data dictionary for each of the following:
- USER_TABLES/ - USER_OBJECTS/- USER_CATALOG or USER_CAT

6. Make these questions that the students must answer, i.e. "what code would you use to see the names of all tables owned by you?"

To see the names of tables owned by the user (you):

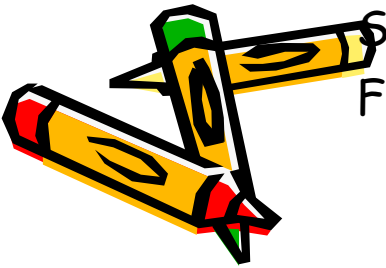
```
SELECT table_name
FROM user_tables
```

To view distinct object types owned by the user:

```
SELECT DISTINCT object_type
FROM user_objects;
```

To view all objects owned by the user:

```
SELECT*
FROM user_catalog;
```



TRY IT / SOLVE IT

1. CREATE a table called Artists. Add the following to the table:
- artist ID
- first name
- last name
- band name
- email
- hourly rate
- song ID from d_songs table

```
CREATE TABLE artists
(artist_id NUMBER(4),
first_name VARCHAR2 (15),
last_name VARCHAR2 (15),
band_name VARCHAR2 (15),
email VARCHAR2(15),
hourly_rate NUMBER (8),
song_id NUMBER(5));
```

2. INSERT one artist from the d_songs table.

3. INSERT one artist of your own choosing; leave song_id blank.

4. ALTER TABLE adding a column of your choice.

5. ALTER TABLE to modify one column.

6. ALTER TABLE to drop one column.

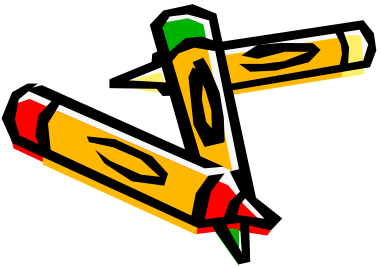
7. ADD COMMENTS to the table.

8. SET one column as UNUSED.

9. RENAME the table.

10. TRUNCATE the table.

11. DROP the table.



DATA DICTIONARY

Fiecare baza de date ORACLE contine un dictionar de date, care se mai numeste "master catalog", ce contine toate tabelele, viziunile(tabele virtuale) si obiectele din baza de date. El insusi este de fapt un set de tabele, deoarece toate datele sunt memorate in tabele.Gandeste un dictionar de date ca fiind "tabele care descriu alte tabele"

The tables and views in the data dictionary contain information about:

- Users and their privileges
- Tables, columns and their data types, integrity constraints, indexes
- Privileges granted on database objects
- Storage structures of the database
- In fact, everything in the database.

The data stored in the data dictionary is also often called "metadata." It consists of two levels: internal and external. The internal level contains tables that are used by the various DBMS software components. These are normally not visible to end users. The external level provides many views on these base tables to access information about objects and structures at different levels of detail.

You can query the data dictionary to view various database objects owned by you. To see all the views available in the data dictionary, use this SQL command:

```
SELECT *
FROM DICTIONARY;
```

The data dictionary tables frequently used are:

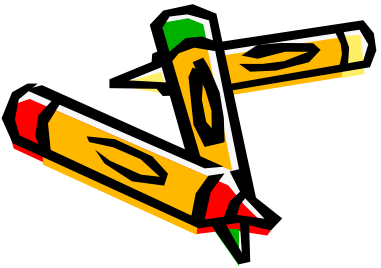
- USER_TABLES
- USER_OBJECTS
- USER_CATALOG or USER_CAT

To see the names of tables owned by the user (you):

```
SELECT table_name
FROM user_tables;
```

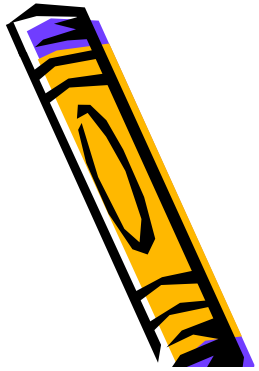
To view distinct object types owned by the user:

```
SELECT DISTINCT object_type
FROM  user_objects;
```



To view all objects owned by the user:

```
SELECT *
FROM user_catalog;
```



Using Data Types

- For character values: CHAR (fixed size, maximum 2000 characters); VARCHAR2 (variable size, maximum 4000 characters); CLOB (variable size, maximum 4 billion characters)
- For number values: NUMBER (variable size, maximum precision 38 digits)
- For date and time values: DATE, TIMESTAMP …., INTERVAL
- For binary values (eg multimedia: JPG, WAV, MP3 and so on): RAW (variable size, maximum 2000 bytes); BLOB(variable size, maximum 4 billion bytes).

- VARCHAR2
  - Examples: Name, Address
- NUMBER
  - Examples: Price, Quantity
- DATE
  - Examples: DOB, Hire Date

- For character values, it is usually better to use VARCHAR2 or CLOB than CHAR, because it saves space and is faster.
- For example, an employee's last name is 'Chang'. In a VARCHAR2(30) column, only the 5 significant characters are stored: C h a n g. But in a CHAR(30) column, 25 trailing spaces would be stored as well, to make a fixed size of 30 characters.
- Number values can be negative as well as positive. For example,NUMBER(6,2) can store any value from +9999.99 down to –9999.99.

EXTENSII ALE TIPULUI DE DATE CALENDARISTIC

The TIMESTAMP data type is an extension of the DATE data type which allows fractions of a second.

Memoreaza datele calendaristice la nivel de nanosecunda (intre 0-9 caractere, implicit 6)

For example, TIMESTAMP(3) allows 3 digits after the whole seconds, allowing values down to milliseconds to be stored.

TIMESTAMP …. WITH [LOCAL] TIME ZONE

TIMESTAMP…. WITH TIME ZONE = TIMESTAMP+ diferenta de fus orar
TIMESTAMP…. WITH LOCAL TIME ZONE = TIMESTAMP normalizat la diferenta de fus orar a bazei de date (returneaza timpul local fata de coordonatele universale)

LOCAL-> cand o coloana este selectata intr-o declaratie, timpul este convertit automat in timpul zonei

TIMESTAMP …. WITH [LOCAL] TIME ZONE

TIMESTAMP WITH TIME ZONE stores a time zone value as a displacement from Universal Coordinated Time or UCT (previously known as Greenwich Mean Time or GMT).

TIMESTAMP …. WITH [LOCAL] TIME ZONE

TIMESTAMP WITH TIME ZONE stores a time zone value as a displacement from Universal Coordinated Time or UCT (previously known as Greenwich Mean Time or GMT).

For example, a value of '21-AUG-03 08:00:00 – 5:00'
means 8:00 am 5 hours behind UTC. This is US Eastern Standard Time (EST).

TIMESTAMP WITH LOCAL TIME ZONE is the same, but with one difference: when this column is SELECTed in a SQL statement, the time is automatically converted to the selecting user's time zone.

UCT- UNIVERSAL COORDINATED TIME
GMT- GREENWICH MEAN TIME
EST- EASTERN STANDARD TIME

```
CREATE TABLE time_example
(first_column      TIMESTAMP WITH TIME ZONE,
 second_column TIMESTAMP WITH LOCAL TIME ZONE);

INSERT INTO time_example (first_column, second_column)
VALUES      ('15-NOV-03 08:00:00  -5', ''15-NOV-03 08:00:00
-5');
```

Both values are stored with a time displacement of –5 hours (EST).
But now a user in Istanbul executes:

```
SELECT * FROM time_example;

FIRST_COLUMN                          SECOND_COLUMN
---------------------------           ------------------------------
15-NOV-03 08.00.00.000000             15-NOV-03
15.00.00.000000
```

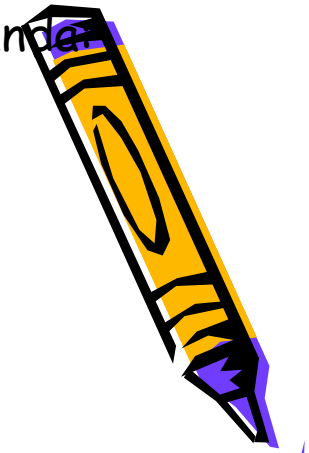
Istanbul time is 7 hours ahead of EST; when it's 8am in New York City, it's 3pm in Istanbul.

ORDER_DATE	DATA
24-MAR-07 10.02.00,000000 AM	24-03-2007

1 rows returned in 0,01 seconds Download

ora Romaniei 18.02
data 24 martie 2007
(returneaza timpul local al serverului de date , fata de coordonatele universale tine cont de diferenta de fus orar)

Să exemplificăm aceste tipuri de date creând o tabelă de test cu comanda:



```
create table test3
(data1 DATE,
data2 TIMESTAMP(5),
data3 TIMESTAMP(5) WITH TIME ZONE,
data4 TIMESTAMP(5) WITH LOCAL TIME ZONE)
```

Vom insera acum o linie nouă în această tabelă:

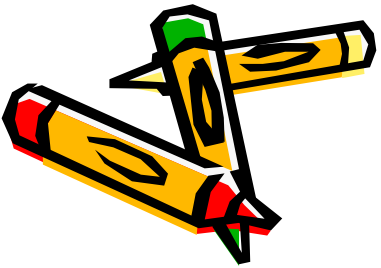
```
insert into test3
values(sysdate,systimestamp,systimestamp,systimestamp)
```

şi la afişarea tabelei

```
select * from test3
```

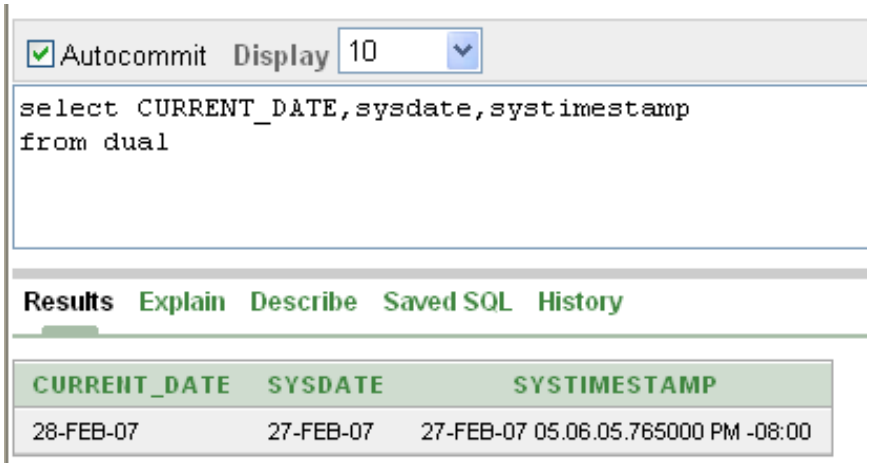
vom obţine

DATA1	DATA2	DATA3	DATA4
27-FEB-07	27-FEB-07 05.43.08.61258 AM	27-FEB-07 05.43.08.61258 AM -06:00	27-FEB-07 11.43.08.61258 AM



Oracle oferă un număr foarte mare de funcţii care operează asupra datelor calendaristice, exemplificând prin cele mai importante dintre acestea.

SYSDATE – returnează data şi ora curentă a serverului bazei de date.
CURRENT_DATE – returnează data şi ora curentă a aplicaţiei client. Aceasta poate să difere de data bazei de date.
SYSTIMESTAMP – returnează data în formatul **TIMESTAMP**.



Funcţiile **SYSDATE**, **CURRENT_DATE** şi **SYSTIMESTAMP**

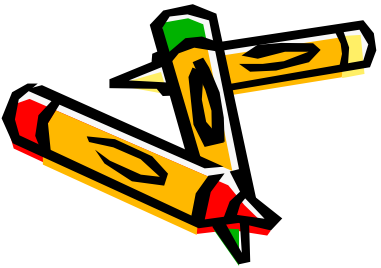
INTERVAL DATA TYPES

Intervalul de timp intre 2 date calendaristice

- INTERVAL YEAR TO MONTH stores a period of time measured in years and months.

- INTERVAL DAY TO SECOND stores a period of time measured in days, hours, minutes and seconds.

```
CREATE TABLE time_example1
(loan_duration   INTERVAL YEAR(3) TO MONTH,
day_duration     INTERVAL DAY(3) TO SECOND);
```



SINTAXA

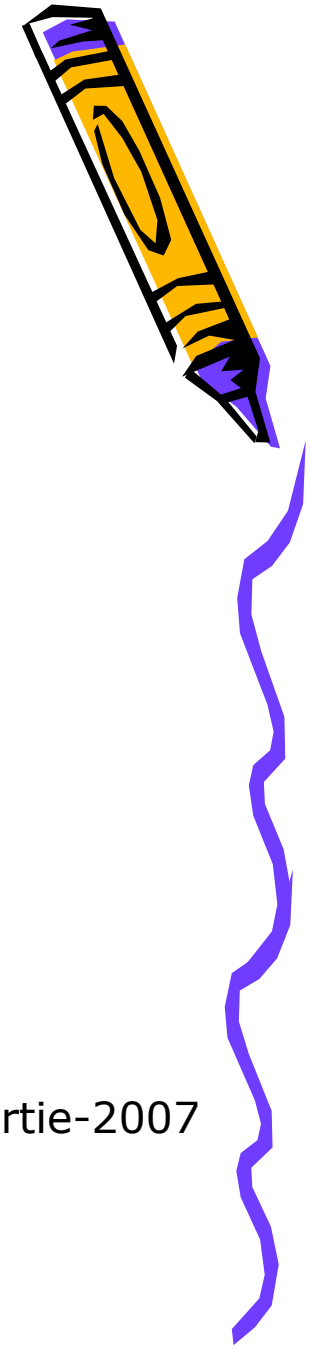
INTERVAL YEAR
[(year_precision)] TO MONTH

INTERVAL YEAR TO MONTH: Stores a period of time in years/months

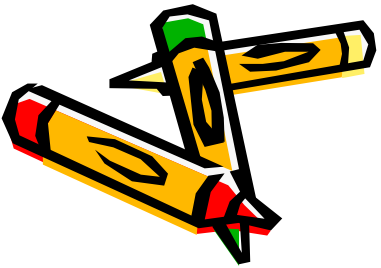
```
CREATE TABLE time_ex2(
  school_year_duration INTERVAL YEAR(3) TO MONTH)

INSERT INTO time_ex2(school_year_duration) VALUES
 (INTERVAL '9' MONTH(3))

SELECT TO_CHAR(sysdate + school_year_duration, 'dd-Mon-
  yyyy')
 FROM time_ex2
```

TO_CHAR(SCHOOL_YEAR_DURATION+SYSDATE,'DD-MON-YYYY')
24-Dec-2007

1 rows returned in 0,01 seconds Download



Retine o perioada de timp in ani si luni.
(3) Inseamna precizia datei
Interval('9') este un interval de 9 luni.
Rezultatul este 9 luni de la data de azi : 24-martie-2007

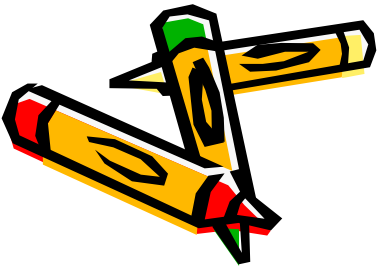


INTERVAL DAY TO SECOND: Stores a more precise period of time (days/hours/minutes/seconds)

CREATE TABLE time_ex3(day_duration INTERVAL DAY (3) TO SECOND)

INSERT INTO time_ex3(day_duration) VALUES (INTERVAL '180' DAY(3))

SELECT sysdate + day_duration "Half Year" FROM time_ex3



INTERVAL DAY … TO SECOND

Use this when you need a more precise difference between two date-time values.

The data type syntax is:

INTERVAL DAY
[(day_precision)]
TO SECOND
[(fractional_seconds_precision)]

day_precision is the maximum number of digits in the DAY date-time field. The default is 2.

fractional_seconds_precision is the number of digits in the fractional part of the SECOND date-time field. The default is 6.

1.You need to store the SEASONAL data in months and years. Which data type should you use?

INTERVAL YEAR TO MONTH

2. To store time with fractions of seconds, which datatype should be used for a table column?

TIMESTAMP

3. Data in the EXEMPLU_TIME column needs to be stored in days, hours, minutes and seconds. Which data type should you use?

INTERVAL DAY TO SECOND

4. Evaluate this CREATE TABLE statement:

```
CREATE TABLE sales
( sales_id NUMBER(9),
customer_id NUMBER(9),
employee_id NUMBER(9),
description VARCHAR2(30),
sale_date TIMESTAMP WITH LOCAL TIME ZONE DEFAULT SYSDATE,
sale_amount NUMBER(7,2));
```

Which business requirement will this statement accomplish?

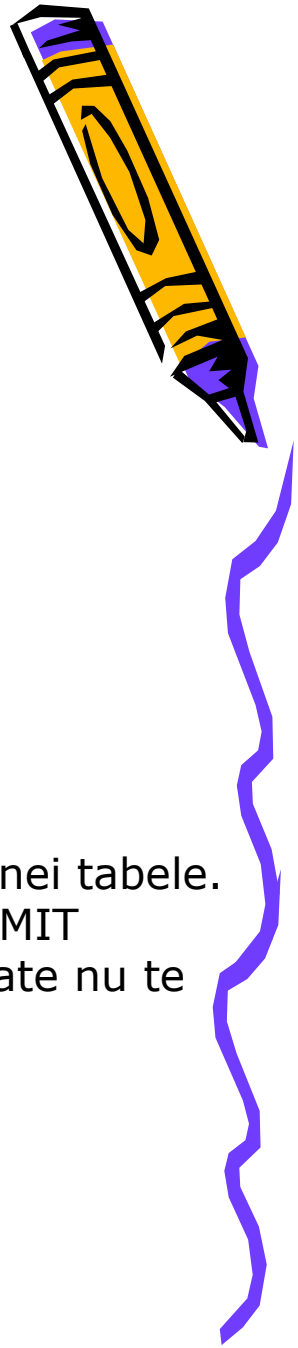
Sales identification values could be either numbers or characters, or a combination of both.

All employee identification values are only 6 digits so the column should be variable in length.

Description values can range from 0 to 30 characters so the column should be fixed in length.

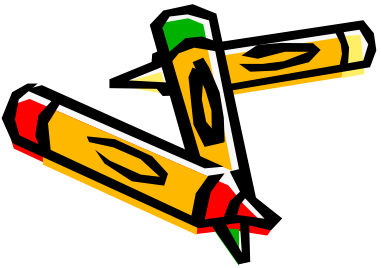
Today's date should be used if no value is provided for the sale date. (*)

DDL – Data Definition Language



- ALTER TABLE
- DROP TABLE
- RENAME
- TRUNCATE
- COMMENT

Sunt comenzi care permit modificarea structurii unei tabele. Acestea sunt intotdeauna IREVERSIBILE (au COMMIT automat, tranzactia este finalizata), odata executate nu te mai poti intoarce la structura anterioara.



Modifying a Table

ALTER TABLE

- Use ALTER TABLE TO:
  - ADD a new column (Always goes at the end of the table)
  - MODIFY an existing column (Change data type, size, or default value – see restrictions)
  - Define a DEFAULT value for a column
  - DROP COLUMN (Cannot drop all columns)
  - SET UNUSED / DROP UNUSED

You can add or modify a column in a table, but you cannot specify where the column appears. A newly added column always becomes the last column of the table.

Also, if a table already has rows of data and you add a new column to the table, the new column is initially null for all the rows.

Atentie!

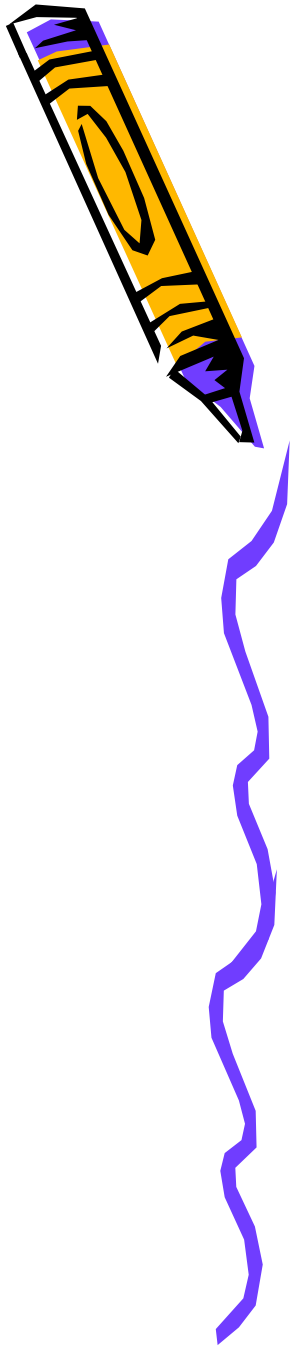
MODIFY nu modifica datele din coloana ci structura acesteia (tip, dimensiune, valoarea implicita) Datele din coloana trebuie sa se portiveasca cu noua structura, altfel va da o eroare

DEFAULT defineste o valoare implicita pentru o coloana ce nu a avut DEFAULT creat

SET UNUSED nu mai permite accsesul la coloana specificata

Datele sunt tot acolo, dar neutilizabile. Cu DROP UNUSED vor fi sterse definitiv coloanele setate enterior cu SET UNUSED.

ADDING A COLUMN

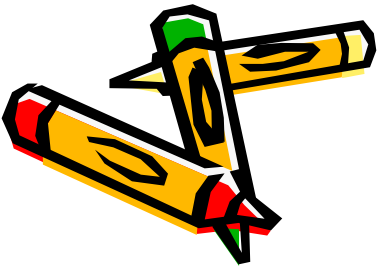


ALTER TABLE tablename
ADD (column name datatype [DEFAULT expression],
column name datatype [DEFAULT expression], ...

For example:

ALTER TABLE copy_f_staffs
ADD (hire_date DATE DEFAULT SYSDATE);

ALTER TABLE copy_f_staffs
ADD (e_mail_address VARCHAR2(80));

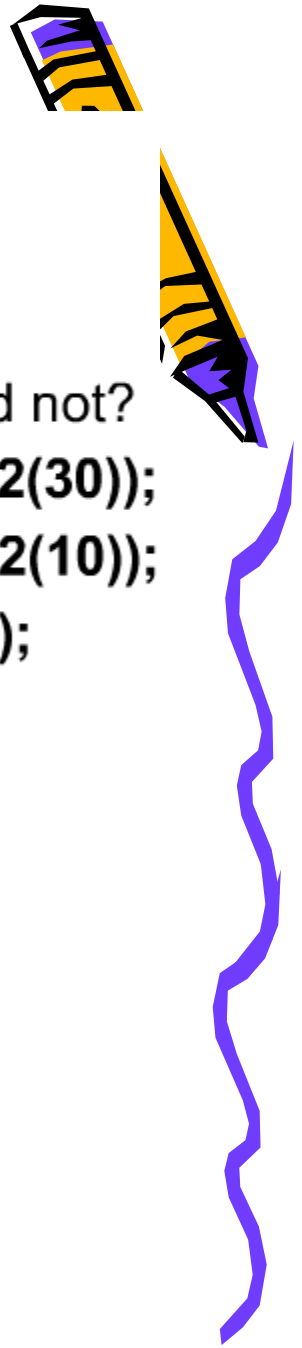


ALTER TABLE: MODIFYING A COLUMN

Modifying a column can include changes to a column's data type, size, and DEFAULT value. Rules and restrictions when modifying a column are:

- You can increase the width or precision of a numeric column.
- You can increase the width of a character column.
- You can decrease the width of a column only if the column contains only null values or if the table has no rows.

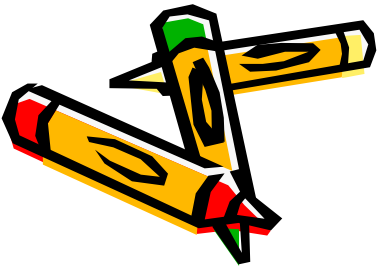
- You can change the data type only if the column contains null values.
- You can convert a CHAR column to VARCHAR2 or convert a VARCHAR2 column to CHAR only if the column contains null values or if you do not change the size.
- A change to the DEFAULT value of a column affects only later insertions to the table.



```
CREATE TABLE mod_emp
  (last_name      VARCHAR2(20),
   salary          NUMBER(8,2));
```

Which of these modifications would be allowed, and which would not?

1. **ALTER TABLE mod_emp MODIFY (last_name VARCHAR2(30));**
2. **ALTER TABLE mod_emp MODIFY (last_name VARCHAR2(10));**
3. **ALTER TABLE mod_emp MODIFY (salary NUMBER(10,2));**
4. **ALTER TABLE mod_emp MODIFY (salary NUMBER(8,2) DEFAULT 50);**



DROPPING A COLUMN

SQL syntax:

ALTER TABLE tablename DROP COLUMN column name;

When dropping a column the following rules apply:

- A column to be dropped may or may not contain data.
- Only one column can be dropped at a time.
- You can't drop all of the columns in a table; at least one column must remain.
- Once a column is dropped, the data values in it cannot be recovered.

For example:

ALTER TABLE copy_f_staffs DROP COLUMN manager_target;

SET UNUSED COLUMNS

Dropping a column from a large table can take a long time. A quicker alternative is to mark the column as unusable. The column values remain in the database but cannot be accessed in any way, so the effect is the same as dropping the column.

Syntax:

ALTER TABLE tablename SET UNUSED (column name);

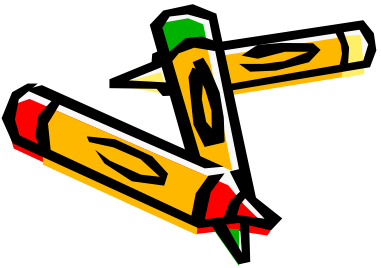
In fact, you could add a new column to the database with the same name as the unused column. The unused columns are there, but invisible!

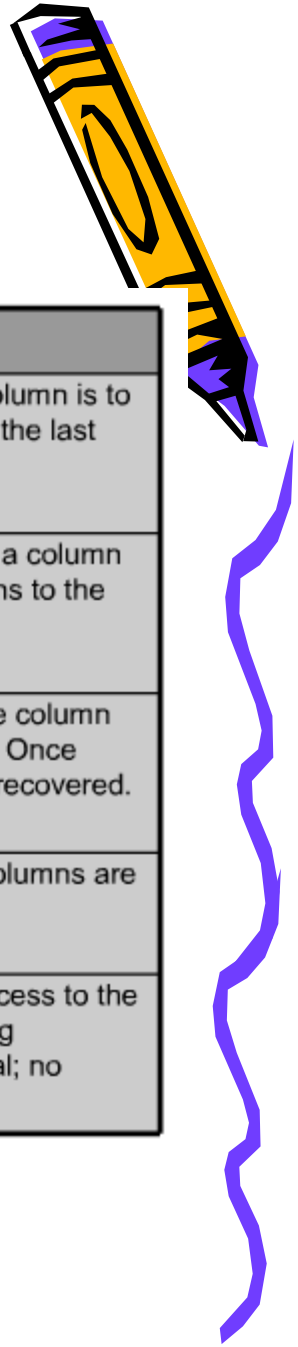
Example:

```
ALTER TABLE copy_f_staffs
    SET UNUSED (manager_budget);
```

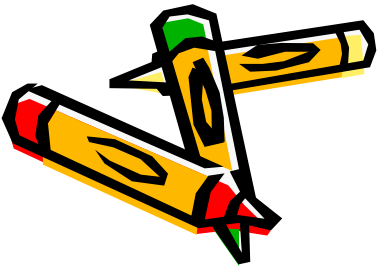
DROP UNUSED COLUMNS removes all columns currently marked as unused. You use this statement when you want to reclaim the extra disk space from unused columns in a table.
ALTER TABLE tablename DROP UNUSED COLUMNS;

Example:
ALTER TABLE copy_f_staffs DROP UNUSED COLUMNS;



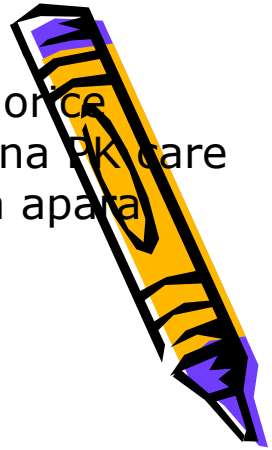


Syntax	Outcomes	Concerns
ALTER TABLE tablename ADD (column name datatype [DEFAULT expression], column name datatype [DEFAULT expression], …	Adds a new column to a table	You cannot specify where the column is to appear in the table. It becomes the last column.
ALTER TABLE tablename MODIFY (column name datatype [DEFAULT expression], column name datatype, …	Used to change a column's datatype, size, and default value	A change to the default value of a column affects only subsequent insertions to the table.
ALTER TABLE tablename DROP COLUMN column name;	Used to drop a column from a table	The table must have at least one column remaining in it after it is altered. Once dropped, the column cannot be recovered.
ALTER TABLE tablename SET UNUSED (column name);	Used to mark one or more columns so they can be dropped later	Does not restore disk space. Columns are treated as if they were dropped.
ALTER TABLE tablename DROP UNUSED COLUMNS	Removes from the table all columns currently marked as unused	Once set unused, there is no access to the columns; no data displayed using DESCRIBE. Permanent removal; no rollback.



DROP TABLE

DROP TABLE nu functioneaza in orice conditii. Daca tabela are o coloana PK care este FK intr-o alta tabela pot sa apara probleme.



- Removes all reference to a table
- All data is deleted
- It is irreversible!

Example:

DROP TABLE copy_items

TRUNCATE

- Removes all rows in a table
- Releases storage space

Example:

TRUNCATE TABLE copy_items

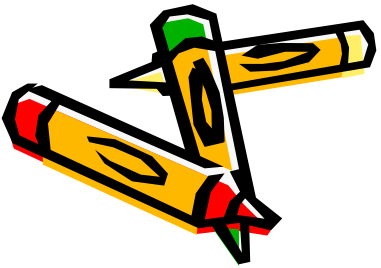
TRUNCATE sterge toate liniile mult mai rapid decat DELETE si elibereaza spatiul de memorie

RENAME

- Changes the name of a table

Example:

RENAME TABLE copy_items
TO inventory_items



- You cannot roll back row removal.
- You must be the owner of the table or have been given DELETE ANY TABLE system privileges.

COMMENT ON TABLE

Syntax:
COMMENT ON TABLE tablename | COLUMN table.column
IS 'place your comment here';

Add a comment up to 2,000 bytes

Example:
COMMENT ON TABLE employees
IS 'Western Region only';

SELECT table_name, comments FROM user_tab_comments;

TABLE_NAME	COMMENTS
EMPLOYEES	Western Region only

If you want to drop a comment previously made on a table or column, use the empty string (''):
COMMENT ON TABLE employees IS '' ;

Permite adaugarea unui comentariu despre structura tabelei.

Se poate face un comentariu la nivel de tabela sau la nivel de coloana. In al doilea caz, coloana trebuie sa fie prefixata de numele tabelei din care provine.

Orice comentariu poate fi vizualizat cu comanda DESCRIBE numele_tabelei.

TRY IT / SOLVE IT

1. In your o_employees table, enter a new column called "Termination." The data type for the new column should be VARCHAR2. Set the DEFAULT for this column as SYSDATE to appear as character data in the format: February 20th, 2003.

```
ALTER TABLE o_employees
ADD (termination VARCHAR2(20) DEFAULT
TO_CHAR(SYSDATE, 'Month ddth, YYYY'));
```

2. DELETE all employees in the o_employees table whose employee_id's are between 100 and 150.

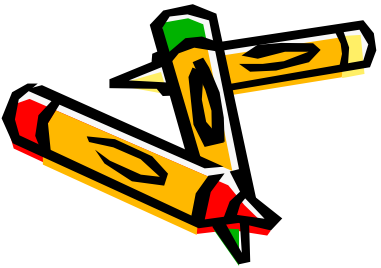
```
DELETE FROM o_employees
WHERE employee_id BETWEEN 100 AND 150;
```

3. Employee William Gietz retired on August 1, 2004. Change his termination date in your o_employees table.

```
UPDATE o_employees
SET termination = '01-AUG-04'
WHERE employee_id = 206;
```

4. Create a new column in the o_employees table called start_date. Use the TIMESTAMP WITH LOCAL TIME ZONE as the data type

```
ALTER TABLE o_employees
ADD start_date TIMESTAMP WITH LOCAL TIME ZONE
```



TRY IT / SOLVE IT

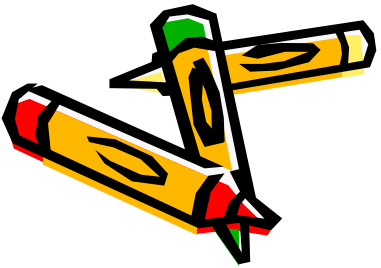
5. Add new employee Amy Kimura to the Human Resources department. Her employee ID number is 220. Her hire date was September 15, 2004, and her start date was September 29, 2004, at 8:30:00 a.m.

```
INSERT INTO o_employees (employee_id, first_name,
last_name, email, hire_date, job_id, start_date)
VALUES(220, 'Amy', 'Kimura','AKIMURA','15-SEP-
04','HR_MAN','29-SEP-04 08:30:00 AM');
```

6. Set the column commission_pct to UNUSED.
After executing the UNUSED command, issue a SELECT * command. What effect did this have on the column?

```
ALTER TABLE o_employees
SET UNUSED (commission_pct);
```

The commission_pct column is marked so it can be dropped when system resources are lower. This column is treated as if it were dropped, even though the column data remains in the table's rows. Once marked UNUSED, this column will return no data when you run a select statement against it.



TRY IT / SOLVE IT

7. Drop the commission_pct column.

```
ALTER TABLE o_employees
DROP UNUSED COLUMNS
```

8. Add the following comment to the o_jobs table:
"New job description added"
View the data dictionary to view your comments.

```
COMMENT ON TABLE o_jobs
IS 'New job description added';

SELECT *
FROM USER_TAB_COMMENTS;
```

9. Rename the o_jobs table to o_job_description.

```
RENAME o_jobs to o_job_description
```

10. Truncate the o_job_description table. Then do a SELECT * statement. Are the columns still there? Is the data still there?

```
TRUNCATE TABLE o_job_description;
```

The TRUNCATE TABLE command removes all rows from the table and releases storage space used by the table.